Kumulative Habilitation eingereicht an der
Technischen Fakultät der
Albert-Ludwigs-Universität Freiburg im Breisgau

Spatial Modeling and Robot Navigation

Dr. Cyrill Stachniss

August 1, 2009

Abstract

This habilitation thesis deals with the broad topic of robot navigation and with its foundations in spatial modeling, scene understanding, and action generation. Robot navigation is the process of autonomously making a sequence of decisions that allows a mobile robot to travel robustly to selected locations in the environment. This ability involves a large set of problems that need to be solved including sensor data interpretation, state estimation, environment modeling, scene understanding, learning, coordination, and motion planning. The ability to navigate autonomously has special practical relevance, since navigations tasks are embedded in most robotic applications.

In this work, we present innovative solutions for acquiring models of the environment with mobile robots and for robustly navigating using these models. A representation of the environment is needed for a wide range of robotic applications. Here, a model of the environment representing its geometry is only a starting point. To allow for intelligent decision making or higher level functionalities, a robot needs information about the topology of the space as well as knowledge about objects in the scene in order to act robustly and interact in real world settings. These problems are addressed in this work. We furthermore present novel approaches for decision making, trajectory planning, localization, exploration, and related topics. We propose probabilistic solutions to a variety of these problems and consider different types of robots with different sensing modalities. This includes classical wheeled robots as well as computer-controlled cars, flying vehicles, and humanoid robots. Our solutions are important building blocks that enable robots to operate autonomously and effectively in the real world.

The contributions of this habilitation thesis are innovative solutions to a variety of open problems in the context of model learning and efficient robot navigation which captured the attention of research community to-date. Most of the presented approaches are of probabilistic nature, they explicitly consider uncertainty, and include learning techniques. All developed approaches have been applied and evaluated on real mobile robots in realistic settings, have proven robustness, and have shown significant improvements over state-of-the-art methods in robotics.

Contents

Part I

Summary

Chapter 1

Introduction

One of the main goals in service robotics is to develop truly autonomous robots that support us humans. Home environments are envisioned as one of the key application areas for service robots. Robots operating in such environments are typically faced with a variety of problems that result from the rather unstructured surroundings compared to industrial applications.

The ability to model the surrounding space and at the same time to navigate autonomously and reliably has significant practical relevance – not only in domestic service robotics. In most robotics applications, autonomous navigation is an integral part of the robot's mission. Most tasks can only be carried out if enough information about the scene the robot operates in is available. Ideally, a robot is able to autonomously infer appropriate models of the environment by observation, without requiring manual teaching.

The contribution of this habilitation thesis are solutions to various problems in the context of model learning and robot navigation. Here, the term navigation does not refer to path planning only. It includes state estimation, localization, i.e., estimating the position of the robot given a map of the environment, exploration, i.e., steering a robot so that it can efficiently construct accurate models of the space, as well as path planning and controlling the motion of robotic actuators such as manipulators.

In this work, we present methods that enable robots to achieve such desired capabilities. All techniques developed in the context of this habilitation thesis have been evaluated on physical robots and the experiments, that are presented in the individual papers, illustrate the achievements over the state-of-the-art. The individual scientific contributions can furthermore be seen as building blocks for achieving integrated autonomous systems. Some of the presented techniques are available as open source implementations to the community and are frequently used by other research groups. Furthermore, the majority of recorded robotic datasets is provided to the public.

In this cumulative habilitation thesis, the citations [J1]-[J9] refer to journal publications whereas [C1]-[C25] refer to conference papers and [W1]-[W5] to reviewed workshop papers respectively. All citations with numbers only refer to other publications. The summary of our publications is organized as follows.

Chapter 2: Mapping Environments with Mobile Robots

We summarize our novel solutions to the problem of acquiring the geometric representation of the environment in view of the uncertainty of sensor data and the robot's position. This is often referred to as the simultaneous localization and mapping (SLAM) problem. SLAM is a "chicken-or-egg" problem since a map is needed for localization while a good pose estimate is required to build a map. We will present efficient techniques for constructing maps from sensor data in two-dimensional as well as in three-dimensional scenes using different types of sensors and robotic platforms.

Our *key contributions* in the context of SLAM are:

1. An improved technique for computing 2D grid maps using a particle filter. The approach extends our previously presented method for particle filter-based SLAM and provides an analysis of the often made assumption of Gaussian proposal distributions in this context. Our approach overcomes limitations imposed by the Gaussians while being computationally as efficient as an approach using a Gaussian proposal.
2. A learning approach for bridging the gap between feature-based and grid-based SLAM. By means of reinforcement learning, we enable a robot to estimate which underlying representation of its surroundings is best suited to correct the pose of the robot. The resulting, adaptive approach is able to switch in each step of the algorithm between a feature and a grid map in order to use the best of both worlds.
3. An innovative approach to actively deciding whether observed features should be considered during SLAM or neglected for reasons of efficiency. The approach, which based on reinforcement learning, allows robots with highly limited computational resources to solve navigation tasks in unknown environments.
4. A highly efficient method for computing maximum likelihood maps estimated by means of pose-graph optimization. Our technique, which is inspired by stochastic gradient descent, allows for correcting large errors in the trajectory estimate of the robot and can deal with bad initial guesses. This method is one of the most efficient approaches in the SLAM community at the moment.
5. A framework for objectively measuring the performance of different SLAM algorithms based on the notation of relative errors.

Chapter 3: Localization

This chapter covers the pose estimation problem given a model of the environment. Pose estimation refers to the problem of computing and updating the position and orientation of the robot relative to a given map of the environment. We present novel sensor models for probabilistic Monte-Carlo localization to exploit the properties of the sensors that are often used in robotics in order to improve the localization performance. In contrast to Chapter 4, we focus here on robots equipped with laser range finders.

Our *key contributions* in the context of robot localization are novel methods for obtaining better sensor models which can be applied in the popular Monte-Carlo localization frame-

work. We show how the localization performance can be significantly improved by

1. exploiting specific sensor properties as well as surface properties of scanned objects and
2. explictly considering dependencies of nearby laser range observations within one scan.

Chapter 4: Special Sensing Modalities

Laser range finders are probably the most frequently used sensors in the robotics community. In this chapter, we will present innovative solutions to problems that arise when other sensing modalities are used. First, we focus on cameras as alternative sensors to laser range finders, second, we consider gas concentration sensors for acquiring gas distribution maps.

Vision sensors become more and more popular in robotics, as they are at the same time cheap, light-weight, and they constitute a rich source of information about the environment. Monocular cameras, however, do not directly provide proximity information, which is often required by traditional robotic state estimation approaches. We present solutions for coping with visual data in the context of robotics. This includes metric localization of a robot based on monocular image data and the estimation of proximity information from single images in indoor environments. The latter allows mobile robots to avoid using laser range finders while being able to estimate the extends of free space in the surroundings. We furthermore illustrate how to exploit image databases like Flickr.com as an additional source of information for a robot. Based on geo-referenced and labeled photographs, we show how to localize pictured objects such as buildings in the scene.

Besides the use of cameras, we describe a novel approach to estimating gas distributions using a mobile robot equipped with gas sensors. This is an important task for robots deployed for pollution monitoring, surveillance of industrial facilities, or inspection of contaminated areas. The gas distribution modeling problem has a different nature than most other robotic perception problems due to the specific characteristics of gas distributions and their chaotic structure. By modeling gas distributions with a probabilistic regression technique, we are able to learn predictive models for gas distributions.

Our *key contributions* presented in this chapter are:

1. A technique for metrically localizing a robot using only a monocular camera. The presented technique is based on the Monte-Carlo localization framework and proposes a sensor model that does not require a fixed data association for the observed visual features. As a result, a robust, vision-only localization technique for a humanoid robot was successfully developed.
2. A new approach for estimating proximity information based on omnidirectional images similar to the results obtained by traditional range finders. The approach uses Gaussian process regression in combination with dimensionality reduction to establish a correspondence between the image data and proximity information obtained on a training set. We learn a regressor that can be used on robots not equipped with laser range finders for mapping, localization, or navigation. The maps obtained with our

approach show a quality that is roughly comparable to maps build with robotic sonar sensors.

3. An application of the optimization solution to the SLAM problem presented in Chapter 2 to flying vehicles using a down-looking camera. We present a so-called SLAM front-end that estimates incremental motion constraints as well as loop-closing constraints from image data. The resulting system is easy to implement and has successfully been applied to different light-weight flying platforms.
4. An approach to estimating the location of visual landmarks in an environment based on geo-referenced photographs. This is a step towards using tagged, geo-referenced images, for example, obtained from Flickr.com or similar databases to estimate the location of the pictured objects. The approach applies RPROP-based optimization to find the landmark locations that best explain the data. Examples are shown for buildings located in Freiburg downtown.
5. A regression-based approach to gas distribution modeling via a Gaussian process mixture model and an efficient way to learn gas distribution maps with this approach. The mixture components are used to explictly consider the effect of local patches within the in general chaotic structure of gas distributions. We show that the resulting predictive models perform significantly better than existing gas distribution modeling approaches used in the robotics community.

Chapter 5: Towards Understanding Environments

To go beyond the aspects of pure geometry, we present our solution to the problem of learning a hybrid model of the space covering the topology and geometry of an environment as well as semantic information. Here, semantic information refers to things such as labels, vegetation, or specific objects. This hybrid representation can be used subsequently to instruct robots in a natural way by humans. We furthermore address aspects of the scene analysis problem focusing on objects in the surroundings of the robot. We will present solutions for classifying objects observed with a laser range finder in an unsupervised way and show how to learn actuation models for observed objects.

Our *key contributions* towards understanding environments are innovative approaches for:

1. Learning hybrid representations of environments modeling the topology of the scene, its geometry, as well as semantic information. The approach utilizes findings presented in Chapter 2 and estimates semantic labels for places in the scene by means of the AdaBoost algorithm in combination with simple features extracted from sensor data. By smoothing the obtained labels using probabilistic relaxation labeling and a region extraction method, the topology of the scene can be inferred.
2. Estimating which parts of an outdoor environment contain vegetation. This is a valuable information for robots navigating outdoors. By utilizing the remission values of laser range data in combination with the proximity information, we learn classifiers using the support vector machine framework. By analyzing vibrations measured with an

inertial measurement unit, we furthermore found an efficient way for easily generating a large amount of training data for the learning phase.

3. Classifying objects perceived with a 3D laser range scanner in an unsupervised fashion. By means of latent Dirichlet allocation, distributions over objects in a scene can be determined by a mobile robot without supervision. These distributions allow for grouping unknown objects and for assigning new objects to an existing categorization. We additionally show how tactile sensors can be used to classify and identify objects.
4. Modeling articulated objects that a robot observes in typical domestic environments such as doors, cabinets with drawers, or other collections of moving parts. Only based on observations, we are able to learn kinematic models by inferring the connectivity of rigid parts and the articulation models for the corresponding links. Our method uses a mixture of parameterized and parameter-free Gaussian process representations and finds low-dimensional manifolds that provide the best explanation of the given observations. The models in turn can be used predict the motion of object parts to facilitate actions such as grasping or planning manipulation trajectories.
5. Identifying objects based on tactile information. Our approach applies two arrays of pressure sensors in combination with the bag-of-features approach often used in computer vision to distinguish and identify objects grasped by a robot.

Chapter 6: Action Selection for Navigation

This chapter covers different robotic problems in which the central aspect is the action a single robot or a team of robots has to carry out to efficiently solve the given task. In detail, we focus on single and multi-robot exploration, navigation amongst deformable objects, autonomous computer-controlled cars, and the imitation of a human demonstrator for learning manipulative tasks by demonstration.

First, we present our solutions to enable robots to autonomously explore an unknown environment in an effective way. Here, we address the single and the multi-robot exploration problem, presenting information-theoretic concepts for exploring the three-dimensional space and a solution for effectively coordinating a team of exploring robots.

Second, we introduce a new method to deal with environments that contain non-rigid objects. Deformable objects that frequently occur in real world environments are, for example, curtains or plants. We present a technique that considers the deformability of objects during robot navigation.

Third, we address autonomous cars. Computer-controlled cars represent an application area that is closely related to robotics since similar state estimation and planning problems need to be solved under tight time constraints. We present our autonomous Smart car that is able to drive autonomously on streets and in rough terrain.

Finally, we address the problem of teaching a robot manipulation tasks. In our system, a robot is able to derive an abstract task description by observing a human demonstrator. A manipulation robot is then able to reproduce the observed task even in modified settings.

Our *key contributions* in the context of decision making for autonomous navigation are techniques for

1. Transferring our previously developed information-gain driven exploration technique for two-dimensional scenes into the three-dimensional world. Our approach allows an exploring robot to estimate the potential gain of future observation. It reasons about potential observations it may obtain when carrying out an action. This leads to strategies that effectively reduce the uncertainty in the robot's belief.
2. Efficiently coordinating a team of robots by an improved coordination mechanism. The approach exploits the structure of indoor environments and segments the space to generate areas which the individual robots have to explore separately. The segment-based assignment is then solved by means of the Hungarian method. This approach reduces the amount of redundant work and the risk of interference between robots and thus yields a more efficient exploration strategy.
3. Path planning and collision avoidance for environments containing deformable objects. Our planning method operates in combination with an existing physical simulation engine to estimate the deformations caused by the interaction between robots and the objects in the scene. Besides efficiently answering path queries, the approach can avoid collisions with unknown dynamic and potentially rigid obstacles.
4. Autonomously driving a computer-controlled car. We present our development of a real, autonomous car that is fully controlled by a set of computers, perceives the environment based on its sensors, builds three-dimensional models of the scene, accurately localizes itself, plans and executes trajectories in previously unknown or partially known environments.
5. Instructing a robot by demonstration. Our contribution here is a probabilistic framework based on a dynamic Bayesian network for learning manipulative tasks. By repeatedly observing a demonstrator, a robot can infer which parts of a manipulative task are relevant for describing the abstract action. The approach is able to successfully reproduce tasks such as pick and place tasks or cleaning a whiteboard based on a few observation sequences only.

Chapter 7: Conclusion

Finally, we will conclude this habilitation thesis followed by the publications submitted for this cumulative habilitation.

Chapter 2

Mapping Environments with Mobile Robots

Models of the environment are needed for a wide range of robotic applications, including transportation tasks, search and rescue, and efficient vacuum cleaning. Learning maps has therefore been a major research focus in the robotics community over the last decades. The key problem when learning maps with mobile robot is the uncertainty in the pose of the robot during data acquisition as well as the noise in the sensor used to observe the environment. In the literature, the mobile robot mapping problem under pose uncertainty is often referred to as the *simultaneous localization and mapping* (SLAM) or *concurrent mapping and localization* (CML) problem [19; 20; 23; 35; 37; 55; 57; 58; 77; 49]. SLAM is considered to be a complex problem because to localize itself a robot needs a consistent map of the scene while at the same time for acquiring the map the robot requires a good estimate of its own pose. This mutual dependency among the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space.

2.1 Map Learning by Means of Rao-Blackwellized Particle Filters

Efficient mapping system that apply Rao-Blackwellized particle filters to maintain a joint posterior about the trajectory of the robot and the map of the environment typically assume a Gaussian proposal distribution [5; 33; 37; 55; 57; 56; 70]. A proposal distribution inside a particle filter is used to efficiently draw the next generation of samples. In general, the closer the proposal is to the target distribution, the more efficient is the algorithm.

Based on a previously developed SLAM system that uses such a Gaussian proposals and has been published in the PhD thesis [70], we investigate a way to determine how often the observation likelihood function from which the proposal is typically derived is actually Gaussian or not [C18]. We show that in most cases the Gaussian assumption is a valid approximation, however, in around 3% to 6% of all cases, the distribution is clearly multi-modal. We therefore present in [C18] an alternative sampling strategy based on a two-step sampling procedure that is able to accurately

cover the different modes of the observation likelihood function while being as effective as the Gaussian proposal.

In addition to that, we propose an orthogonal optimization to the algorithm which allows particles with similar trajectory estimates to share a map of the environment [J9], [C24]. In the typical formulation of the Rao-Blackwellized particle filter for mapping, each sample has to maintain its own model of the world conditioned on the trajectory estimate. Depending on the number of samples needed to build accurate maps, this can introduce large memory requirements. Our method overcomes this serious shortcoming of Rao-Blackwellized particle filter-based approaches and allows practical implementations to maintain around one order of magnitude more samples while requiring approximatively the same run time.

We furthermore illustrate how such kind of mapping techniques can be adapted in order to be applicable for legged robots. Especially humanoids have become a popular research platform in the robotics community. Compared to wheeled vehicles, they have several drawbacks such as stability problems, limited payload capabilities, violation of the flat world assumption, and they typically provide only very rough odometry information, if at all. In our approach [C14], we investigate the problem of learning accurate grid maps with humanoid robots by adapting the previously described mapping approach to dealing with some of the above-mentioned difficulties. As a result, our approach is able to robustly learn accurate medium-sized maps with a humanoid equipped with a laser range finder. We present an experiment in which our mapping system builds highly accurate maps using data acquired with a humanoid in our office environment containing two loops. The resulting maps have a similar accuracy as maps built with a wheeled robot.

2.2 Bridging the Gap between Grid-based and Feature-based SLAM

One important design decision for the development of autonomously navigating mobile robots is the choice of the representation of the environment. This includes the question which type of features should be used or whether a dense representation such as an occupancy grid map is more appropriate. We present an approach [J5], [C21] which performs SLAM using multiple representations of the environment simultaneously. It uses reinforcement learning to decide when to switch to an alternative representation method depending on the current observation. This allows the robot to update its pose and map estimate based on the representation that models the surrounding of the robot in the best way. The approach has been implemented on a real robot and evaluated in scenarios, in which a robot has to navigate in- and outdoors and therefore switches between a landmark-based representation and a dense grid map. In practical experiments, we demonstrate in [J5] that our approach allows a robot to robustly map environments which cannot be adequately modeled by either of the individual representations.

2.3 Active Feature Selection for Navigation Tasks under Limited Resources

The trend towards light-weight robots and other embedded systems introduces strong resource restrictions to applied algorithms. Such systems and especially small flying vehicles have higher limitations with respect to the computational power and memory capacity than a wheeled robot. It is thus important to develop efficient algorithms that scale with the computational constraints of the underlying hardware.

Efficient algorithms to address the SLAM problem in combination with data association techniques (such as nearest neighbor assignment [57], sampling-based [56], or via the Hungarian method [W2]) are often only one building block for carrying out mapping task on small scale robots. An orthogonal alternative is to actively select observations and/or a subset of features that should be incorporated into the mapping procedure since the memory and computational requirements increase with the number of landmarks that need to be maintained by the robot.

In practice, there are many scenarios in which the number of visible landmarks during a navigation task is significantly larger than the number of landmarks which can be processed efficiently using an embedded device. This leads to the question which landmark should be stored and maintained by the robot to optimally solve the navigation task. A landmark is only useful if it contributes to keep an accurate pose estimate of the robot at the right time and in this way is valuable for the navigation task. We therefore investigate in [C6] a technique for learning a landmark selection policy that optimizes the navigation task carried out by the robot given its computational or memory constraints. The approach combines Kalman filter-based SLAM with reinforcement learning to learn general policies to trigger the incorporation of new features.

We show in [C6] that the policies learned with our approach are not limited to the environment they have been learned in. Rather, they can also be applied successfully in environments with different properties of the underlying landmark distribution.

2.4 Maximum Likelihood Mapping by Pose Graph Optimization

To address the map learning problem from a more general point of view that focusing on one specific type of robot or sensor setting, we developed an approach that separates the interpretation of the sensor data from the estimation problem itself. An elegant framework for separating these two problems is the use of a so-called SLAM *front-end* and *back-end*. As the front-end, one refers to as the interpretation of the sensor data and the back-end as the approach that performs the estimation or optimization. In theory, this separation can be found in most probabilistic estimation techniques for SLAM such as extended Kalman filter, unscented Kalman filter, particle filters, information filters, etc. In practice, however, assumptions about the sensor or the vehicle are often incorporated into the design of the SLAM algorithm in order to build a high performance realization.

One attractive way that allows for realizing an efficient SLAM system while having a strictly

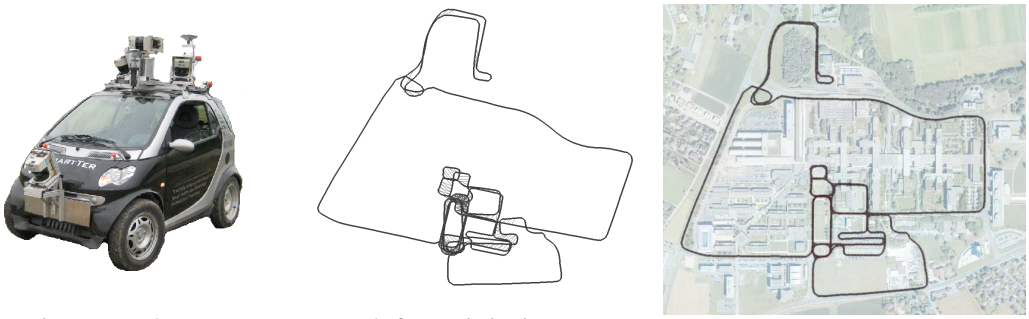
separated front-end and back-end are the so-called pose graph-based or network-based approaches. In this formulation, one models the poses of the robot during mapping by nodes in a graph [21; 26; 29; 35; 38; 52; 59; 60; 83]. Spatial constraints between poses that result from observations and from odometry are encoded in the edges between the nodes. Approaches for solving this formulation of the SLAM problem seek to find the configuration of the nodes that maximizes the observation likelihood encoded in the constraints. A highly related view to this problem is given by the spring-mass model in physics. In this view, the nodes are regarded as masses and the constraints as springs connected to the masses. The minimal energy configuration of the springs and masses describes a solution to the mapping problem.

Popular solutions to compute a pose graph configuration that minimizes the error introduced by the constraints are iterative approaches. They can be used to either correct all poses simultaneously [35; 42; 52; 83] or to locally update parts of the graph [21; 29; 38; 59; 60], [J6], [C22]. Depending on the used technique, different parts of the graph are updated in each iteration. The strategy for defining and performing these local updates has a significant impact on the convergence speed.

We developed a novel approach [J6], [C22; C19; C17], that is inspired by a variant of stochastic gradient descent in the spirit of Olson's algorithm [60] but parameterizes the nodes in the graph according to a tree structure. This structure allows a robot to efficiently update local regions in each iteration of the algorithm. This methods converges significantly faster to accurate pose graph configurations than existing methods and compared to other approaches to 3D mapping, our technique utilizes a more accurate way to distribute the rotational error over a sequence of poses [J6]. Furthermore, the complexity of our approach scales with the size of the environment and not with the length of the trajectory as it is the case for most alternative methods. In addition to that, we developed an incremental solution [C17] designed for online problems by reusing previously computed solutions and by optimizing only updating parts of the map.

We demonstrated that this approach is well suited to deal with different robotic platforms and different sensor modalities. We used the approach using laser range data obtained with traditional SICK scanners [C22] on mobile robots, using SICK scanners on a rotating platform to obtain 3D data mounted on a car [C23] [J6], using a Velodyne 3D laser range scanners on a car [J6], based on vision data with a helicopter and by manually carrying a camera [J7] as well as on a blimp. This illustrates that our method is well suited to be applied to a wide range of mapping problems. We show in [J6] that our approach is highly robust to the initial configuration of the graph and outperforms related state-of-the-art methods such as Olson's algorithm [60] or Multi-level relaxation proposed by Frese *et al.* [29]. An example for an uncorrected and corresponding corrected trajectory of an autonomous car obtained with our approach is shown in Figure 2.1.

All such methods operating on pose graphs have a drawback that they have to store the whole trajectory of the robot. The is not critical for standard mapping scenarios but in case of life-long map learning, the increasing memory and computational resources can become problematic. We therefore developed a technique [J2] to efficiently prune the pose graph. Our technique considers the information gain of observation and is able to discard poses that do not provide new information. As a result, the computational complexity does not increase when the robot moves in known



instrumented car | before optimization | after optimization

Figure 2.1: The left image shows the instrumented car used to record the data around the EPFL campus in Lausanne. The middle and right image depict the constraint network before and after optimization. The corrected network is overlayed with an aerial image.

areas. Thus, a robot operating in a bounded environment can apply our technique for life long map learning.

2.5 Towards Benchmarking of SLAM Approaches

Whereas dozens of different techniques to tackle the SLAM problem have been proposed, there is no gold standard for comparing the results of different SLAM algorithms. In the community of feature-based estimation techniques, researchers often measure the Euclidian or Mahalanobis distance between the estimated landmark location and the true location (if this information is available). As we illustrate in [J3] and [C4], comparing results based on an absolute reference frame can have serious shortcomings. In the area of grid-based estimation techniques, people often use visual inspection to compare maps or overlays with blueprints of buildings.

We address the problem of creating an objective benchmark for comparing SLAM approaches. We propose a framework [J3], [C4] for analyzing the results of SLAM approaches based on a metric for measuring the error of the corrected trajectory. The metric uses only relative relations between poses and does not rely on a global reference frame. Such relative constraints result from externally provided, that means known or accurately measured, distances between locations in the space. This idea is related to graph-based SLAM approaches discussed above, namely to consider the energy that is needed to deform the trajectory estimated by a SLAM approach into to ground truth trajectory. Our method enables us to compare SLAM approaches that use different estimation techniques or different sensor modalities since all computations are made based on the corrected trajectory of the robot.

A certain disadvantage of our method is that it requires manual work that has to be carried out by a human that knows the topology of the environment. The manual work, however, has to be done only once for creating a benchmark dataset and than allows other researchers to evaluate their methods with rather low efforts. To simplify comparisons for future researchers, we provide sets

of relative relations needed to compute our metric for an extensive set of datasets frequently used in the SLAM community. The relations have been obtained by manually matching laser-range observations to avoid potential errors caused by matching algorithms. Our benchmark framework allows the user an easy analysis and objective comparisons between different SLAM approaches.

Chapter 3

Localization

Estimating the pose of a robot relative to a *given* map is a well studied problem in robotics. Most of the effective solutions today rely on probabilistic estimation techniques such as Kalman or information filters in case of roughly Gaussian estimates (e.g., for GPS-based localization) or particle filters to model arbitrary densities.

The approaches presented in this chapter extend the particle filter-based Monte-Carlo localization [17] in the sense that we provide better sensor models for sensor often used in robotics to improve the performance of the pose estimation system. In this chapter, we focus on laser-based localization only. Localization based on cameras will be addressed in Section 4.1.

3.1 A Sensor Model for Localization with Low-Cost Laser Range Finders

One of the key challenges in context of probabilistic localization, however, lies in the design of the so-called observation model $p(z \mid x, m)$ which is a likelihood function that specifies how to compute the likelihood of an observation z given the robot is at pose x in a given map m. For probabilistic approaches the proper design of the likelihood function is essential.

Many successful approaches to localization rely on data provided by range sensors [2; 34; 28; 43; 78]. Laser range sensors provide distance and bearing information to objects in the environment. In practice, one has to deal with erroneous readings (sometimes called "maximum-range" readings) that result from poor-reflecting surfaces or readings obtained in situations in which no obstacle is within the measurement range of the sensor. Especially, low-cost laser range sensors such as the popular Hokuyo URG-04LX suffer from erroneous readings caused by objects with low reflecting surface properties. One popular approach that explicitly models failures corresponding to low or non reflectance is the ray-cast model proposed by Fox *et al.* [28]. This model, however, as well as most others do not take into account that the likelihood of erroneous readings depends on the reflection properties of the corresponding surfaces.

We present in [C8] a novel approach that explicitly considers the reflection properties of surfaces and thus the expectation of valid range measurements. In addition to the expected range

measurement, we compute the probability of reflectance for a beam given the relative pose of the robot to the obstacle taking into account the angle of incidence of the beam. We estimate the reflection properties of surfaces using data collected with a mobile robot equipped with a laser range scanner. As we demonstrate in experiments carried out with a real robot, our technique leads to significantly improved localization results compared to a state-of-the-art observation model. With our models, a robot can faster reduce its pose uncertainty and focus the belief during global localization.

3.2 A Sensor Model for Accurate Laser Range Finders

In our method [C11], we propose a novel probabilistic observation model for accurate proximity sensors such as SICK laser range finders. Our method has two advantages over most previous approaches. First, it explicitly considers the dependencies between the individual beams of a range scan, and second, is accounts for the multi-modal nature of the observation function. It does so while still considering that the observation is obtained from a time-of-flight proximity sensor such as a laser range finder. This is achieved by considering place-dependent measurement models and utilizing a Gaussian mixture model together with a dimensionality reduction technique.

Given the high resolution of typical laser range finders (.25 to 1 degree), the assumption that all beam are independence leads to highly peaked likelihood function. In practice, this problem is dealt with by sub-sampling the measurements [80], by introducing minimal likelihoods for beams, or by other means of regularization of the resulting likelihoods (see Arulampalam *et al.* [3]). In contrast to this, we propose to overcome the peakedness is to consider that the likelihood models are location-dependent and that the location of the robot is modeled by set finite set of pose hypotheses (particles).

The key idea is to determine for each location in the space a set of potential observations, here considering the whole scan not individual beams. This done not only for one pose but also for a local neighborhood. Our approach then learns a high-dimensional Gaussian mixture observation model for each pose in the space. This model is computed in a preprocessing step and can then be used online to compute the likelihood of an observation.

In practical experiments carried out with data obtained with a real robot, we demonstrate that our new model substantially outperforms existing sensor models and allows for highly accurate and peaked pose estimate, being highly robust to perturbations, and allowing for fast global localization.

Chapter 4

Special Sensing Modalities

Cameras have become popular sensors in the robotics community. Compared to proximity sensors such as laser range finders, they have the advantage of being cheap, lightweight, and energy efficient. The drawback of cameras, however, is the fact that due to the projective nature of the image formation process, it is not possible to sense depth information directly. In this chapter, we address robotic vision problems, namely localization based on camera images, range sensing based on single images, a front-end for vision-based SLAM, and landmark location estimation based on geo-referenced images.

4.1 Visual Localization

The ability of a robot to localize itself is required for most robotic applications and this topic was studied intensively in the past. Many approaches exist that use distance information provided by a proximity sensor for localizing a robot in the environment. However, for some types of robots, proximity sensors are not the appropriate choice because they do not agree with their design principle. Humanoid robots, for example, which are constructed to resemble a human, are typically equipped with vision sensors and lack proximity sensors like laser scanners. Therefore, it is natural to equip these robots with the ability of vision-based localization.

In our work [C25], we present an approach to vision-based mobile robot localization that uses a single perspective camera. We apply the well-known Monte-Carlo localization (MCL) technique [17] to estimate the robot's position. MCL uses a set of random samples, also called particles, to represent the belief of the robot about its pose. To locate features in the camera images, we use the Scale Invariant Feature Transform (SIFT) developed by Lowe [51].

Whereas existing systems, that perform metric localization and mapping using SIFT features, apply stereo vision in order to compute the 3D position of the features [5; 24; 65; 66], we rely on a single camera only during localization. Since we want to concentrate on the localization aspect, we facilitate the map acquisition process by using a robot equipped with a camera and a proximity sensor. During mapping, we create a 2D grid model of the environment. In each cell of the grid, we store those features that are supposed to be at that 2D grid position. Since the number of observed

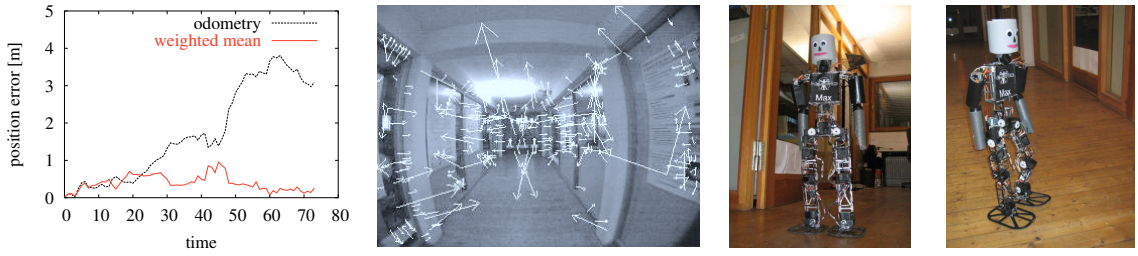


Figure 4.1: Evolution of the error during a typical localization experiment (left), typical observation obtained by the robot (second left) and two pictures of the humanoid robot Max during our experiments.

SIFT features is typically high, we appropriately down-sample the number of features in the final map. During MCL, we then rely on a single perspective camera and do not use any proximity information. Our approach estimates for clusters of particles the set of potentially visible features using ray-casting on the 2D grid. We then compare those features to the features extracted from the current image. In the observation model of the particle filter, we consider the difference between the measured and the expected angle of similar features. By applying the ray-casting technique, we avoid comparing the features extracted out of the current image to the whole database of features (as the above mentioned approaches do), which can lead to serious errors in the data association. As we demonstrate in practical experiments with a mobile robot in an office environment [C25], our technique is able to reliably track the position of the robot. We also present experiments illustrating that the same map of SIFT features can be used for self-localization by different types of robots equipped with a single camera only and without proximity sensors.

4.2 Range Sensing from Omnidirectional Vision

The major role of perception, in humans as well as in robotic systems, is to discover geometric properties of the current scene in order to act in it reasonably and safely. For artificial systems, vision sensors provides a rich source of information about the local environment, since it captures the entire scene – or at least the most relevant part of it – in a single image. Much research has thus concentrated on the question of how to extract geometric scene properties, such as distances to nearby objects, from such images.

This task is complicated by the fact that only a projection of the scene is recorded and, thus, it is not possible to sense depth information directly. From a geometric point of view, one needs at least two images taken from different locations to recover the depth information analytically. An alternative approach that requires just one monocular camera image and that we follow in [J1] and [C16], is to learn from previous experience how visual appearance is related to depth. Such an ability is also highly developed in humans, who are able to utilize monocular cues for depth perception [74].

As a motivating example, consider the left image of Figure 4.2, which shows the image of an

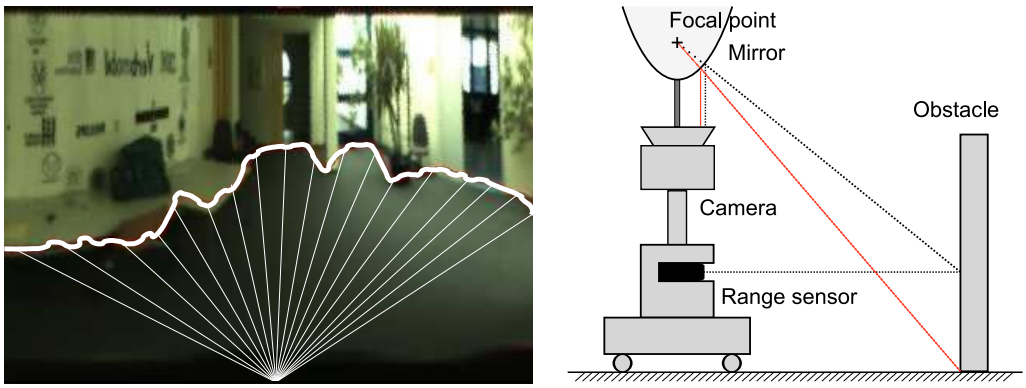


Figure 4.2: Left: Our approach estimates proximity information from a single image after having learned how visual appearance is related to depth. Right: Robot setup for the easy acquisition of training data using a mobile robot equipped with an omnidirectional camera (monocular camera with a parabolic mirror) as well as a laser range finder.

office environment (180° of an omnidirectional image warped to a panoramic view). Overlayed in white, we visualize the most likely area of free space that is predicted by our approach. We aim at learning the function that, given an image, maps measurement directions to their corresponding distances to the closest obstacles. Such a function can be utilized to solve various tasks of mobile robots including local obstacle avoidance, localization, mapping, exploration, or place classification.

The contribution of our work [J1], [C16] is a new approach to range estimation based on omnidirectional images. The task is formulated as a supervised regression problem in which the training set is acquired by combining image date with proximity information provided by a laser range finder. In a first step, we extract visual features from the image data. We extract for every viewing direction α a vector of visual features v from a single image. Features are obtained by supervised and unsupervised dimensionality reduction techniques as well as manually designed features based on algorithms for edge detection. We consider principal component analysis, linear discriminant analysis, as well as local linear embedding for dimensionality reduction.

In the second step, these low-dimensional features serve as the input to a learning engine that seeks to infer range information. The learning approach aims to find the relationship between visual input and the free space around the robot. We phrase the problem as learning the range function $f(\mathrm{v}) = y$ that maps the visual input v to distances y. We learn this function in a supervised manner using a training set of observed features and corresponding laser range measurements.

As a learning framework in our proposed system, we apply Gaussian processes [63] since this technique is able to model non-linear functions and offers a direct way of estimating uncertainties for its predictions. The GP framework allows us to compute a Gaussian estimate for any new query input that is not included in the training set. It provides a range prediction y together with a predictive uncertainty of each feature input v.

To additionally consider the angular dependencies, we apply as a third step the Gaussian beam

processes (GBP) model [62] to learn a heteroscedastic GBP for the set of predicted indexed by their bearing angles and make the final range predictions.

In sum, to obtain the prediction of a full range scan given one omnidirectional image, our approach proceeds as follows:

1. Warp the omnidirectional image into a panoramic view.
2. Extract for every pixel column i a vector of visual features v_i.
3. Use a GP to make independent range predictions about y_i.
4. Learn a heteroscedastic GBP for the set of predicted ranges $\{y_i\}_{i=1}^{n}$ indexed by their bearing angles α_i and make the final range predictions for the same bearing angles.

We evaluated our visual range prediction approach a different dataset recorded at the University of Freiburg and the DFKI in Saarbruecken. Besides measuring the prediction accuracy, we applied a probability mapping approach to the sensor data. Figure 4.3 presents the laser-based maps (left column) and maps using the predicted ranges from the vision data (right column) for the two environments (Freiburg on top and Saarbruecken below). In both cases, it is possible to build a map, which is comparable to maps obtained with infrared proximity sensors [36] or sonars [79].

4.3 Vision-based Map Learning for Flying Vehicles

In Section 2.4, we described our SLAM back-end, that allows to compute pose graphs with low error configurations. Our approach described here, is the corresponding SLAM front-end that can be applied to extract constraints from camera images. We developed a novel approach presented in [J7] and [C20] that allows aerial vehicles to acquire visual maps of large environments using an attitude sensor and low quality cameras pointing downwards. Such a setup can be found on different air vehicles such as blimps or helicopters. Our system deals with cameras that provide comparably low quality images which are also affected by significant motion blur. Our technique uses visual features and estimates the correspondences between features using a variant of the PROSAC algorithm. This allows our approach to extracting spatial constraints between camera poses which can then be used to address the SLAM problem by applying graph methods. We additionally address the problem of efficiently identifying loop closures which is essential for SLAM. Furthermore, our approach can operate in two different configurations: with a stereo as well as with a monocular camera. If a stereo setup is available, our approach is able to learn visual elevation maps of the ground. If, however, only one camera is carried by the vehicle, our system can be applied by making a flat ground assumption providing a visual map without elevation information. The advantages of our approach is that it is easy to implement, provides robust pose and map estimates, and that is suitable for small flying vehicles. Figure 4.4 depicts our blimp and helicopter used to evaluate this work as well as an example camera image obtained with our light-weight camera. In [J7], [C20], we present several experiments with flying vehicles which demonstrate that our method is able to construct maps of large outdoor and indoor environments.

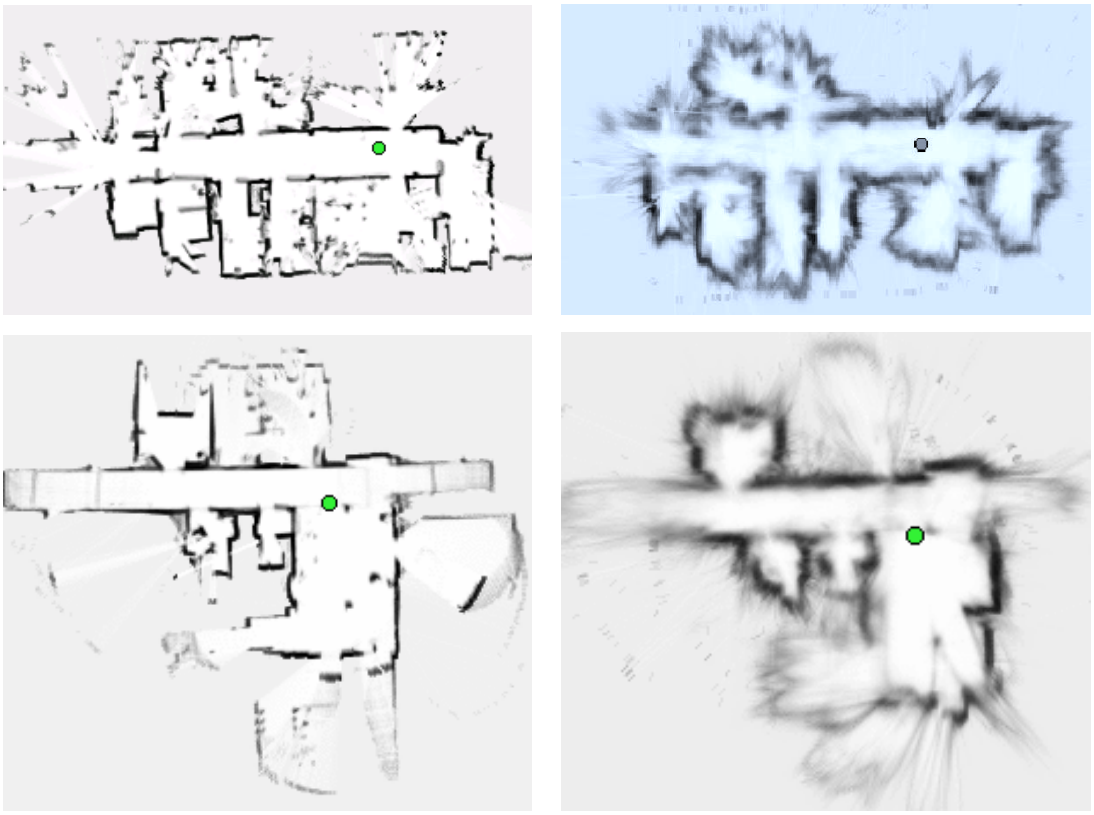


Figure 4.3: Example maps of the Freiburg AIS lab (top row) and DFKI Saarbruecken (bottom row) using real laser data (left) and the range predictions of our approach extracted from omnidirectional camera images (right).

4.4 Estimating Landmark Locations from Geo-Referenced Photographs

In addition to the classical map learning approach in robotics where the robot carries its sensors, we furthermore explore the possibility to use other resources for estimating the location of landmarks. Popular Internet resources such as Flickr or Google Image Search offer a large amount of real world imagery. Many of these images contain geo-references, i.e., the locations where the photographs have been taken in longitude and latitude coordinates as well as manual annotations such as marked image regions and a tag word like “cathedral”. We are interested in how this large amount of freely available data can be used to infer quantitative knowledge about the world. We focus on the problem of localizing a discrete set of distinct landmarks on a larger spatial scale, like a town or city center based on a set of geo-referenced photographs of an environment annotated with labels for distinct landmarks.

A robot that is able to utilize a so far unused source of information offers new ways for building models of places it has not observed directly. It furthermore allows a robot to also refine or annotate

Figure 4.4: Two aerial vehicles used to evaluate our mapping approach as well as an example image recorded from an on-board camera.

exiting models. Consider, for example, a mobile tour guide robot deployed to a city center or to an archaeological site. Given the localized landmarks and the corresponding imagery, the system could offer a large range of location-dependent information without requiring a human expert to collect and formalize this knowledge.

In our work [C10], we consider the problem of estimating the positions of landmarks given a set of geo-referenced photographs. The longitude and latitude information of the locations from which the photos have been taken are assumed to be known approximatively by means of a standard consumer GPS device. By combining this data with labeled regions in the photos referring to objects, such as buildings, our approach is able to localize these buildings and to determine the direction from which the photo has been taken. In contrast to bearing-only SLAM, our approach does not require a continuous image stream from a camera. We furthermore assume to have no knowledge about the orientation of the camera at any point in time. We address this problem by formulating it as an optimization problem. As we showed in [C10], we are able to accurately localize the labeled buildings based on photos taken in an urban environment.

Figure 4.5 depicts the downtown area of Freiburg, where the location of six distinct buildings are marked by circles. Our approach is able to estimate the positions of such landmarks based on a set of photos taken while walking through the city center (estimated locations marked by crosses).

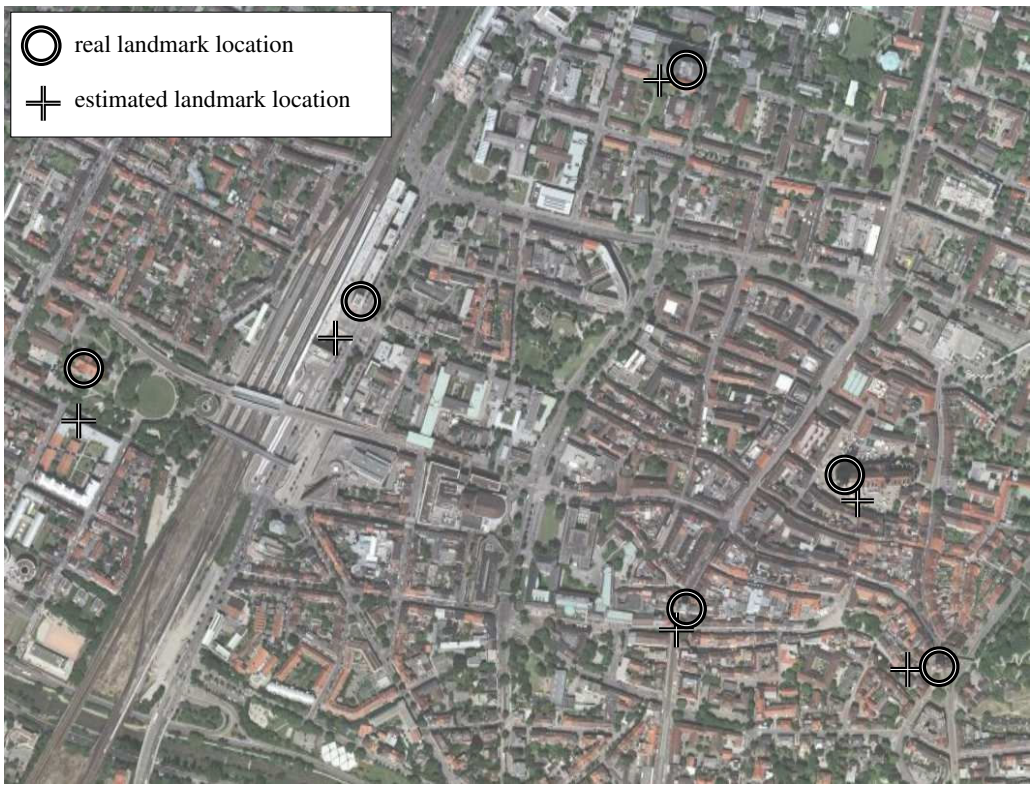


Figure 4.5: The downtown area of Freiburg and the location of six distinct buildings (Herz-Jesu Kirche, train station, Chemie Hochhaus, Muenster, Martinstor, Schwabentor). The circles show the true locations, the circles the one estimated by our approach.

4.5 Gas Distribution Modeling

The problem of modeling gas distributions has important applications in industry, science, and every-day life. Mobile robots equipped with gas sensors are deployed, for example, for pollution monitoring in public areas [22], surveillance of industrial facilities producing harmful gases, or inspection of contaminated areas within rescue missions.

Building gas distribution maps is a challenging task in principle due to the chaotic nature of gas dispersal and because only point measurements of gas concentration are available. The complex interaction of gas with its surroundings is dominated by two physical effects. First, on a comparably large timescale, *diffusion* mixes the gas with the surrounding atmosphere achieving a homogeneous mixture of both in the long run. Second, turbulent air flow fragments the gas emanating from a source into intermittent *patches* of high concentration with steep gradients at their edges [64]. This chaotic system of localized patches of gas makes the modeling problem a hard one. Precise physical simulation of the gas dynamics in the environment requires immense computational resources as well as precise knowledge about the physical conditions, which is not known in most practical scenarios.

When considering gas concentration measurements obtained with a mobile robot, we observed that distributions often consists of a rather smooth "background" signal and several peaks, which indicate high gas concentrations. So, the challenge in gas distribution mapping is to model this background signal while being able to cover also the areas of high concentration and their sharp boundaries. Since it is comparably costly to acquire measurements, one is also interested in reducing the number of samples needed to build a representation. It is important to note that the noise is dominated by the large fluctuations of the instantaneous gas distribution and not by the electronic noise of the gas sensors.

In our approach [J4], [C13], we address the task of modeling a gas distribution by finding a probabilistic model that best explains the observations and that is able to accurately predict new ones. To achieve this, we treat gas distribution mapping as a supervised regression problem. We derive a solution by means of a sparse mixture model of Gaussian processes [82] that is able to handle both physical phenomena highlighted above.

Gaussian processes (GPs) [63] are a non-linear, non-parametric regression technique. It provides a prediction and in addition to that a predictive uncertainty of the estimate. In GPs, one places a prior on the space of functions $p(f)$ using the following definition. A Gaussian process is a collection of random variables, any of which have a joint Gaussian distribution.

More formally, if we assume that $\{(\mathrm{x}_i, f_i)\}_{i=1}^n$ with $f_i = f(\mathrm{x}_i)$ are samples from a Gaussian process and define $\mathbf{f} = (f_1, \ldots, f_n)^\top$, we have

$$\mathrm{f} \sim \mathcal{N}(\mu, \mathrm{K}) \,, \quad \mu \in \mathbb{R}^n, \mathrm{K} \in \mathbb{R}^{n \times n} \,. \tag{4.1}$$

The interesting part of the model is the covariance matrix K which is specified by $[\mathrm{K}]_{ij} := \mathrm{cov}(f_i, f_j) = k(\mathrm{x}_i, \mathrm{x}_j)$ using a covariance function k. It defines the covariance of any two function values $\{f_i, f_j\}$ sampled from the process given their input vectors $\{\mathrm{x}_i, \mathrm{x}_j\}$ as parameters.

Intuitively, the covariance function specifies how similar two function values $f(\mathrm{x}_i)$ and $f(\mathrm{x}_j)$ are depending only on the corresponding inputs. A standard choice for k is a squared exponential function but also other functions are frequently used such as the Matern kernel.

Let $\mathrm{X} = [\mathrm{x}_1; \ldots; \mathrm{x}_n]^\top$ be the $n \times d$ matrix of the inputs and X_* be defined analogously for multiple test data points. In the GP model, any finite set of samples is jointly Gaussian distributed. A prediction of $f(\mathrm{X}_*)$ yields a Gaussian with predictive mean $\bar{f}(\mathrm{X}_*)$ and variance $\mathbb{V}[f(\mathrm{X}_*)]$.

In a *GP mixture model*, one uses a set of GPs which are the individual components. One uses a so-called gating function to define the influence of the individual components over the input space. The hyperparameters of the components as well as the gating function is learned using the Expectation Maximization algorithm [18]. In the EM algorithm, it is required to compute an correspondance probability for each training input of belonging to one component. Since standard GPs require a crisp assinment of the inputs, modifications to the GP appraoch are required. A soft assignment can be achived by introducing individual observation noise terms for the training data which incorporate the assignement probabilities. As a result, one obtains a model that is well suited to capture the properties of gas distributions by allowing a rather smooth "background" process in combination with a process for modeling localized gas pachtes and therehot-spot like characters.

The three main steps of the GP mixture model approach are:

1. **Initializing the Mixture Components:** In a first step, the initial component is computed based in samples data points. To improve the estimate of gas concentration in areas that are poorly modeled by this initial model, we learn an "error GP model" that captures the absolute differences between a set of target values and the predictions so far. By initializing future mixture components based on input data for locations sampled from the error GP, the new componets will improve the estimate in so far badly approximated areas.

2. **Iterative Learning via Expectation-Maximization:** The Expectation Maximization (EM) algorithm is used to learn the mixture correspondance variables for the inputs and to optimize the hyperparameters of the covariance function of the GPs.

   In the E-step, we estimate the probability $P(z(\mathrm{x}_j) = i)$ that data point j corresponds to model i. This is done by computing the marginal likelihood of each data point for all models individually. Thus, the new $P(z(\mathrm{x}_j) = i)$ is computed given the previous estimate as

$$P(z(\mathrm{x}_j) = i) \;\leftarrow\; \frac{P(z(\mathrm{x}_j) = i) \cdot \mathcal{N}_i(y_j; \mathrm{x}_j)}{\sum_{k=1}^{m} P(z(\mathrm{x}_j) = k) \cdot \mathcal{N}_k(y_j; \mathrm{x}_j)} \,. \tag{4.2}$$

   In the M-step, we update the components of our mixture model. This is achieved by integrating the probability that a data point belongs to a model component into the individual GP learning steps. This is achieved by modifying the prediction mean estimate to

$$\bar{f}_i(\mathrm{X}_*) \;=\; k(\mathrm{X}_*, \mathrm{X}) \left[k(\mathrm{X}, \mathrm{X}) + \Psi^i\right]^{-1} \mathrm{y} \,, \tag{4.3}$$

where Ψ^i is a matrix with

$$[\Psi^i]_{jj} = \frac{\sigma_n^2}{P(z(\mathrm{x}_j) = i)} \tag{4.4}$$

and zeros in the off-diagonal elements (instead of $\sigma_n^2 \mathrm{I}$ as in the standard GP gas, with obervation noise σ_N). The matrix Ψ^i allows us to consider the probabilities that the individual inputs belong to the corresponding components.

3. **Learning the Gating Function:** The gating function defines for an arbitrary data point the likelihood of being assigned to the individual mixture components. The EM algorithm learns the assignment probabilities for all training inputs x_j, maximizing the overall data likelihood. To generalize these assignments to the whole input space, we place another GP prior on the gating variables. Concretely, we learn a gating GP for each component i that uses the x_j as inputs and the $z(\mathrm{x}_j)$ obtained from the EM algorithm as targets. Let $\bar{f}_i^z(\mathrm{x})$ be the prediction of z for the i-th GP. Given this set of m GPs, we can compute the correspondence probability for a new test point x_* as

$$P(z(\mathrm{x}_*) = i) = \frac{\exp(\bar{f}_i^z(\mathrm{x}_*))}{\sum_{j=1}^m \exp(\bar{f}_j^z(\mathrm{x}_*))} . \tag{4.5}$$

Carrying out these three steps sequentially, allows us to efficiently learn a GP mixture model that turns out to be well suited to model distributions of gas concentrations. In experiments with real robots equipped with an ethanol sensor and cups of ethanol distributed in different environments, we were able to show that our approach is well-suited to model gas distributions. An example distribution is depicted in Figure 4.6.

We compared our method with three different approaches: grid-based averaging with linear interpolation for areas where no observations have been obtained. Second, the kernel extrapolation technique of Lilienthal and Duckett [50] and finally to a standard GP approach. For comparisons, we used the MSE (see Figure 4.7) and the average negative log likelihood using a a test set of gas concentration measurements in a cross-validation fashion. We can show that our approach clearly outperforms the spatial averaging using the grid and standard GPs. Futhermore, our approach also outperformed the kernel extrapolation technique of Lilienthal and Duckett significantly. In constrast to kernel extrapolation and spatial averaging, our approach is also able to provide a predictive uncertainty of the estimate and in this way provides an uncertainty of the prediction.

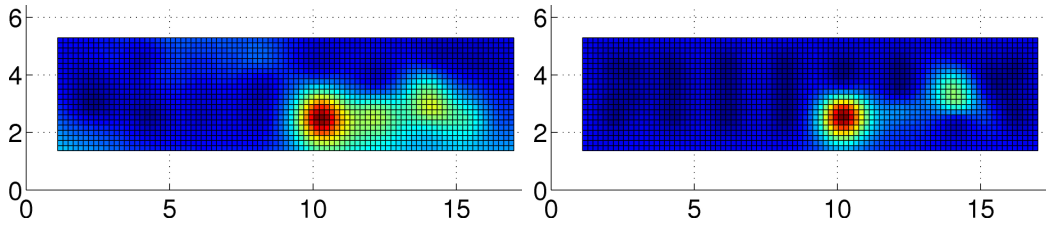


Figure 4.6: Example illustration of a gas distribution learned from concentration data recorded in a corridor environment using our GP mixture model. Left: predictive mean, right: predictive uncertainty. The gas source was placed at the location (10, 3).

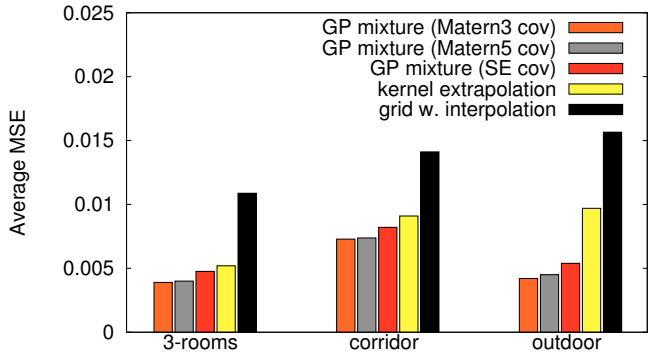


Figure 4.7: Experimental comparison of our GP mixture model with different covariance functions to other state-of-the art alternatives in three real-world setting. The bars show the mean squared error of predicted gas concentration w.r.t. the measured one on a test set, averaged over 10 runs.

Chapter 5

Towards Understanding Environments

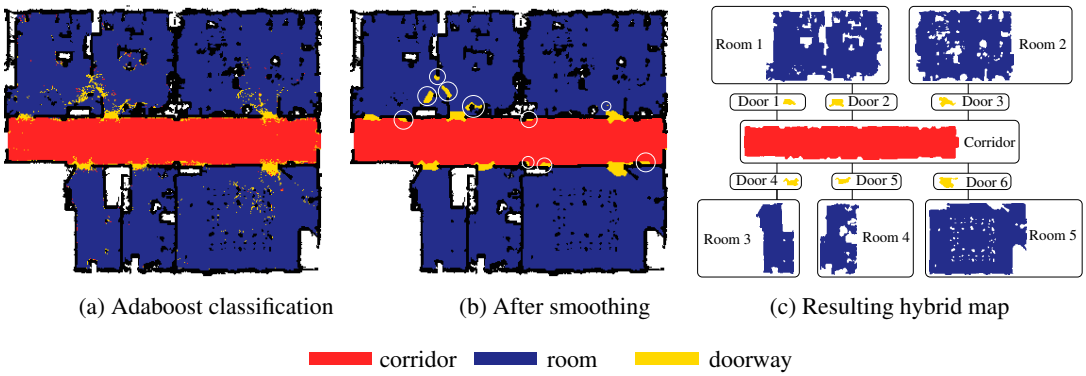
In real world settings, robots need more information about their surroundings than a metric description of the obstacles in the world. In this chapter, we introduce our achievements that allow robots to get a better understanding of the scene. First, we show how to learn hybrid models of the environment combining metric, topological as well as semantic information. Second, we present an approach that allows a robot to group similar objects observed in a scene and compute an assignment of objects to classes. Finally, we present an approach to learning kinematic models of everyday objects such as door or drawers based on observation obtained with a mobile robot.

5.1 Hybrid Maps

In the mobile robot map learning community, one typically distinguishes between the type of model the mapping approach learns: metric or topological maps. Metric maps like such as occupancy, feature, or geometric maps model the objects observed by the sensor. These maps are often used to explicitly represent obstacles and driveable areas. Typically, the resulting model strongly depends on the sensors used by the robot to perceive its environment. Metric maps often bear resemblance to floor plans used in architecture.

For different robotic tasks, however, the robot can improve its capabilities or performance if sematic or topological information is available. In contrast to metric maps, topological maps model the structure of the environment using a graph. The different places in the environment are represented by nodes in that graph. Topological maps became also popular in the robotics community because they are believed to be cognitively more adequate. Compared to metric maps, they can be stored in a compact manner and can facilitate the communication with the users.

While most other mapping approaches address either metric or topological map learning, we focus on constructing a hybrid map that covers metric aspects as well as the topology of the environment [J8]. This enables a mobile robot to use the best suited model for the task it performs. Our approach consists of two steps. In the first one, we apply a highly efficient particle filter to solve the simultaneous localization and mapping problem, see previous chapter and [C18], [J9]. This step is based on grid maps and eliminates the pose uncertainty of the robot by selecting the



(a) Adaboost classification (b) After smoothing (c) Resulting hybrid map

Figure 5.1: This figure shows a map of the building 79 at the University of Freiburg. Image (a) depicts the result of applying the sequential AdaBoost with a classification rate of 97.3%, and (b) depicts the result of applying relaxation and the detection of incorrect labeled regions (marked with circles), and (c) the final hybrid map showing the topology and the corresponding regions as metric models with semantic labels.

most likely configuration based on a joint posterior about the trajectory of the robot and the map of the environment.

In the second step, we use the grid map resulting from the first step in order to infer the topology. The key ideas is to estimate semantic information about local areas as proposed in a previous work and during the PhD thesis [70; 53]. The approach computes rather simple geometric features from laser range scans and estimates a semantic label such as corridor, room, or doorway for each location in the space using the AdaBoost algorithm. This approach, however, treats every location in the space individually. Due to the structure of environments made by humans, the semantic class does not change randomly between nearby poses. Therefore, it makes sense to consider a smoothing approach between places located close together. Therefore, we apply probabilistic relaxation labeling to smooth the semantic labels. This reduces the risk of false classifications and improves the estimate of the semantic class of poses. Based on the smoothed class labels and the grid map, one can extract distinct places in the environment that refer, for example, to individual rooms. Building a graph structure that connects the individual places based on the connectivity, that can easily be extracted from the grid map, allows a mobile robot to learn a consistent topological representation of the space and at the same time an accurate metric model. One advantage of this approach is that the nodes of the resulting graph correspond to the individual semantic regions. This links the metric and the topological representations. As a result, a robot is finally able to maintain a hybrid representation of the space allowing it to select the model that is best suited to solve a given task.

Figure 5.1 illustrates the extraction of semantic information and the resulting hybrid model for an office space at building 79 at the University of Freiburg.

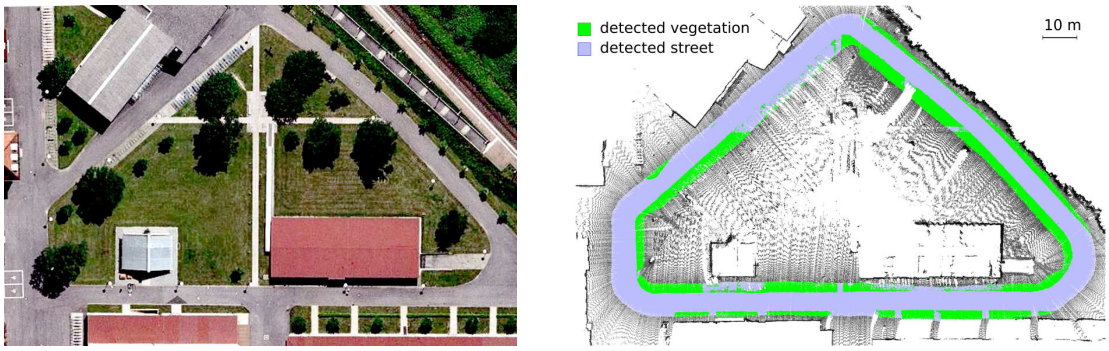


Figure 5.2: Left: Aerial image of the computer science campus in Freiburg, courtesy of Google Maps, Copyright 2008, DigitalGlobe. Right: Corresponding mapping result obtained with a mobile robot driving on the paved road around the building. The image shows a 2D projection of the 3D map including labeled areas.

5.2 Vegetation Estimation

The techniques described above is designed for indoor environments. In most outdoor navigation scenarios including autonomous cars, autonomous wheelchairs, surveillance robots, or transportation vehicles, also the classification of the terrain plays an important role as most of the robots have been designed for navigation on streets or paved paths rather than on natural surfaces covered by grass or vegetation. The navigation outside of paved paths might be uncomfortable for passengers or even might introduce the risk for the robot of getting stuck. Furthermore, driving on grass will in general increase wheel slippage and in this way lead to potential errors in the odometry. In addition to that, a technique for terrain classification can directly combined with the above described approach to learn hybrid maps for outdoor scenes. Accordingly, the robust detection of vegetated areas is an important requirement for robots in any of the above-mentioned situations.

We therefore developed a novel approach [C3] for vegetation detection from laser measurements by exploiting the laser remission values. In our algorithm, the laser remission is modeled as a function of distance, incidence angle, and material. We classify surface terrain based on 3D scans of the surrounding of the robot based on support vector machines. The model is learned in a self-supervised way using vibration-based terrain classification. Practical experiments demonstrate that our approach yields a classification accuracy of over 99% in all tested settings. This evaluation was carried out based on a series of real-world experiments at the campus of the University of Freiburg. An example map obtained with a mobile robot is shown in Figure 5.2. We also demonstrate in [C3] how the learned classifier can be used to improve autonomous navigation.

5.3 Classifying Objects in Scenes

In home environments, which are envisioned as one of the key application areas for service robots, a robot does not only need a model of the space but also has to deal with a variety of different objects. The ability to distinguish objects based on observations and to relate them to known

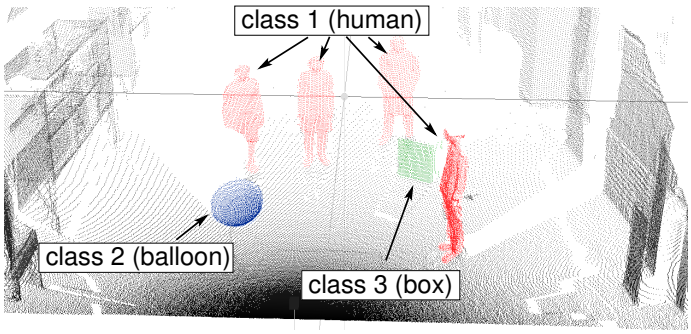


Figure 5.3: Example of a scene observed with a laser range scanner mounted on a pan-tilt unit. Points with the same color resemble objects belonging to the same class.

classes of objects is important for autonomous service robots. Analyzing a scene and identifying objects and their classes based on sensor data is a hard problem due to the varying appearances of the objects belonging to specific classes. In our work [C2], we consider a robot that can observe a scene with a 3D laser range scanner. The goal is to learn a model for object classes unsupervised, to perform a consistent classification of the observed objects, and to obtain a correct classification of unseen objects belonging to one of the known object classes. As an illustrating example, consider Figure 5.3 which depicts a typical point cloud of a scene considered. It contains four people, a box, and a balloon-like object. The individual colors of the 3D data points illustrate the corresponding object classes that we want our algorithm to infer.

An important distinction between different approaches to object detection and recognition is the way the objects or classes are modeled. Models can be engineered manually, learned from a set of labeled training data (supervised learning) or learned from unlabeled data (unsupervised learning). While the former two categories have the advantage that detailed prior knowledge about the objects can be included easily, the effort for manually building the model or labeling a significant amount of training data becomes infeasible with increasing model complexity and larger sets of objects to identify. Furthermore, in applications where the objects to distinguish are not known beforehand, a robot needs to build its own model, which can then be used to classify the data.

The contribution of our work is a novel approach for discovering object classes from range data in an unsupervised fashion and for classifying observed objects in new scans according to these classes. Thereby, the robot has no a-priori knowledge about the objects it observes. Our approach operates on a 3D point cloud recorded with a laser range scanner. We apply latent Dirichlet allocation (LDA) [7], a method that has recently been introduced to seek for topics in text documents [32]. The approach models a distribution over feature distributions that characterize the classes of objects. Compared to most popular unsupervised clustering methods such as k-means or hierarchical clustering, no explicit distance metric is required. To describe the characteristics of surfaces belonging to objects, we utilize a variant of spin-images as local features that serve as input to the LDA. The learned feature distributions can subsequently be used as models for the classification of unseen data. An important property of our approach is that it is unsupervised and

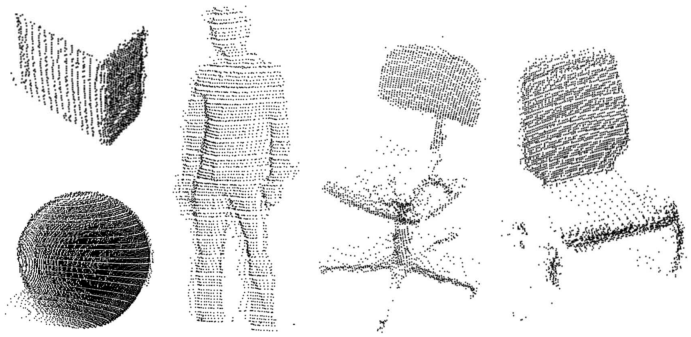


Figure 5.4: Example point cloud segments of a box, balloon, human, swivel chair, and regular chair.

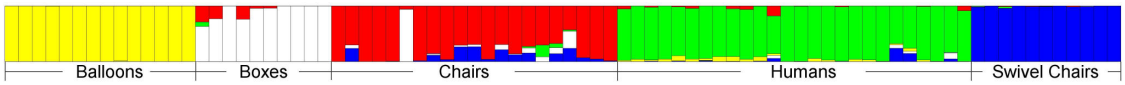


Figure 5.5: Resulting topic mixtures for 82 segments containing scanned objects computed via LDA (the labels were not provided to the system).

does not need labeled training data to learn the partitioning.

We show in practical experiments on real 3D data that a mobile robot following our approach is able to identify similar objects in different scenes while at the same time labeling dissimilar objects differently. Examples for 3D range scans of objects under consideration are depicted in Figure 5.4, consisting of boxes, balloons, humans, and different types of chairs. Even for datasets containing complex objects with varying appearance such as humans, we achieve a robust performance with over 90% correctly grouped objects. An example visually illustrating the performance of our approach is depicted in Figure 5.5. It shows the topics assigned by our approach to a set of 82 scan segments. The labels in this diagram show the true object class. Each color in the diagram denotes one topic and the ratios of colors denote for each object segment the class assignment weight. As the diagram shows, except of one chair, all objects are grouped correctly when using the maximum likelihood assignment.

We furthermore demonstrate that our approach clearly outperforms unsupervised clustering approaches such as hierarchical clustering. LDA does not only achieve higher classification accuracy throughout the entire parameter range, it is also less sensitive to the choice of parameters.

5.4 Identifying Objects by Tactile Sensing

We also considered tactile sensing as a potential source of information in order to detect objects in the scene. In [C5], we present a novel approach for identifying objects using touch sensors installed in the finger tips of a robot manipulator. Our approach operates on low-resolution intensity images that are obtained when the robot grasps an object. We apply a bag-of-words approach

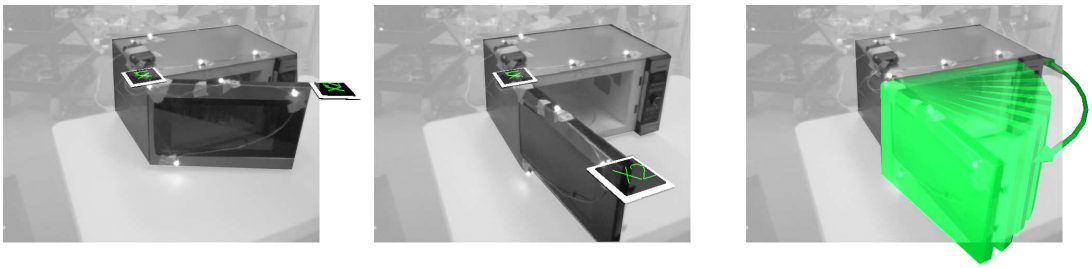


Figure 5.6: Left and middle: examples for observations of a moving door of a microwave oven. Right: visualization of the 1-dimensional kinematic model of the door learned by our approach.

that is frequently used in the vision community [1; 14; 15; 86] for object or scene identification. By means of unsupervised clustering on training data, our approach learns a vocabulary which is used to generate a histogram codebook. The histogram codebook models distributions over the vocabulary and is the core identification mechanism. As the objects are larger than the sensor, the robot typically needs multiple grasp actions at different positions to uniquely identify an object. To reduce the number of required grasp actions, we apply a decision-theoretic framework that minimizes the entropy of the probabilistic belief about the type of the object. To efficiently calculate the expected information gain of the next grasp action, we approximate it based on the distribution of potential observations extracted from the training data. In our experiments carried out with various industrial and household objects, we demonstrate that our approach is able to discriminate between a large set of objects. We furthermore show that using our approach a robot is able to distinguish visually similar objects that have with different elasticity properties by using only the information from the touch sensor.

5.5 Learning Kinematic Models of Objects

Robots operating in home environments must be able to also interact with articulated objects such as doors or drawers. Ideally, robots are able to autonomously infer articulation models by observation. This means that a robot is able to infer and model potential movements of objects which is a prerequisite for actions such an autonomously opening doors. We therefore developed an approach [C1] to learn kinematic models of such object.

The considered problem can be formulated as follows: Given a sequence of locations from observed objects parts, learn a compact kinematic model describing the *whole* articulated object. This kinematic model has to define (i) which parts are connected, (ii) the dimensionality of the latent (not observed) actuation space of the object, and (iii) a kinematic function between different body parts in a generative way allowing a robot to reason also about unseen configurations.

We assume that objects consist of different rigid parts and our approach has to infer the connectivity of that rigid parts. Additionally, we learn articulation models for the corresponding links. Our approach uses a mixture of parameterized models, for example, to describe typical joints

such as a rotational joint or a prismatic joint, as well as parameter-free kinematic models. The parameter-free models apply a mixture of non-linear dimensionality reductions to find the manifold that encodes the hidden action variable and Gaussian process regression. Our approach then selects models of the individual links that provide the best explanation of the given observations. Our approach has been implemented and thoroughly evaluated. We demonstrate in the experimental section that our technique allows for learning accurate models of different articulated objects from real data. An example of the door of a microwave oven as well as the infered model of the door is depicted in Figure 5.6. This is an important step towards autonomous robots understanding and actively handling objects in their environment.

Chapter 6

Action Selection for Navigation

This chapter presents solutions to different robotic problems in which the central aspect is action selection for a single robot or a team of robots. In detail, we focus on single and multi-robot exploration, navigation amongst deformable objects, computer-controlled cars, and the imitation of a human demonstrator to learn manipulative tasks by demonstration.

6.1 Exploration

There are several applications like planetary exploration, reconnaissance, rescue, mowing, or cleaning in which the complete coverage of a terrain belongs to the integral parts of a robotic mission. To allow a mobile robot to explore an unknown environment on its own, the robot has to generate exploration trajectories that allow for perceiving the environment with its sensors. In the past, different variants of the robot exploration problem have been studied. One typically distinguishes between single-robot and multi-robot exploration problems. In the context of the single-robot exploration problem, key questions are the generation of trajectories so that the robot can efficiently cover the space with its sensors and the ability to deal with the uncertainty in the sensor data and the pose information. In case multiple robots are deployed, the focus often lies on the coordination of the robots' actions to avoid redundant work and physical interference between robots.

Both problems received considerable attention in the past in the robotics community. Most exploration approaches, however, assume that the robot lives in a plane and do not consider the full 3D space. To overcome this limitation, we developed a novel exploration system [W3] that allows a robot to explore the 3D space learning 3D multi-level surface maps from laser range data. The approach can be seen as an extension of our previous work [70; 71] in the sense that it estimates the expected information gain of future observations.

In case multiple robots are used to explore an environment, the coordination of the individual agents is crucial for high performance. Based on our previous work [10], we presented a framework [C12] that utilized typical structures in indoor environments. By extracting regions which typically correspond to rooms and assigning robots to explore such sub-regions separately, the

coordination of the agents can be significantly improved.

6.1.1 Information Gain-based Exploration in 3D

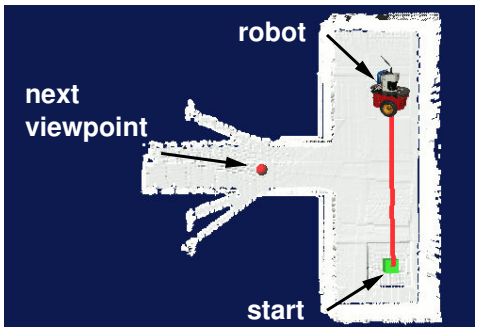
Our 3D information gain-based exploration approach [W3] is an exploration technique that extents known techniques from 2D [70; 71] into the three-dimensional space. It allows us to address problems which are not encountered in traditional 2D representations such as negative obstacles, roughness, and slopes of non-flat environments. Our approach constructs a full three-dimensional model using so-called multi-level surface (MLS) maps. MLS maps [83], [61], [C23], [W4], use a two-dimensional grid structure that stores multiple elevation values. In particular, they store in each cell of a discrete grid the height of the surface in the corresponding area. In contrast to elevation maps, MLS maps allow us to store multiple surfaces in each cell. Each surface is represented by a Gaussian with the mean elevation and its uncertainty. This representation enables a mobile robot to model environments with structures like bridges, underpasses, buildings, or mines.

Our approach is formulated in a probabilistic way and the robot selects actions that reduce its uncertainty in the world model when selecting new target locations for sweeping the environment with its sensor. In order to evaluate the information gain of a candidate viewpoint, we perform a ray-cast operation on the map to determine the 3D patches in the map that are likely to be hit by a laser measurement. This allows us to efficiently reason about the potential measurements. The approach then estimates the expected information gain which is the reduction of uncertainty in its model. It finally selects the candidate viewpoint that minimizes the model uncertainty as well as the travel cost to reach the corresponding view point. Figure 6.1 illustrates parts of the decision process and the trajectories resulting from two real world experiments with a mobile robot. It should be noted that our approach is also able to deal with negative obstacles like, for example, abysms, which is a problem of robots exploring a three-dimensional world.

6.1.2 Coordinated Multi-Robot Exploration by Space Segmentation

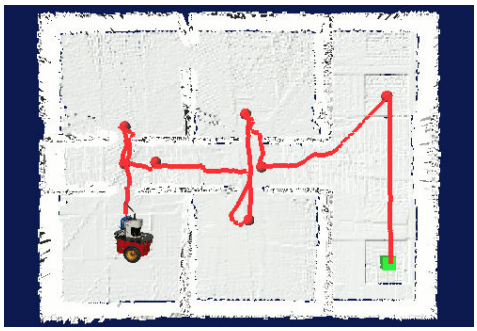
In our work [C12], we consider the problem of efficient exploration with teams of mobile robots that seek to minimize the overall time required to complete the mission. To achieve this, the robots have to select appropriate target locations, to minimizing the traveled distance, to avoid interference between agents, and to avoid redundant work. The coordination task can roughly be separated into two subsequent tasks. First, the identification of potential exploration targets and second, the assignment of the individual robots to the target locations.

A popular method for generating potential exploration targets has been proposed by Yamauchi *et al.* [85]. In this approach, robots are sent to the borders between the explored and the unexplored space called frontiers. A large set of approaches operate based on frontiers or very similar concepts exists [10; 31; 67; 72; 73; 88]. Such coordination strategies consider individual locations rather than segments of the environment. Segmentation approaches which have recently received an increased amount of attention [9; 30; 87], [J8]. In our work, we introduce a new online coordination strategy for multi-robot exploration. It uses a segmentation of the already explored area to assign robots to segments instead of directly assigning them to

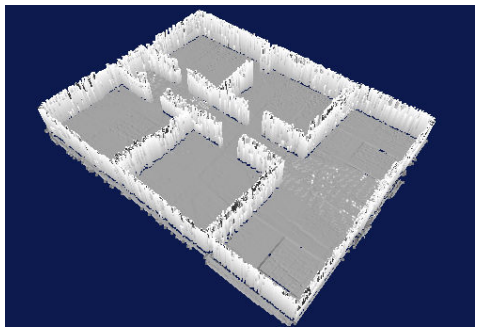


(a) Robot reached the first viewpoint.

(b) Robot reached the fourth viewpoint.



(c) Robot reached the final viewpoint.



(d) Perspective view of the final map.

Figure 6.1: Exploration in a simulated indoor environment. One can see four rooms, a corridor, and the foyer where the robot started the exploration.

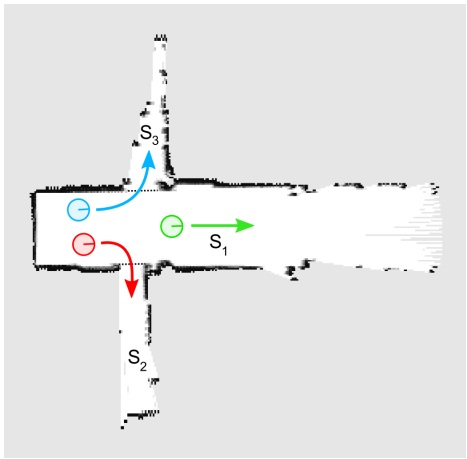


Figure 6.2: Illustration of coordination by space segmentation: Target assignments for the robots are obtained by exploiting different segments extracted from the partial map.

frontier targets. In our current implementation, the segmentation consists of typically indoor structures such as individual rooms, corridors, or similar. A small illustration of a segmentation into three regions during exploration is given in Figure 6.2. Based on this segmentation, the robots are distributed over the environment. To achieve this, the Hungarian method [44; 45] is applied to find the optimal pairing of robots and targets given a cost function which encodes the reachability of segments. This approach distributes a team of robots more effectively than existing methods. This leads to a reduction of redundant work as well as the avoidance of interference between robots and as a result, the exploration time is significantly reduced.

6.2 Navigation amongst Deformable Objects

Path planning is an elementary problem in robotics and the ability to plan collision-free paths is a precondition for numerous applications of autonomous robots. The path planning problem has traditionally received considerable attention in the past and has been well-studied. The majority of approaches, however, has focused on the problem of planning paths in static environments and with rigid obstacles [47; 12; 48]. In the real world, not all obstacles are rigid and considering this knowledge can enable a robot to accomplish navigation tasks that otherwise cannot be carried out. For example, in our everyday life we deal with many deformable objects such as plants, curtains, or cloth and we typically utilize the information about the deformability of objects when carrying out a movement.

As a motivating example, consider a robot that needs to pass through a curtain to move from its current position to the goal location since no other path exists in the environment. In this particular situation, traditional approaches that do not take the deformability of objects into account, will fail since no collision-free path exists. In contrast to this, approaches that know about deformability of objects are able to determine the deformation cost introduced by passing the curtain and to utilize this information during path planning.

One potential method of taking deformations of objects into account is by generating trajectories using a method such as probabilistic roadmaps [39] and considering deformable objects as free space. When answering path queries, the planner has to simulate the deformation of the non-rigid objects resulting from the interaction with the robot. However, performing an appropriate physical simulation typically requires significant computational efforts which makes such an approach unsuitable for online trajectory planning. Therefore, we propose an approach [C15; C7] to learn an approximative deformation cost function in a preprocessing step. The advantage of our method is that this function can be evaluated efficiently during planning. In this way, our approach reduces the time to solve a path query from several minutes to a few hundred milliseconds.

The contribution of our work is an approach [C15] to mobile robot path planning that explicitly considers deformable objects in the environment. It employs the probabilistic roadmap method and learns a deformation cost function using an appropriate physical simulation engine [68; 75; 76; 69] that is based on Finite Element theory. Our approach trades off the travel cost with the deformation cost when answering path queries and can be executed online. Two snapshots of an experiment are depicted in Figure 6.3. For each situation, the simulated deformation and the real ones are shown.

In order to apply such an approach in the real world, the robot furthermore needs to be able to appropriately interpret its sensory input during the interaction with the deformable objects. For example, during the interaction, the robot necessarily gets close to the deformable object so that its field of view might get obstructed. For safe navigation, however, the robot still needs to be able to identify the measurements that do not correspond to the deformable object and come from other, unexpected, and possibly rigid objects.

We therefore developed a probabilistic approach [C7] that allows a mobile robot to distinguish measurements caused by deformable objects it is interacting with from ordinary measurements.

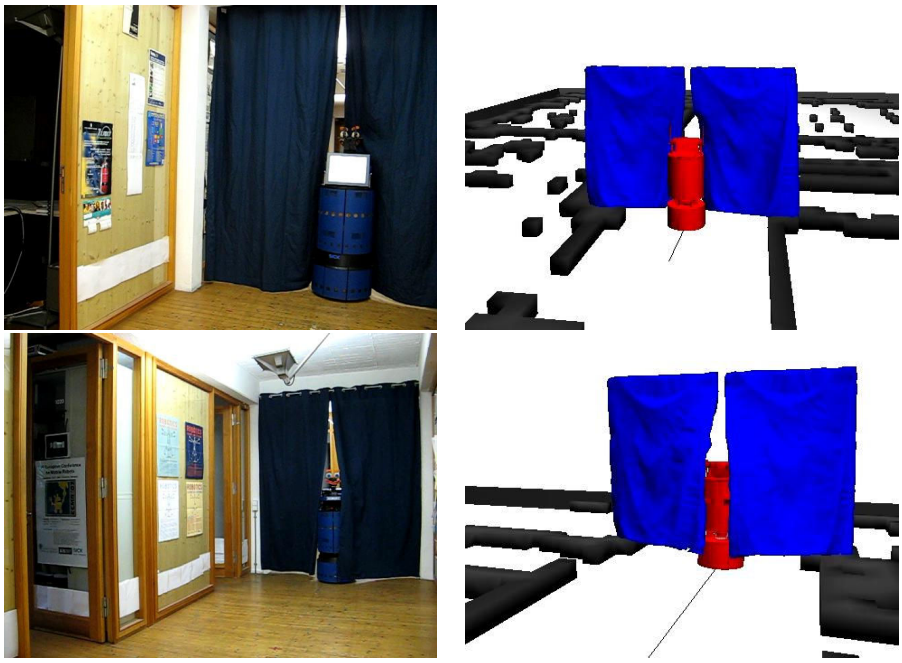


Figure 6.3: Real and simulated deformations for two situations.

This allows the robot to utilize standard reactive collision avoidance techniques like potential fields [41] or dynamic window techniques [8; 27; 54] by filtering out measurements that are caused by the deformable objects the robot is interacting with. Additionally, the ability to reliably identify measurements not perceiving parts of the deformable object enables the robot to correctly interpret them also for the sake of collision avoidance. Our approach has been implemented on a real robot and evaluated in a collision avoidance task carried out while the robot interacts with a curtain. The results demonstrate that our approach allows the robot to safely avoid obstacles while it is interacting with a deformable object.

Figure 6.4: Our computer-controlled Smart car and the sensor setup mounted on the roof of the car. The right image shows our car during the first European Landrover Trials (ELROB).

6.3 Computer-controlled Cars

One interesting application area for techniques developed in the robotics community are computer-controlled cars that can perceive the environment and drive autonomously. Especially, since the DARPA Gran Challenge [16], the usage of cars instead of classical mobile robots became popular [13; 81; 84].

In a joint effort of the EPFL in Lausanne, the ETH Zurich, and the University of Freiburg, we developed a computer-controlled car based on a Smart car. The Smart car has been modified and equipped with five laser range finders, an inertial measurement unit, differential GPS, cameras, and four computers. To goal of this project was to build an autonomous platform that is able to build three-dimensional models of the environment, localize within these models, and can to plan and execute collision free trajectories to a given target location. Photos of the car and its sensors are shown in Figure 6.4.

6.3.1 Car Modifications

In order to turn the Smart car (Smart fortwo coupé passion 2005) into a robotic car, a series of modifications have been carried out (more details can be found [W4]):

- An automotive engine control unit (ECU) has been installed between the computers and the vehicles own CAN bus to allow for a clear interfacing.
- Customized access to the power steering system of the car.
- A separated emergency break system that mechanically activates the break pedal.
- An own electronic board to set gas commands.
- A 24V power generator has been installed to the engine output axis in order to power all the electronic devices.

To perceive the environment, a set of sensors has been installed in the car, namely:

- Three laser scanner sensors for obstacle avoidance and navigation (SICK LMS291-S05).
- Two rotating laser range scanners to obtain a 3D scanning device.
- Perspective and omnidirectional cameras.

Figure 6.5: Overlay of the estimated trajectory and the ortho-photo of the EPFL campus. The zones where the GPS was not available are highlighted. The total traveled distance is around $2300\,m$. The labels (a), (b), and (c) identify areas where GPS errors occurred but which did not cause pose estimation failures due to the multi-sensor data fusion.

- A differential GPS system (Omnistar Furgo 8300HP).
- an optical gyroscope (KVH DSP3000).
- An inertial measurement unit (Crossbow NAV420).

In addition to that, four industrial PCs were installed in the car to carry out the necessary computations. The key functionality the car provided are

- Localization
- Mapping
- Path planning

6.3.2 GPS-based Localization

The localization system of the car mainly relies on GPS information. GPS alone, however, is not sufficient to provide smooth pose estimates and furthermore cannot deal with GPS loss. We therefore fused the data obtained by the inertial measurement unit, the differential GPS, the optical gyro, and the wheel encoders.

Our localization system applies the inverse form of the Kalman filter, i.e., the information filter. This filter has the property of summing information contributions from different sources in the update stage. This characteristic is advantageous when many sensors are involved which is the case in our application. As shown in [C23], the system allows for accurate pose estimation even in case of temporary GPS loss. Figure 6.5 depicts a trajectory estimate of the vehicle during an experiment over an ortho-photo of the EPFL campus.

6.3.3 Mapping

To achieve 3D mapping capabilities of the car, we developed a system that computes local three-dimensional models of the surroundings of the vehicle. This is done by building a local multi-level surface map [83],[C23],[W4] based on the laser range data. To apply the graph-based SLAM approach presented in Chapter 2, a pose graph is created in which each node is linked to a local multi-level surface map. To obtain constraints between nodes, we use a registration techniques based on the iterative closest point algorithm. Instead of registering single points, we developed a matching approach based on MLS maps [83], [61], [C23], [W4]. To efficiently register whole maps, we incorporate a local traversability measure into the registration procedure which speeds up the matching significantly.

An example for a map learned from the dataset used above to briefly illustrate the localization capabilities of our system is shown in Figure 6.6. In this image, the yellow cells indicate traversable area for the car, red to non-traversable ones and blue to vertical (also non-traversable) objects. A second example for a 3D map learned at a military test side is depicted in Figure 6.7.

Such maps can then be used to plan trajectories and specify a goal location for the robot. Often, local maps modeling the obstacles in the surroundings of the vehicle are sufficient for navigation and global maps are not needed in most cases. There are, however, situations in which a map is inherently needed to successfully perform the navigations task. This is, for example, the case for autonomous parking in a large garage as shown by Kümmerle *et al.* [46].

6.3.4 Motion Planning

To actually plan a trajectory, the model of the environment can be used for global planning applying, for example, the popular A* algorithm or efficient variants. In our work [W5], we used Field D* [25] since it allows for efficient replanning. This algorithm provides low-cost 2D paths through grid-based representations of an environment and is able to repair these paths when accounting for new information as the vehicle observes obstacles during its traverse. These 2D paths, however, do not take into account the heading constraints of the vehicle. Instead, the approach approximates the least-cost path to the goal for a vehicle that can turn in place. Since Field D* does not encode the mobility constraints of the vehicle, it cannot be used alone for trajectory planning for the vehicle. Consequently, we combine it with a local planner to provide feasible paths. One way to do this is to use the Field D* path as the input to the local planner, which will then track this path to the goal. As the vehicle navigates through the environment, the global Field D* path is updated based on new information received through the onboard sensors, and the trajectories generated by the local planner are subsequently updated to reflect the new global path. This approach works well in static and structured environments, where the Field D* path can be quite accurately tracked using the local planner.

For an unstructured driving scenario, the situation is more complicated because tracked dynamic obstacles may interfere entirely with the global path. Thus, it may not be possible to track a planned path using a local planner. Instead, we need to evaluate a more general set of possible local trajectories for the vehicle to execute, including some that do not follow the path reported by

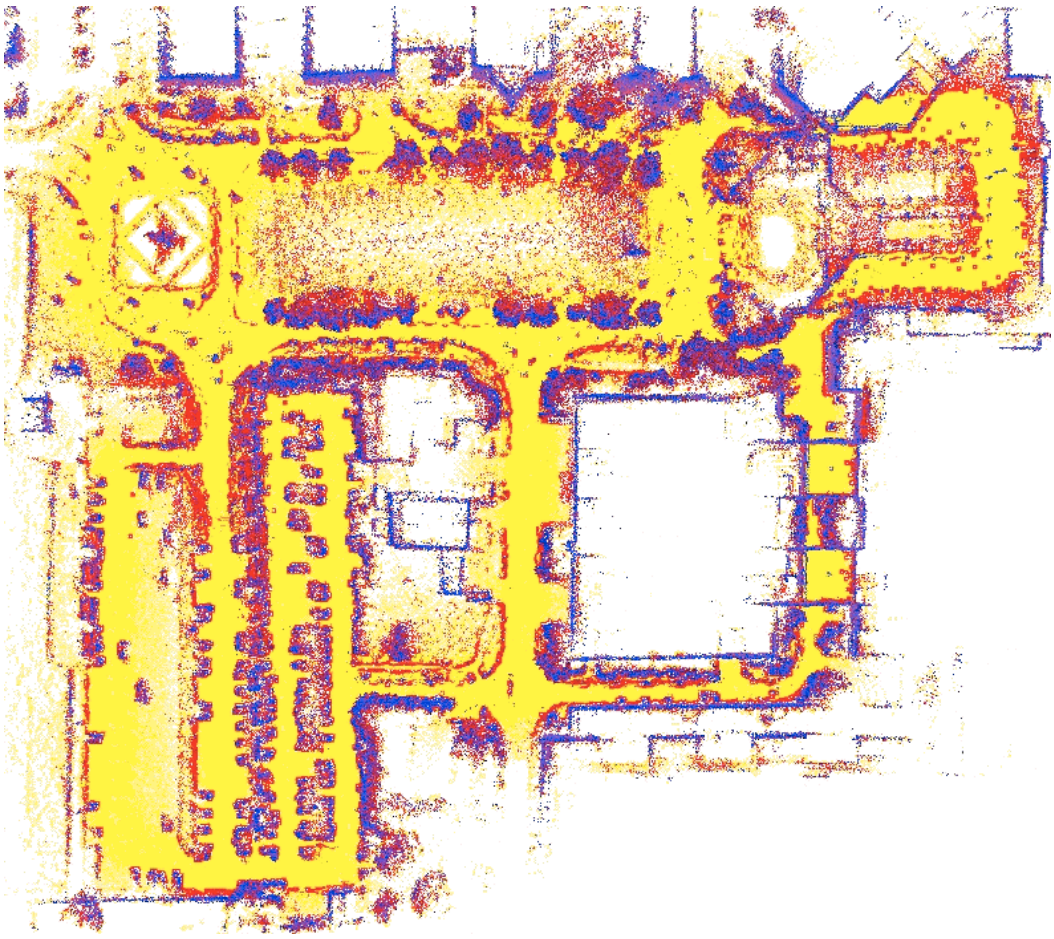


Figure 6.6: Top view of a MLS map with a cell size of 50cm x 50cm showing the EPFL campus (compare to the ortho-photo in Figure 6.5). The yellow surface patches are classified as traversable. The area scanned by the robot spans approximately 300 by 250 meters. During the data acquisition, the robot traversed five nested loops with a length of approximately 2,300m.

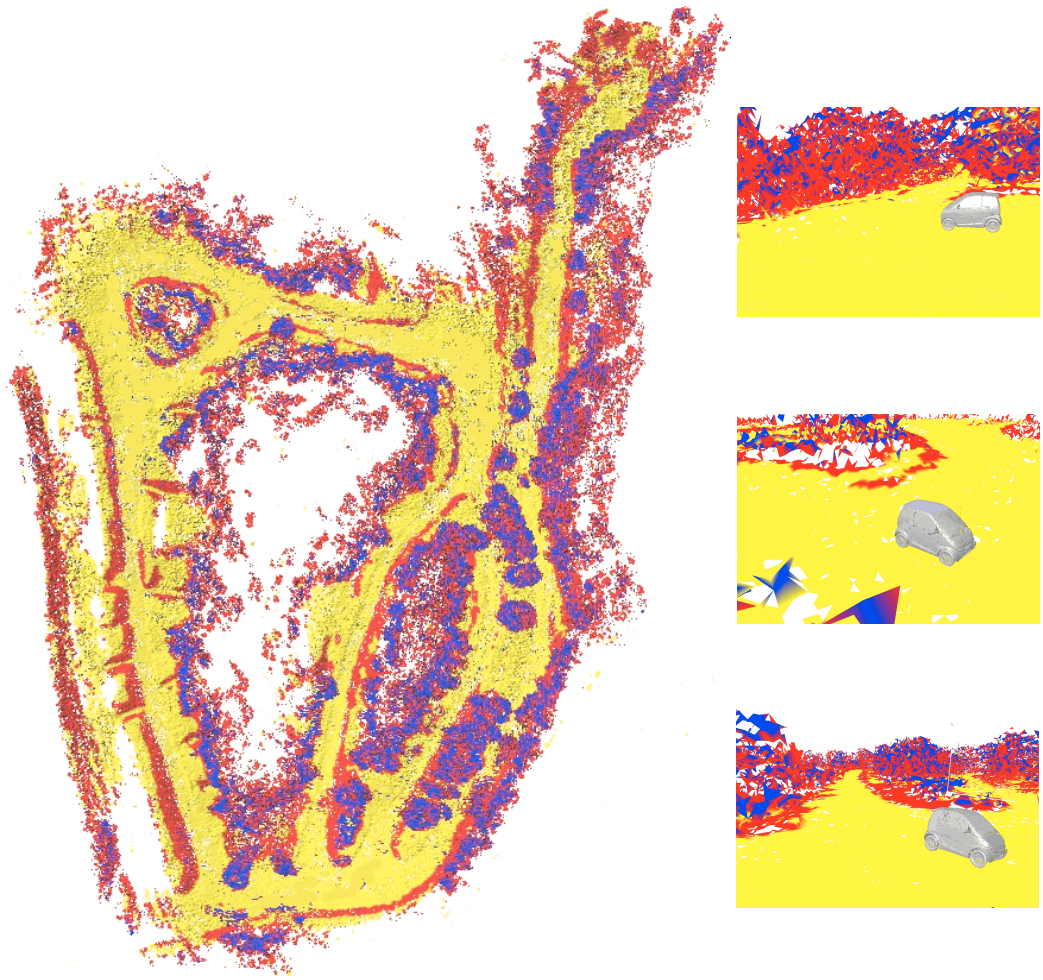


Figure 6.7: The left-hand image shows a top view of a MLS map of a military test site with a cell size of 50cm x 50cm. During the data acquisition, the robot traversed three nested loops with a length of approximately 1,200m. On the right-hand side three cutouts with the visualized Smart car are depicted. The yellow surface patches are classified as traversable.

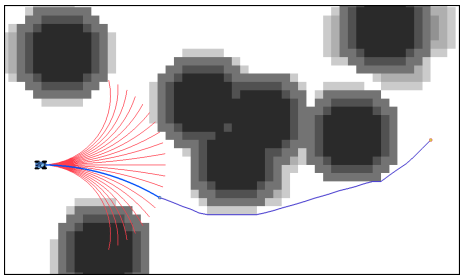


Figure 6.8: Motion planning illustration: A set of feasible local trajectories is generated (shown in red). The cost of each of these trajectories is computed based on the cost of the cells the trajectory travels and curvature information. A global path is planned from the end of each local trajectory to the goal and the cost of this path is added to the cost of the trajectory. The best trajectory is shown in blue, along with the global path from the end of this trajectory to the goal.

the planning algorithm. To achieve this, we use an approach that follows a large body of work on outdoor mobile robot navigation (see [40]). We use a local planner to generate a set of possible local trajectories and then evaluate each trajectory based on both, the cost of the trajectory itself (in terms of curvature, terrain, distance, etc), as well as the cost of a global path from the endpoint of the local trajectory to the goal. Rather than planning a single global path from the current vehicle position to the goal, global paths are planned from each local trajectory endpoint. Figure 6.8 shows an illustrative example of such a combined approach. Here, a set of local arc-based trajectories are shown in red, with the best trajectory shown in blue. The best trajectory was selected based on a combination of the cost of the trajectory itself and the cost of a global path from the end of the trajectory to the goal (the goal is shown as a filled circle at the right of the figure). The global path from the end of the best trajectory to the goal is also shown in blue.

In sum, the combination of techniques presented here [C23], [W4], [W5] are well-suited to set up an autonomous, computer-controlled car that can perform basic driving. The developed system is a step towards intelligent cars. There is, however, a large gap between current systems and human-like driving capabilities.

6.4 Learning by Demonstration for Acquiring Manipulative Skills

Several techniques exist for transferring new skills to robots. A promising technique is called "imitation learning" or "learning by demonstration": Here, a robotic system observes an instructor that is performing a task [4; 6]. From multiple demonstrations, the robot has to infer a generalized task description and to reproduce it accordingly even under modified conditions.

Teaching skills by direct demonstration is a very natural way of skill transfer in humans and animals. In the aim to create versatile, adaptable, and sociable robotic platforms, research on the mechanisms of learning new behaviors by observation has a very high potential. Furthermore, learning from demonstrations can speed up the learning of complex behaviors enormously as it provides strong prior information for the learning process. This can reduce the search space for traditional learning algorithms significantly such that previously intractable tasks can be learned. In addition to that, teaching a robot by means of demonstrations can be carried out by non-experts.

In our work [C9], we show that imitation learning is well suited as a user-friendly instruction method for manipulation tasks. Our approach uses motion capture data generated by a vision system to track body parts of a human instructor and the 3D positions of relevant objects in the scene. The body configuration as well as the relations between objects and body parts of the demonstrator are in turn modeled as normally distributed observation nodes of a dynamic Bayesian network (DBN). For reproducing an observed skill, the network is evaluated at each time step in order to infer the most-likely action. Within our framework, new constraints can be dynamically added to the network, e.g., to incorporate collision avoidance during reproduction in order to deal with unforeseen obstacles.

Our relation-based approach extends the recent work of Calinon and Billard [11] and formalize the problem by means of a DBN. We furthermore allow for incorporating additional constraints for modeling unexpected obstacles that should be considered during imitation.

To illustrate the capabilities of our method [C9], Fig. 6.9 illustrates a reproduction of a whiteboard cleaning task by the robot that has been demonstrated by a human. First, a human repeatedly cleaned a whiteboard in an area bounded by 4 markers. Then, we attached a sponge to the robot and let it perform the demonstrated task. The corresponding photos are depicted in Fig. 6.9. A second example illustrates the capabilities of the robot to reproduce tasks in the presence of unknown obstacles. We showed the robot during reproduction phase an obstacle marker that was not there during the learning phase, see first image of Fig. 6.10. Then, the robot had to clean the whiteboard while avoiding obstacles (and thus not cleaning the area of the marker). For reasons of illustration, we removed the marker during the experiment but kept it in the internal memory of the robot. In this way, the reader can see that the robot did not clean the corresponding area. Eight photos were taken during the reproduction and are depicted in Fig. 6.10. To avoid the area marked as an obstacle area, the robot lifts the sponge away from the whiteboard (in the direction of the observing camera).

In sum, the presented an approach to imitation learning enables a robot to observe, generalize, and reproduce tasks from observing a human demonstrator. We formalized the problem using a dynamic Bayesian network that is used for learning relations between the observed positions of the

Figure 6.9: The reproduction of the board cleaning task by our robot. It imitates the zig-zag movement for cleaning the board with the sponge. Note that the learned task representation allows for cleaning differently sized surfaces based on the markers.

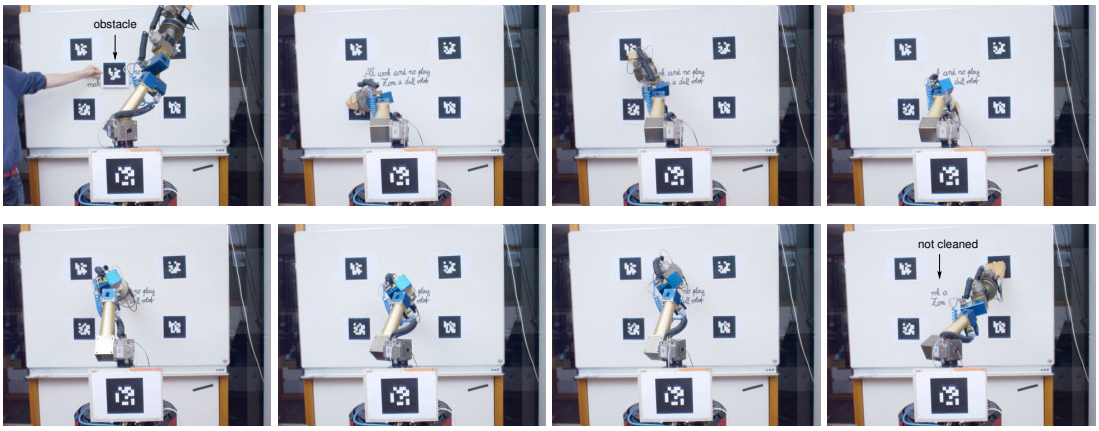


Figure 6.10: A cleaning task in the presence of an obstacle. Initially, the position of the obstacle is shown to the system. Then, the robot cleans the board avoiding the obstacle. As can be seen in the last frame, parts of the text in the area of the obstacle marker was not wiped out.

objects and body parts of the instructor. Additional constraints, for example, to avoid unforeseen obstacles can be added online. To imitate the action of a human, we estimate the actions that maximize the joint probability distribution represented by the DBN. We evaluated the approach and showed that a real robot equipped with a manipulator can learn and reproduce demonstrated actions. Based on a pick-and-place and a whiteboard cleaning task, we illustrated the flexibility of the method to generalize over different spatial setups.

Chapter 7

Conclusion

In this cumulative habilitation thesis, we summarized the developments and achievements made in the context of learning models for mobile robots and autonomous robot navigation. A key concept found in most of the presented approaches are probabilistic techniques explicitely considering the uncertainty in the data. We developed innovative solutions to several important problems in robotics, including:

- Various innovative techniques for efficient large scale robot map learning under significant uncertainty as well as an appropriate benchmark solution for SLAM systems.
- Effective sensor models for Monte-Carlo localization techniques that better exploit the information contained in the sensor data.
- Efficient probabilistic solutions to typical state estimation problems in robotics using different sensor modalities, including proximity data and visual data, gas sensors, and tactile sensing.
- A novel technique for learning hybrid maps modeling metric, semantic, and topological information.
- Supervised and unsupervised approaches for scene analysis and object modeling that allow the robot to better understand its surroundings.
- Efficient strategies for single- and multi robot exploration.
- A novel solution for robot navigation in environments containing deformable objects.
- A computer-controlled car that is able to autonomously navigate, localize, and map its environment.
- A learning by demonstration framework for teaching manipulation tasks.

Most of the developed techniques provide fundamental solutions to the addressed problems and are based on probabilistic techniques. They offer robust solutions that can be applied under significantly changed settings. This is illustrated by the fact that a series of techniques have been applied to different robotic platforms, ranging from computer-controlled cars, over wheeled indoor robots to humanoids and light-weight flying vehicles. In addition to that, we presented a variety of solutions for different sensor modalities including laser range information, camera data, inertial, tactile as well as gas sensors. All approaches have been evaluated on real robotic platforms, in

realistic settings, often using publicly available datasets to allow for benchmarking. Whenever possible, the techniques developed have been statistically evaluated. As presented in the individual papers, all techniques show a significant improvement over the state-of-the-art in robotics.

We see the approaches presented here as building blocks for achieving integrated autonomous systems that will support us in our everyday life. Despite the encouraging results reported in this habilitation thesis, there is, of course, space for further improvements. Robots still act based on what humans program. Furthermore, a large amount of background knowledge that we as humans have learned during our whole life is not easily accessible for robots. There is large list of capabilities robots need acquire until we can call them truly autonomous agents. Nevertheless, we are approaching this goal and a series of tasks that have been considered as difficult ten years ago can be solved rather effectively today, also using techniques presented here.

Journal Articles

[J1] C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin. A nonparametric learning approach to range sensing from omnidirectional vision. *Robots and Autonomous Systems*. Conditionally accepted.

[J2] H. Kretzschmar, G. Grisetti, and C. Stachniss. Life-long map learning for graph-based simultaneous localization and mapping. *KI – Kuenstliche Intelligenz*. Accepted for publication.

[J3] R. Kuemmerle, B. Steder, M. Ruhnke, G. Grisetti, C. Stachniss, C. Dornhege, and A. Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*. In press.

[J4] C. Stachniss, C. Plagemann, and A.J. Lilienthal. Gas distribution modeling using sparse gaussian process mixtures. *Autonomous Robots*, 26:187ff, 2009.

[J5] K.M. Wurm, C. Stachniss, and G. Grisetti. Bridging the gap between feature- and grid-based slam. *Robots and Autonomous Systems*, 2009. In press.

[J6] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009. In press.

[J7] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual slam for flying vehicles. *IEEE Transactions on Robotics*, 24(8):1088–1093, 2008.

[J8] C. Stachniss, G. Grisetti, O. Martínez-Mozos, and W. Burgard. Efficiently learning metric and topological maps with autonomous service robots. *it – Information Technology*, 49(4):232–238, 2007.

[J9] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and accurate slam with rao-blackwellized particle filters. *Robots and Autonomous Systems*, 55(1):30–38, 2007.

Conference Papers

[C1] J. Sturm, V. Predeap, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard. Learning kinematic models for articulated objects. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Pasadena, CA, USA, 2009. To appear.

[C2] F. Enders, C. Plagemann, C. Stachniss, and W. Burgard. Scene analysis using latent dirichlet allocation. In *Proc. of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

[C3] K.M. Wurm, R. Kuemmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009. To appear.

[C4] W. Burgard, B. Steder, R. Kuemmerle, M. Ruhnke, G. Grisetti, C. Stachniss, C. Dornhege, A. Kleiner, and Juan D. Tardos. How to compare the results of slam algorithms. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009. To appear.

[C5] A. Schneider, J. Sturm C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009. To appear.

[C6] H. Strasdat, C. Stachniss, and W. Burgard. Which landmark is useful? learning selection policies for navigation in unknown environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[C7] B. Frank, C. Stachniss, R. Schmedding, W. Burgard, and M. Teschner. Real-world robot navigation amongst deformable obstacles. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[C8] M. Bennewitz, C. Stachniss, S. Behnke, and W. Burgard. Utilizing reflection properties of surfaces to improve mobile robot localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[C9] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[C10] H. Kretzschmar, C. Stachniss, C. Plagemann, and W. Burgard. Estimating landmark locations from geo-referenced photographs. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[C11] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[C12] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[C13] C. Stachniss, C. Plagemann, A.J. Lilienthal, and W. Burgard. Gas distribution modeling using sparse gaussian process mixture models. In *Proc. of Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

[C14] C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[C15] B. Frank, M. Becker, C. Stachniss, M. Teschner, and W. Burgard. Efficient path planning for mobile robots in environments with deformable objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[C16] C. Plagemann, F. Endres, J. Hess, C. Stachniss, and W. Burgard. Monocular range sensing: A non-parametric learning approach. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[C17] G. Grisetti, D. Lordi Rizzini, C. Stachniss, E. Olson, and W. Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[C18] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard. Evaluation of gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[C19] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[C20] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3d using attitude and noisy vision sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[C21] K.M. Wurm, C. Stachniss, G. Grisetti, and W. Burgard. Improved simultaneous localization and mapping using a dual representation of the environment. In *Proc. of the European Conference on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

[C22] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[C23] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, W. Burgard, and R. Siegwart. Towards mapping of cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[C24] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Speeding-up rao-blackwellized slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 442–447, Orlando, FL, USA, 2006.

[C25] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 143–157. Springer-Verlag Berlin Heidelberg, Germany, 2006.

Workshop Papers

[W1] P. Pfaff, R. Kuemmerle, D. Joho, C. Stachniss, R. Triebel, and W. Burgard. Navigation in combined outdoor and indoor environments using m ulti-level surface maps. In *Workshop on Safe Navigation in Open and Dynamic Environments a t the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[W2] H. Strasdat, C. Stachniss, M. Bennewitz, and W. Burgard. Visual bearing-only simultaneous localization and mapping with improved feature matching. In *Fachgespräche Autonome Mobile Systeme (AMS)*, Kaiserslautern, Germany, 2007.

[W3] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous exploration for 3d map learning. In *Fachgespräche Autonome Mobile Systeme (AMS)*, Kaiserslautern, Germany, 2007.

[W4] P. Lamon, C. Stachniss, R. Triebel, P. Pfaff, C. Plagemann, G. Grisetti, S. Kolski, W. Burgard, and R. Siegwart. Mapping with an autonomous car. In *Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[W5] S. Kolski, D. Furgeson, C. Stachniss, and R. Siegwart. Autonomous driving in dynamic environments. In *Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[2] K.O. Arras, R. Philippsen, N. Tomatis, M. de Battista, M. Schilt, and R. Siegwart. A navigation framework for multiple mobile robots and its application at the Expo.02 exhibition. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[3] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. In *IEEE Transactions on Signal Processing*, volume 50, pages 174–188, 2002.

[4] P. Bakker and Y. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *In AISB96 Workshop on Learning in Robots and Animals*, pages 3–11, 1996.

[5] T.D. Barfoot. Online visual motion estimation using FastSLAM with SIFT features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[6] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008.

[7] D.M. Blei, A.Y. Ng, M.I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[8] O. Brock and O. Khatib. High-speed naviagation using the global dynamic window approach. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[9] E. Brunskill, T. Kollar, and N. Roy. Topological mapping using spectral clustering and classification. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, October 2007.

[10] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[11] S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling skill constraints in joint space and task space. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[12] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, 2005.

[13] L.B. Cremean, T.B. Foote, J.H. Gillula, G.H. Hines, D. Kogan, K.L. Kriechbaum, J.C. Lamb, J. Leibs, L. Lindzey, C.E. Rasmussen, A.D. Stewart, J.W. Burdick, and R.M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal on Field Robotics*, 2006.

[14] G. Csurka, L. Dance, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. of the European Conf. on Computer Vision (ECCV), Wokshop on Statistical Learning in Computer Vision*, pages 59–74, 2004.

[15] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *Int. Journal of Robotics Research*, 27(6):647–665, 2008.

[16] DARPA. Darpa grand challenge rulebook. Website, 2004. http://www.darpa.mil/grandchallenge05/Rules 8oct04.pdf.

[17] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[18] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[19] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1009–1014, San Francisco, CA, USA, 2000.

[20] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 176–183, Stanford, CA, USA, 2000.

[21] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Journal of Autonomous Robots*, 12(3):287 – 300, 2002.

[22] DustBot. DustBot - Networked and Cooperating Robots for Urban Hygiene. http://www.dustbot.org, 2008.

[23] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[24] P. Elinas and J.J. Little. σMCL: Monte-Carlo localization for mobile robots with stereo vision. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[25] D. Ferguson and A. Stentz. Field d*: An interpolation-based path planner and replanner. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2005.

[26] J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2004.

[27] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 1997.

[28] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[29] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[30] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In Manuela M. Veloso, editor, *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 2109–2114, 2007.

[31] B.P. Gerkey and M.J. Matarić. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758– 768, 2002.

[32] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5228–5235, 2004.

[33] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[34] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[35] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, CA, USA, 1999.

[36] Y.S. Ha and H.H. Kim. Environmental map building for a mobile robot using infrared range-finder sensors. *Advanced Robotics*, 18(4):437–450, 2004.

[37] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, Las Vegas, NV, USA, 2003.

[38] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1055–1060, 2001.

[39] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[40] A. Kelly. *An intelligent predictive control approach to the high speed cross country autonomous navigation problem*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1995.

[41] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. Int. Conf. on Intelligent Autonomous Systems (IAS'4)*, 1995.

[42] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.

[43] K. Konolige and K. Chou. Markov localization using correlation. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1999.

[44] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1):83–97, 1955.

[45] H.W. Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics*, 3:253–258, 1956.

[46] R. Kümmerle, D. Hähnel, D. Dolgov, S. Thrun, and W. Burgard. Autonomous driving in a multi-level parking structure. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3395–3400, Kobe, Japan, 2009.

[47] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[48] S.M. LaValle. *Planning Algorithms*. Cambridge Univ. Press, 2006.

[49] J.J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 169–179, Breckenridge, CO, USA, 2000.

[50] A. Lilienthal and T. Duckett. Building Gas Concentration Gridmaps with a Mobile Robot. *Robotics and Autonomous Systems*, 48(1):3–16, 2004.

[51] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[52] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

[53] O. Martínez-Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1742–1747, Barcelona, Spain, 2005.

[54] J. Minguez and L. Montano. Nearness diagram navigation (nd): A new real time collision avoidance approach. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2094–2100, 2000.

[55] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

[56] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1985–1991, Taipei, Taiwan, 2003.

[57] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[58] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[59] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[60] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[61] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping and loop closing. *Int. Journal of Robotics Research*, 2007.

[62] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A non-parametric bayesian measurement model for range finders. In *Robotics: Science and Systems (RSS)*, Atlanta, Georgia, USA, June 2007.

[63] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[64] P.J.W. Roberts and D.R. Webster. Turbulent Diffusion. In H. Shen, A. Cheng, K.-H. Wang, M.H. Teng, and C. Liu, editors, *Environmental Fluid Mechanics - Theories and Application*. ASCE Press, Reston, Virginia, 2002.

[65] S. Se, D.G. Lowe, and J.J. Little. Global localization using distinctive visual features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[66] R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.

[67] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 254–259, Atlanta, GA, USA, 1993.

[68] J. Spillmann. Defcol studio. http://www.beosil.com/download/DefColStudio_readme.txt, 2005. Last visited: 2009/06/10.

[69] J. Spillmann, M. Becker, and M. Teschner. Non-iterative computation of contact forces for deformable objects. *Journal of WSCG*, 15(1–3):33–40, 2007.

[70] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, April 2006.

[71] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

[72] C. Stachniss, O. Martínez-Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1692–1697, Orlando, FL, USA, 2006.

[73] C. Stachniss, O. Martinez Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52:205ff, 2009.

[74] G. Swaminathan and S. Grossberg. Laminar cortical mechanisms for the perception of slanted and curved 3-D surfaces and their 2-D pictorical projections. *Journal of Vision*, 2(7):79–79, 11 2002.

[75] M. Teschner, B. Heidelberger, M. Mueller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *Proc. of Computer Graphics International*, pages 312–319, 2004.

[76] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization (VMV)*, pages 47–54, 2003.

[77] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[78] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[79] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

[80] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[81] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal on Field Robotics*, 2006.

[82] V. Tresp. Mixtures of gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2000.

[83] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[84] C. Urmson. *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2005.

[85] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Second International Conference on Autonomous Agents*, pages 47–53, Minneapolis, MN, USA, 1998.

[86] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.

[87] Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building and planning based on graph partitioning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 803–809, 2006.

[88] R. Zlot, A.T. Stenz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Washington, DC, USA, 2002.

Part II

Publications

[J1] C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin. A nonparametric learning approach to range sensing from omnidirectional vision. *Robots and Autonomous Systems*. Accepted for publication.

A Nonparametric Learning Approach to Range Sensing from Omnidirectional Vision

Christian Plagemann[a], Cyrill Stachniss[b], Jürgen Hess[b],
Felix Endres[b], Nathan Franklin[b]

[a] *Stanford University, Computer Science Dept., 353 Serra Mall, Stanford, CA 94305-9010*
[b] *University of Freiburg, Dept. of CS, Georges-Koehler-Allee 79, 79110 Freiburg, Germany*

Abstract

We present a novel approach to estimating depth from single omnidirectional camera images by learning the relationship between visual features and range measurements available during a training phase. Our model not only yields the most likely distance to obstacles in all directions, but also the predictive uncertainties of these estimates. This information can be utilized by a mobile robot to build an occupancy grid map of the environment or to avoid obstacles during exploration—tasks that typically require dedicated proximity sensors such as laser range finders or sonars. We show in this paper how an omnidirectional camera can be used as an alternative to such range sensors. As the learning engine, we apply Gaussian processes, a nonparametric approach to function regression, as well as a recently developed extension for dealing with input-dependent noise. In practical experiments carried out in different indoor environments with a mobile robot equipped with an omnidirectional camera system, we demonstrate that our system is able to estimate range with an accuracy comparable to that of dedicated sensors based on sonar or infrared light.

Key words: omnidirectional vision, learning, range sensing, Gaussian processes

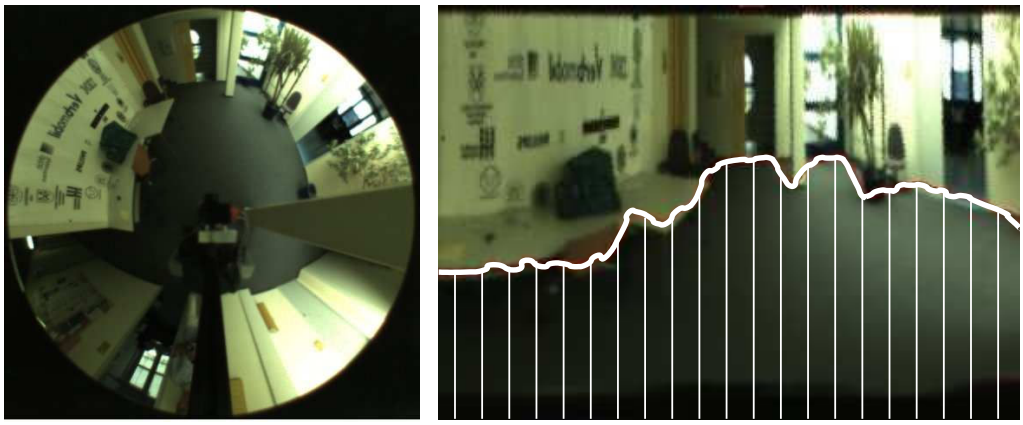


Figure 1: Our system records intensity images (left) and estimates the distances to nearby obstacles (right) after having learned how visual appearance is related to depth.

1. Introduction

The major role of perception, in humans as well as in robotic systems, is to discover geometric properties of the current scene in order to act in it reasonably and safely. For artificial systems, omnidirectional vision provides a rich source of information about the local environment, since it captures the entire scene—or at least the most relevant part of it—in a single image. Much research has thus concentrated on the question of how to extract geometric scene properties, such as distances to nearby objects, from such images.

This task is complicated by the fact that only a projection of the scene is recorded and, thus, it is not possible to sense depth information directly. From a geometric point of view, one needs at least two images taken from different locations to recover the depth information analytically. An alternative approach that requires just one monocular camera image and that we follow here, is to learn from previous experience how visual appearance is related to depth. Such an ability is also highly developed in humans, who are able to utilize monocular cues for depth perception [1]. As a motivating example, consider the right image in Figure 1, which shows the image of an office environment (180° of the omnidirectional image on the left warped to a panoramic view). Overlayed in white, we visualize the most likely area of free space that is predicted by our approach. We explicitly do not try to estimate a depth map for the whole image, as for example done by Saxena *et al.* [2]. Rather, we aim at learning the function that, given an image, maps measurement directions to their corresponding distances to the

Figure 2: Reflections, glass walls and inhomogeneous surfaces make the relationship between visual appearance and depth hard to model. One of the test environments at the University of Freiburg (left) exhibits many of these factors. Our approach was also tested using a standard perspective camera in this challenging environment (right).

closest obstacles. Such a function can be utilized to solve various tasks of mobile robots including local obstacle avoidance, localization, mapping, exploration, or place classification.

The contribution of this paper is a new approach to range estimation based on omnidirectional images. The task is formulated as a supervised regression problem in which the training set is acquired by combining image date with proximity information provided by a laser range finder. We explain how to extract appropriate visual features from the images using algorithms for edge detection as well as for supervised and unsupervised dimensionality reduction. As a learning framework in our proposed system, we apply Gaussian processes since this technique is able to model non-linear functions, offers a direct way of estimating uncertainties for its predictions, and it has proven successful in previous work involving range functions [3].

The paper is organized as follows. First, we discuss related work in Section 2. Section 3 introduces the used visual features and how they can be extracted from images. We then formalize the problem of predicting range from these features and introduce the proposed learning framework in Section 4. In Section 5, we present the experimental evaluation of our algorithm as well as an application to the mapping problem.

2. Related Work

Estimating the geometry of a scene based on visual input is one of the fundamental problems in computer vision and is also frequently addressed in the robotics literature. Monocular cameras do not directly provide 3D information and therefore stereo systems are widely used to estimate the missing depth information. Stereo systems either require a careful calibration to analytically calculate depth using geometric constraints or, as Sinz *et al.* [4] demonstrated, can be used in combination with non-linear, supervised learning approaches to recover depth information. Often, sets of features such as SIFT [5] or SURF [6] are extracted from two images and matched against each other. Then, the feature pairs are used to constrain the poses of the two camera locations and/or the point in the scene that corresponds to the image feature. In this spirit, the motion of a single camera has been used by Davidson *et al.* [7] and Strasdat *et al.* [8] to estimate the location of landmarks in the environment. In their work, Mikusic and Padjla [9] have proposed a similar approach for recovering 3D structure from sequences of omnidirectional images.

Sim and Little [10] present a stereo-vision based approach to the SLAM problem, which includes the recovery of depth information. Their approach contains both the matching of discrete landmarks and dense grid mapping using vision.

An active way of sensing depth using a single monocular camera is known as *depth from defocus* [11] or *depth from blur*. Such approaches typically adjust the focal length of the camera and analyze the resulting local changes in image sharpness. Torralba and Oliva [12] present an approach for estimating the mean depth of full scenes from single images using spectral signatures. While their approach is likely to improve a large number of recognition algorithms by providing a rough scale estimate, the spatial resolution of their depth estimates does not appear to be sufficient for the problem studied in this paper. Dahlkamp *et al.* [13] learn a mapping from visual input to road traversability in a self-supervised manner. They use the information from laser range finders to estimate the terrain traversability locally and then use visual data to extend the prediction to areas outside the field of view of the laser range scanners. In contrast to our method, the laser range data is used at all times since learning is not a separated process as in this paper. Furthermore, different learning techniques and different features have been applied.

The problem addressed by Saxena *et al.* [2] is closely related to our paper. They utilize Markov random fields (MRFs) for reconstructing dense depth maps from single monocular images. Compared to these methods, our Gaussian process

formulation provides the predictive uncertainties for the depth estimates directly, which is not straightforward to achieve in an MRF model. An alternative approach that predicts 2D range scans using reinforcement learning techniques has been presented by Michels *et al.* [14]. Menegatti *et al.* [15] proposed to simulate range scans from detected color transitions in omnidirectional images and to apply scan-matching and Monte-Carlo methods for localizing a mobile robot. Such color transitions are comparable to our set of edge-based features described in Section 3.3, which form the low-level input to the learning approach described in this paper.

Hoiem *et al.* [16] developed an approach to monocular scene reconstruction based on local features combined with global reasoning. Whereas Han and Zhu [17] presented a Bayesian method for reconstructing the 3D geometry of wire-like objects in simple scenes, Delage *et al.* [18] introduced an MRF model on orthogonal plane segments to recover the 3D structure of indoor scenes. Ewert *et al.* [19] extract depth cues from monocular image sequences in order to facilitate image retrieval from video sequences. Their major cue for feature extraction is the motion parallax. Thus, their approach assumes translational camera motion and a rigid scene.

In own previous work [3], we applied Gaussian processes to improve sensor models for laser range finders. In contrast to that, the goal here is to exchange the highly accurate and reliable laser measurements by noisy and ambiguous vision features.

As mentioned above, one potential application of the approach described in this paper is to learn occupancy grid maps. This type of maps and an algorithm to update such maps based on ultrasound data has been introduced by Moravec and Elfes [20]. In the past, different approaches to learn occupancy grid maps from stereo vision have been proposed [21, 22]. If the positions of the robot are unknown during the mapping process, the entire task turns into the so-called simultaneous localization and mapping (SLAM) problem. Vision-based techniques have been proposed by Elinas *et al.* [23] and Davison *et al.* [7] to solve this problem. In contrast to the mapping approach presented in this paper, these techniques mostly focus on landmark-based representations.

The contribution of this paper is a novel approach to estimating the proximity to nearby obstacles in indoor environments from a single camera image. It is an extension of our recent conference paper [24] that first presented the idea of estimating depth from camera images using GP regression. The work presented here additionally considers supervised dimensionality reduction, namely LDA, which allows us to find a low dimensional space in which feature vectors corresponding

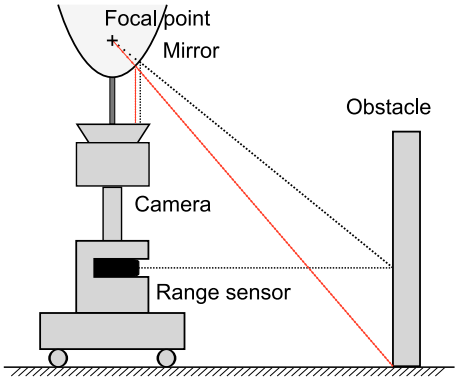


Figure 3: Our experimental setup. The training set was recorded using a mobile robot equipped with an omnidirectional camera (monocular camera with a parabolic mirror) as well as a laser range finder.

to different range measurements are better separated. In this way, the Gaussian process is able to provide better estimates about predicted ranges.

3. Omnidirectional Vision and Feature Extraction

The task of estimating range information from images requires us to learn the relationship between visual input and the extent of free space around the robot. Figure 3 depicts the configuration of our robot used for data acquisition. An omnidirectional camera system (catadioptric with a parabolic mirror) is mounted on top of a SICK laser range finder. This setup allows the robot to perceive the whole surrounding area at every time step as the two example images in Figure 2 illustrate. It furthermore enables the robot to collect proximity data from the laser range finders and relate them to the image data. As a result, our robot can easily acquire training data used in the regression task. The left images in Figure 1 and Figure 2 show typical situations from the two benchmark data sets used in this paper. They have been recorded at the University of Freiburg (Figure 1) and at the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken (Figure 2). By considering these example images, it is clear that the part of an omnidirectional image which is most strongly correlated with the distance to the nearest obstacle in a certain direction α is the strip of pixels oriented in the same direction covering the area from the center of the image to its margins. With the type of camera used in our experiments, such strips have a dimensionality of 420 (140 pixels, each having a *hue*, *saturation*, and a *value* component). To make

these strips easily accessible to filter operators, we warp the omnidirectional images into panoramic views (e.g., the right image in Figure 2) so that angles in the polar representation now correspond to column indices in the panoramic one. This transformation allows us to replace complicated image operations in the polar domain by easier and more robust ones in a Cartesian coordinate system.

In the following, we denote with $\mathbf{x}_i \in \mathbb{R}^{420}$ the individual pixel columns of an image and with $y_i \in \mathbb{R}$ the range values in the corresponding direction, that is, the distances to the closest obstacles, respectively. Before describing how to learn the relationship between the variables $\mathbf{x}$ and y, we discuss three alternative ways of extracting meaningful low-dimensional features $\mathbf{v}$ from $\mathbf{x}$ which can be utilized by the learning algorithm. The first approach applies unsupervised dimensionality reduction (PCA) to compute low-dimensional features. As an alternative, we also consider the linear discriminant analysis (LDA) as an supervised alternative to obtain low-dimensional features. Finally, we discuss the use of manually designed features extracted from the images that can be used for range prediction.

3.1. Unsupervised Dimensionality Reduction

Principal component analysis (PCA) is arguably the most common approach to dimensionality reduction. We apply PCA for reducing the complexity of the data to the raw 420-dimensional pixel vectors that are contained in the columns of the panoramic images. In our approach, the PCA is implemented using eigenvalue decomposition of the covariance matrix of the 420-dimensional training vectors. PCA computes a linear transformation that maps the input vectors onto a new basis so that their dimensions are ordered by the amount of variance of the data set they carry. By selecting only the first k vectors of this basis representing the dimensions with the highest variance in the data, one obtains a low-dimensional representation without losing a large amount of information. The left diagram in Figure 4 shows the remaining fraction of variance after truncating the transformed data vectors after a certain number of components. The right diagram in the same figure shows the 420 components of the first eigenvector for the Freiburg data set grouped by hue, saturation, and value. Our experiments revealed that the value channel of the visual input is more important than hue and saturation for our task.

For the experiments reported on in Section 5, we trained the PCA on 600 input images and retained the first six principal components. This results in a reduction from 420-dimensional input vectors to 6-dimensional ones. The GP model, described in the Section 4, is then learned with these 6D features and is named *PCA-GP* in the experimental section.

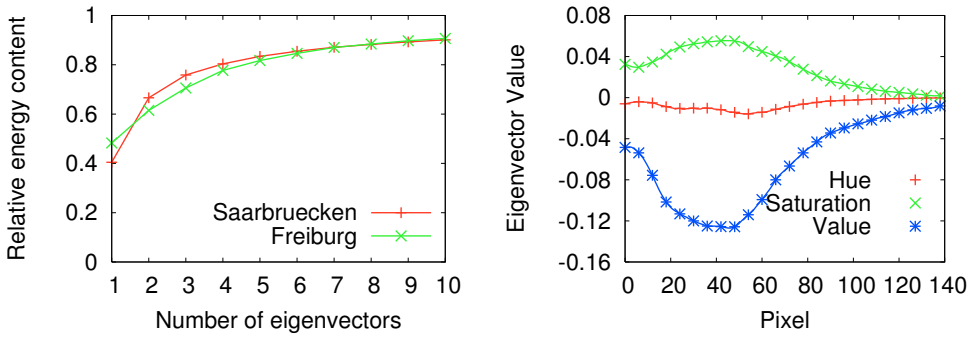


Figure 4: Left: The amount of variance explained by the first principal components (eigenvectors) of the pixel columns in the two data sets. Right: The 420 components of the first eigenvector of the Freiburg data set.

3.2. Supervised Dimensionality Reduction

A drawback of PCA in our regression task is the fact that it does not consider the range values y_i when reducing the dimensionality of the input vectors $\mathbf{x}_i$. In this way, it treats all components of the input vectors equally—no matter how much information they actually carry about the range to be predicted. It can thus be expected that *supervised* dimensionality reduction, where external, dependent variables are considered explicitly, can lead to more accurate predictions. See Alpaydin [25] for an overview of approaches and comparisons. One such technique is the linear discriminant analysis (LDA). LDA is related to PCA in that it also assumes a linear transformation between the original space and the reduced one. But in contrast to PCA, it allows each data point to be given a class label. LDA seeks a low-dimensional space in which the classes of the dataset are separated best as illustrated in Figure 5 for a reduction from $\mathbb{R}^2$ to $\mathbb{R}$.

Let K be the number of classes $\mathcal{C}_i$ and $\mathbf{x}_i$ the d-dimensional inputs. The objective is to find a $d \times k$ matrix W so that $\mathbf{v}_i = W^T \mathbf{x}_i$ with $\mathbf{v}_i \in \mathbb{R}^k$ and so that the classes $\mathcal{C}_i$ are separated best in terms of distances between the $\mathbf{v}_i$. Let $r_{i,t}$ be an indicator variable with $r_{i,t} = 1$ if $\mathbf{x}_t \in \mathcal{C}_i$ and 0 otherwise. Let $\mathbf{m}_i$ be the mean of d-dimensional vectors $\mathbf{x}_i$. Then, the so-called *scatter matrix* of $\mathcal{C}_i$ is

$$S_i \quad = \quad \sum_t r_{i,t} (\mathbf{x}_t - \mathbf{m}_i)(\mathbf{x}_t - \mathbf{m}_i)^T \,,$$

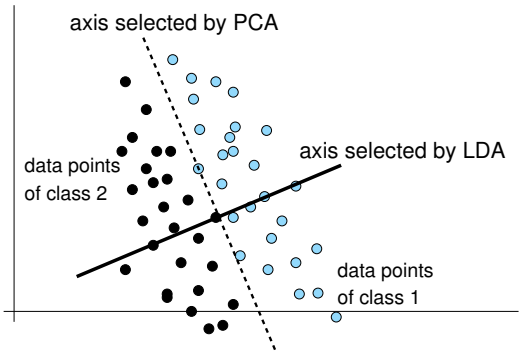


Figure 5: Reduction from $\mathbb{R}^2$ to $\mathbb{R}$ for PCA and LDA: PCA aims to keep the variance in the data while LDA seeks to separate the two classes (illustrated by black and blue) as well as possible.

the *total within-scatter matrix* becomes

$$S_W = \sum_{i=1}^{K} S_i = \sum_{i=1}^{K} \sum_{t} r_{i,t}(\mathbf{x}_t - \mathbf{m}_i)(\mathbf{x}_t - \mathbf{m}_i)^T ,$$

and the *between-class scatter matrix* is

$$S_B = \sum_{i=1}^{K} \left(\sum_{t} r_{i,t} \right) (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T ,$$

with $\mathbf{m} = \frac{1}{K} \sum_{i=1}^{K} \mathbf{m}_i$. Now let us consider the scatter matrices after projecting using W. The between-class scatter matrix after projection is $W^T S_B W$, and the within-scatter matrix accordingly, both are $k \times k$ dimensional. To goal is to determine W in a way that the means of the classes $W^T \mathbf{m}_i$ are as far apart from each other as possible while the spread of their individual projected class samples is small. Similarly to covariance matrices, the determinant of a scatter matrix characterizes the spread and it is computed as the product of the eigenvalues specifying the variance along the eigenvectors. Thus, we aim at finding the matrix W that maximizes

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}.$$

The largest eigenvectors of $S_W^{-1} S_B$ are the solution to this problem.

Applied to the range-regression task, we selected the discretized laser range measurements as class label for each input data point. LDA then projects to a low-dimensional space so that data points corresponding to the discretized range measurements can be separated best. The GP model learned with the low-dimensional features is named *LDA-GP* in our experimental evaluation.

3.3. Edge-based Features

Principal component analysis is an unsupervised method that does not take into account any prior information and also the linear discriminant analysis only uses information about class labels to perform dimensionality reduction to keep the data separated. However, there might be additional information available about the task to be solved—like the fact that distances to the closest obstacles are to be predicted in our case. Driven by the observation that there typically is a strong correlation between the extent of free space and the presence of horizontal edge features in the panoramic image, we also assessed the potential of edge-type features in our approach.

To compute such visual features from the warped images, we apply Laws' convolution masks [26]. They provide an easy way of constructing local feature extractors for discretized signals. The idea is to define three basic convolution masks

- $L_3 = (1, 2, 1)^T$ (Weighted Sum: Averaging),
- $E_3 = (-1, 0, 1)^T$ (First difference: Edges), and
- $S_3 = (-1, 2, -1)^T$ (Second difference: Spots),

each having a different effect on (1D) patterns, and to construct more complex filters by a combination of the basic masks. In our application domain, we obtained good results with the (2D) directed edge filter $E_5 L_5^\top$, which is the outer product of E_5 and L_5. Here, E_5 is a convolution of E_3 with L_3 and L_5 denotes L_3 convolved with itself. After filtering with this mask, we apply an optimized threshold to yield a binary response. This feature type is denoted as *Laws5* in the experimental section. As another well-known feature type, we applied the $E_3 L_3^\top$ filter, i.e. the Sobel operator, in conjunction with Canny's algorithm [27]. This filter yields binary responses at the image locations with maximal gray-value gradients in gradient direction. We denote this feature type as *Laws3+Canny* in Section 5. For both edge detectors, *Laws5* and *Laws3+Canny*, we search along each image

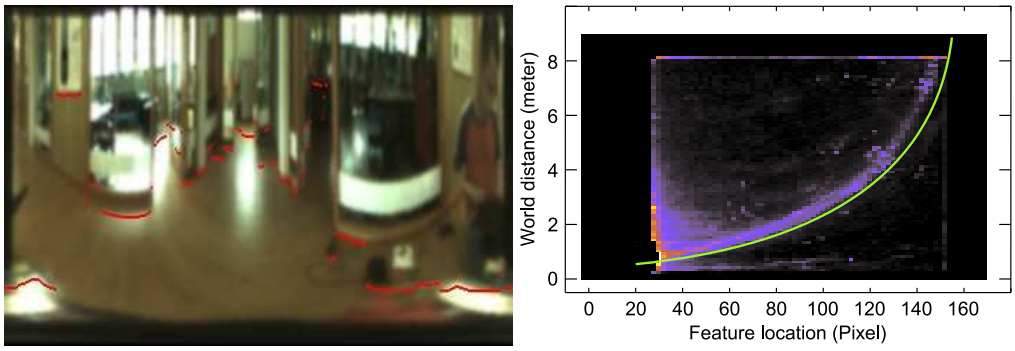


Figure 6: Left: Example *Laws5+LMD* feature extracted from one of the Freiburg images. Right: Histogram for *Laws5+LMD* edge features. Each cell in the histogram is indexed by the pixel location of the edge feature (x-axis) and the length of the corresponding laser beam (y-axis). The optimized (parametric) mapping function that is used as a benchmark in our experiments is overlaid in green.

column for the first detected edge. This pixel index then constitutes the feature value.

To increase the robustness of the edge detectors described above, we applied *lightmap damping* as an optional preprocessing step to the raw images. This means that, in a first step, a copy of the image is converted to gray scale and strongly smoothed with a Gaussian filter, such that every pixel represents the brightness of its local environment. This is referred to as the *lightmap*. The brightness of the original image is then scaled with respect to the lightmap, such that the *value* component of the color is increased in dark areas and decreased in bright areas. In the experimental section, this operation is marked by adding *+LMD* to the feature descriptions. Figure 6 shows *Laws5+LMD* edge features extracted from an image of the Freiburg data set.

All parameters involved in the edge detection procedures described above were optimized to yield features that lie as close as possible to the laser end points projected onto the omnidirectional image using the acquired training set. For our regression model, we can now construct 4D feature vectors $\mathbf{v}$ consisting of the Canny-based feature, the *Laws5*-based feature, and both features with additional preprocessing using lightmap-damping. Since every one of these individual features captures slightly different aspects of the visual input, the combination of all, in what we call the *Feature-GP*, can be expected to yield more accurate predictions than any single one.

As a benchmark for predicting range information from edge features, we also evaluated the individual features directly. For doing so, we fitted a parametric function to training samples of feature-range pairs. This mapping function computes for each pixel location of an edge feature the length of the corresponding laser beam. The right diagram in Figure 6 shows the feature histogram for the *Laws5+LMD* features from one of our test runs that was used for the optimization. The color of a cell (c_x, c_y) in this diagram encodes the relative amount of feature detections that were extracted at the pixel location c_x (measured from the center of the omnidirectional image) and that have a corresponding laser beam with a length of c_y in the training set. The optimized projection function is overlayed in green.

4. Learning Depth from Images

This section presents the learning method used in our approach to find the relationship between visual input and the free space around the robot. Given a training set of images and corresponding range scans acquired in a setting, we can treat the problem of predicting range in *new* situations as a supervised learning problem. The omnidirectional images can be mapped directly to the laser scans since both measurements can be represented in a common, polar coordinate system. Note that our approach is not restricted to omnidirectional cameras in principle. However, the correspondence between range measurements and omnidirectional images is a more direct one and the field of view is considerably larger compared to standard perspective optics.

4.1. Gaussian Processes for Range Predictions

In the spirit of the Gaussian beam processes (GBP) model introduced in [3], we propose to put a Gaussian process (GP) prior on the range function, but in contrast, here we use the visual features $\mathbf{v}$ described in the previous section as indices for the range values rather than the bearing angles α.

We extract for every viewing direction α a vector of visual features $\mathbf{v}$ from an image $\mathbf{c}$ and phrase the problem as learning the range function $f(\mathbf{v}) = y$ that maps the visual input $\mathbf{v}$ to distances y. We learn this function in a supervised manner using a training set $\mathcal{D} = \{\mathbf{v}_i, y_i\}_{i=1}^n$ of observed features $\mathbf{v}_i$ and corresponding laser range measurements y_i. If we place a GP prior (see, e.g., [28]) on the non-linear function f, i.e., we assume that all range samples y indexed by their corresponding feature vectors $\mathbf{v}$ are jointly Gaussian distributed, we obtain

$$y_* \;=\; f(\mathbf{v}_*) \;\sim\; \mathcal{N}(\mu_*, \sigma_*^2) \tag{1}$$

for the noise-free range with

$$\mu_* \;=\; \mathbf{k}_{\mathbf{v}_*\mathbf{v}}^\top (\mathbf{K}_{\mathbf{vv}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{2}$$

$$\sigma_*^2 \;=\; k(\mathbf{v}_*, \mathbf{v}_*) - \mathbf{k}_{\mathbf{v}_*\mathbf{v}}^\top (\mathbf{K}_{\mathbf{vv}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_{\mathbf{v}_*\mathbf{v}} \tag{3}$$

for a new query feature $\mathbf{v}^*$. Here, the matrix $\mathbf{K}_{\mathbf{vv}} \in \mathbb{R}^{n \times n}$ denotes the covariance matrix with $[\mathbf{K}_{\mathbf{vv}}]_{ij} = k(\mathbf{v}_i, \mathbf{v}_j)$. Furthermore, $\mathbf{k}_{\mathbf{v}_*\mathbf{v}} \in \mathbb{R}^n$ is given by $[\mathbf{k}_{\mathbf{v}_*\mathbf{v}}]_i = k(\mathbf{v}_*, \mathbf{v}_i)$, $\mathbf{y} = (y_1, \ldots, y_n)^\top$, and $\mathbf{I}$ is the identity matrix. σ_n denotes the global noise parameter. As covariance function, we apply the squared exponential

$$k(\mathbf{v}_p, \mathbf{v}_q) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2\ell^2} |\mathbf{v}_p - \mathbf{v}_q| \right), \tag{4}$$

where l and σ_f, as well as the global noise parameter σ_n, are the so-called hyper-parameters. A standard way of learning these hyperparameters from data, which we applied in this work, is to maximize the log data likelihood of the training data using scaled conjugate gradients (see, e.g., [28] for details).

A particularly useful property of Gaussian processes for our application is the availability of the predictive uncertainty at every query point. This means that new features $\mathbf{v}_*$ which lie close to points $\mathbf{v}$ of the training set, result in more confident predictions than features which fall into a less densely sampled region of feature space.

4.2. Modeling Angular Dependencies

So far, our model assumes *independent* range variables y_i and it thus ignores dependencies that arise, for instance, because "neighboring" range variables and visual features are likely to correspond to the same object in the environment. Angular dependencies can be included, for example, by (a) explicitly considering the angle α as an additional index dimension in $\mathbf{v}$ or by (b) applying Gaussian beam processes (GBPs) as an independent post-processing step to the *predicted* range scan. While the first variant would require a large amount of additional training data—since it effectively couples the visual appearance and the angle of observation, the second alternative is relatively easy to realize and to implement. Figure 3 gives a graphical representation of the second approach. The gray bars group sets of variables that are fully connected and jointly distributed according to

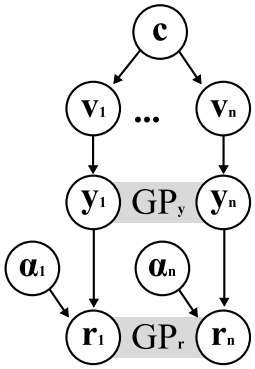


Figure 7: Graphical model for predicting ranges $\mathbf{r}$ from a camera image $\mathbf{c}$. The gray bars group sets of variables that are fully connected and that are jointly distributed according to a GP model.

a GP model. We denote with $\mathcal{GP}_y$ the Gaussian process that maps visual features to ranges and with $\mathcal{GP}_r$ the so-called heteroscedastic GP that is applied as a post-processing step to single, predicted range scans. For $\mathcal{GP}_r$, the task is to learn the mapping $\alpha \mapsto r$ using a training set of *predicted* range values $\mathbf{r}$. Since we do not want to constrain the model to learning from the *mean* predictions $\mu_*(\mathbf{x}_i)$ only, we need a way of incorporating the predictive uncertainties $\sigma_*^2(\mathbf{v}_i)$ for the feature-based range predictions y_*. This can be achieved by not using a fixed noise matrix $\sigma_n^2\mathbf{I}$ as in $\mathcal{GP}_y$ (compare Eq. (2) and Eq. (3)), but instead its heteroscedastic extension

$$\mathbf{R} = \text{diag}\left(\sigma_*^2(\mathbf{v}_1), \ldots, \sigma_*^2(\mathbf{v}_n)\right) \ , \tag{5}$$

see [3]. This matrix does not depend on a *global* noise parameter σ_n, but rather on the individual confidence estimates $\sigma_*^2(\mathbf{v}_i)$, with which $\mathcal{GP}_y$ estimated the corresponding range value. Note that this "trick" of gating out training points by artificially increasing their associated variance was also applied in recent work on modeling gas distributions [29] for deriving a GP mixture model. A more detailed discussion of the approach can be found there and in [30].

4.3. Summary of Our Approach

The full approach that also considers the angular dependencies in a range scan is denoted by the postfix *+GBP* in the experimental evaluation. To obtain the prediction of a full range scan given one omnidirectional image, we proceed as follows:

1. Warp the omnidirectional image into a panoramic view.

2. Extract for every pixel column i a vector of visual features $\mathbf{v}_i$.
3. Use $\mathcal{GP}_y$ to make independent range predictions about y_i.
4. Learn a heteroscedastic GBP $\mathcal{GP}_r$ for the set of predicted ranges $\{y_i\}_{i=1}^n$ indexed by their bearing angles α_i and make the final range predictions r_i for the same bearing angles.

As the following experimental evaluation revealed, this additional GBP treatment (post-processing with $\mathcal{GP}_r$) further increases the accuracy of range predictions. The gain, however, is rather small compared to what the GP treatment $\mathcal{GP}_y$ adds to the accuracy achievable with the baseline feature mappings. This might be due to the fact that the extracted features—and the constellation of several feature types even more so—carry information of neighboring pixel strips, such that angular dependencies are incorporated at this early stage already.

5. Experimental Evaluation

The system for predicting range from single, omnidirectional images described in the previous sections was implemented in `C/C++` and `Python` and tested on two benchmark data sets for image-based localization. The data sets, named Freiburg and Saarbrücken have been acquired in the context of the EU project CoSy. They have been made publicly available at [31] under the names COLD-Freiburg and COLD-Saarbruecken. The data was recorded using a mobile robot equipped with a laser scanner, an omnidirectional camera, and Odometry sensors at the AIS lab of the University of Freiburg and at the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken. The two environments have quite different characteristics—especially in the visual aspects. While the environment in Saarbrücken mainly consists of solid, regular structures and a homogeneously colored floor, the lab in Freiburg exhibits many glass panes, an irregular, wooden floor and challenging lighting conditions.

The goal of the experimental evaluation was to verify that the proposed system is able to make sensible range predictions from single omnidirectional camera images and to quantify the benefits of the GP approach in comparison to conceptually simpler approaches. We document a series of different experiments: First, we evaluate the accuracy of the estimated range scans using (a) the individual edge features directly, (b) the *PCA-GP*, (c) the *LDA-GP*, and (d) the *Feature-GP*, which constitutes our regression model with the four edge-based vision features as input dimensions. Then, we illustrate how these estimates can be used to build grid maps of the environment. We also evaluated whether applying the GBP model,

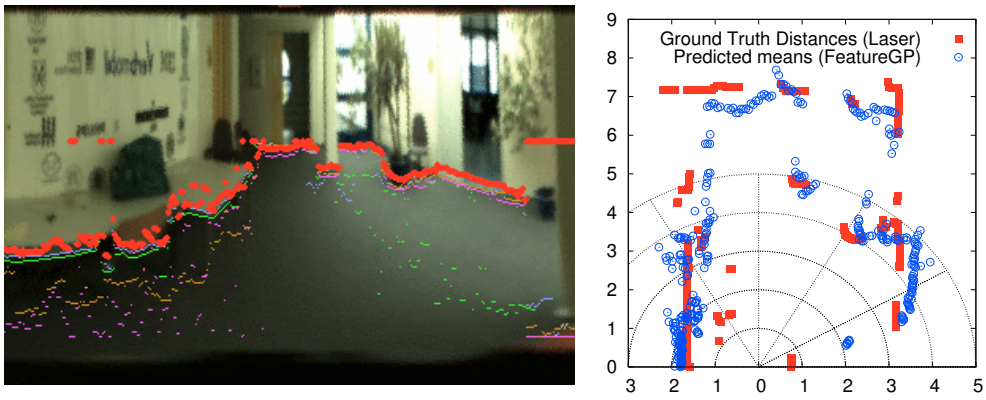


Figure 8: Left: Estimated ranges projected back onto the camera image using the feature detectors directly (small dots) and using the *Feature-GP* model (red points). Right: Prediction results and the true laser scan at one of the test locations visualized from a birds-eye view.

which was introduced in [3], as a post-processing step to the predicted range scans can further increase the prediction accuracy. The GBP model places a Gaussian process prior on the range function (rather than on the function that maps features to distances) and, thus, also models angular dependencies. We denote these models by *Feature-GP+GBP*, *PCA-GP+GBP*, and *LDA-GP+GBP*.

5.1. Quantitative Results

Table 1 summarizes the average RMSE (root mean squared error) obtained for different system configurations, which are detailed in the following. The error is measured as the difference between *measured* laser ranges and ranges *predicted* using the visual input. The first four configurations, referred to as C01 to C04, apply the optimized mapping functions for the different edge features (see Figure 6). Depending on the data, the features provide estimates with an RMSE of between 1.7 m and 3 m. We then evaluated the configurations C05 and C06 which use the four edge-based features as inputs to a Gaussian process model as described in Section 4 to learn the mapping from the feature vectors to the distances. The learning algorithm was able to perform range estimation with an RMSE of around 1 m. Note that we measure the prediction error relative to the recorded laser beams rather than to the true geometry of the environment. Thus, we report a conservative error estimate that also includes mismatches due to reflected laser beams or due to imperfect calibration of the individual components. To give a visual impression of the prediction accuracy of the *Feature-GP*, we give a typical laser scan and the mean predictions in the right diagram in Figure 8.

Table 1: Average errors obtained with the different methods. The root mean squared errors (RMSE) are calculated relative to the mean predictions for the complete test sets.

Configuration	RMSE on test set	
	Saarbrücken	**Freiburg**
C01: Laws5	1.70m	2.87m
C02: Laws5+LMD	2.01m	2.08m
C03: Laws3+Canny	1.74m	2.87m
C04: Laws3+Canny+LMD	2.06m	2.59m
C07: PCA-GP	1.24m	1.40m
C09: LDA-GP	1.20m	1.31m
C05: Feature-GP	1.04m	1.04m
C08: PCA-GP+GBP	1.22m	1.41m
C10: LDA-GP+GBP	1.17m	1.29m
C06: Feature-GP+GBP	1.03m	0.94m

The *PCA-GP* approach (denoted as C07) that does not require engineered features, but rather works on the low-dimensional representation of the raw visual input computed using the PCA. The resulting six-dimensional feature vector is used as input to the Gaussian process model. With an RMSE of 1.2 m to 1.4 m, the *PCA-GP* outperforms all four engineered features, but is not as accurate as the *Feature-GP*. When using LDA for dimensionality reduction (C09) instead of PCA, we observe a reduction of the prediction error by around 4-8 per cent. Also the LDA is outperformed by *Feature-GP* in terms of prediction accuracy. It should be stressed, however, that *PCA-GP* as well as *LDA-GP* do not require any manually defined features as they operate on the observed 420-dimensional pixel columns directly. For configurations C06, C08, and C10, we predicted the ranges per scan using the same methods as above, but additionally applying the GBP model [3] to incorporate angular dependencies between the predicted beams. This post-processing step yields slight improvements compared to the original variants C05, C07, and C09.

The left image in Figure 8 depicts the predictions based on the individual vision features and the *Feature-GP*. It can be clearly seen from the image, that the different edge-based features model different parts of the range scan well. The *Feature-GP* fuses these unreliable estimates to achieve high accuracy on the whole scan. The result of the *Feature-GP+GBP* variant for the same situation is given in Figure 1. The right diagram in Figure 8 visualizes a typical prediction result and

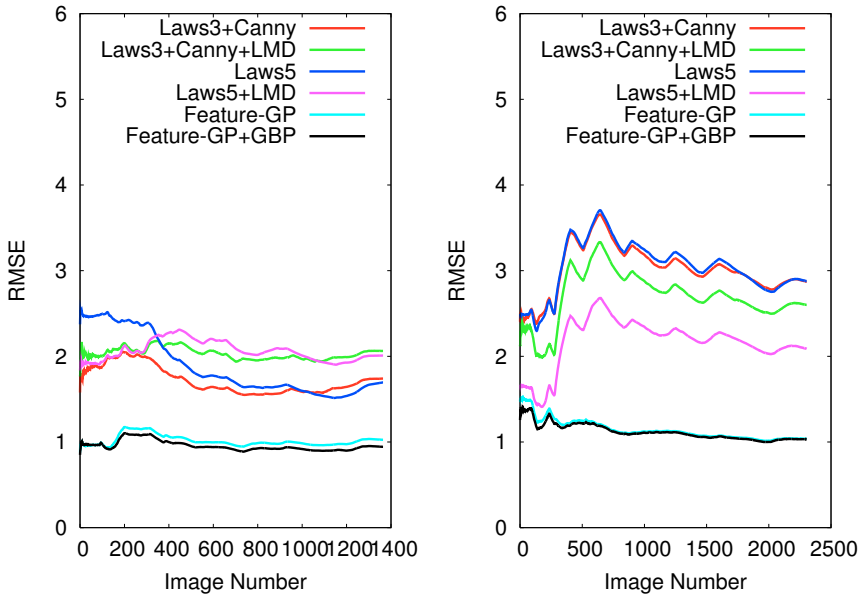


Figure 9: The evolution of the root mean squared error (RSME) for the individual images of the Saarbrücken (left) and Freiburg (right) data sets.

the corresponding laser scan—which can be regarded here as the ground truth—from a birds-eye view. The evolution of the RMSE for the different methods over time is given in Figure 9. As can be seen from the diagrams, the prediction using the *Feature-GP* model outperforms the other techniques and achieves a near-constant error rate.

In summary, our GP-based technique outperforms the individual, engineered features for range prediction. The smoothed approach (C06) yields the best predictions with an RMSE of around 1 m. One can obtain good results by a combination of LDA for dimensionality reduction and GP learning with an error that is only slightly larger (C10 versus C06), even though this unsupervised method does not have access to background information.

5.2. Error Analysis

In this section, we analyze the prediction accuracy of our proposed method beyond the RMSE measure, that is, considering the entire distribution of prediction error in order to identify and document its different causes. The left diagram in Fig. 10 shows the histogram of prediction errors for a typical scan. It is clearly visible, that the reported RMSE values are strongly influenced by the heavy tails

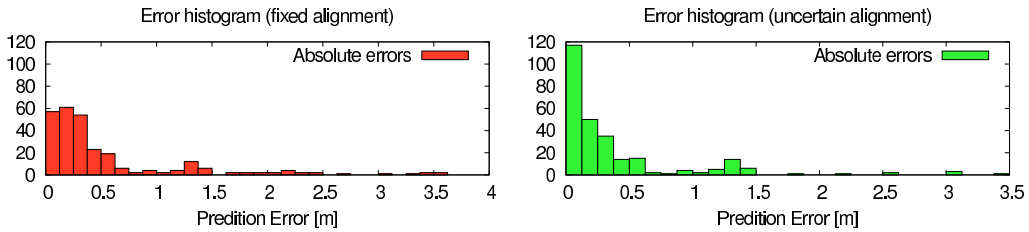


Figure 10: Error histograms for omnidirectional range prediction from single images. Left: Independent predictions for fixed beam orientations. Right: Accounting for uncertain beam orientations due to small angular miscalibration of the test and training setup.

of the error distribution. The large majority of the predictions is accurate (less than 30cm error), while very few predictions have a high error of up to 3m. Close inspection of the results reveals, that such isolated high errors are mostly caused by a small angular misalignment between the camera and the laser scanner which recorded the reference test set. This effect is visualized in the right diagram in Fig. 11. The diagram shows the "true distances" as measured by the laser scanner, the predicted distances and the respective absolute errors. It can be seen that most of the absolute prediction error accounts to beam 18520 (ID in the entire test set) which is located close to a depth discontinuity. Already a very small angular misalignment between laser scanner and camera can lead to such peaks in the error function.

As a result, the reported RMSE values have to be seen as a tool for comparing different approaches and settings rather than as a measure of precision for an actual application. From our experience, error histograms and concrete prediction examples deliver the best picture of the actual precision to be expected.

To show the influence of angular misalignment as well as long-range predictions quantitatively, we give a comparison of different error measures in Tab. 2. In the row labeled "fixed aligment", we give the errors for direct comparisons

Table 2: Comparison of error measures.

	Root Mean Squared Error (RMSE)	Mean Absolute Error	Mean relative Error
Fixed alignment	0.88 m	0.55 m	0.19
Uncertain alignment	0.67 m	0.38 m	0.14

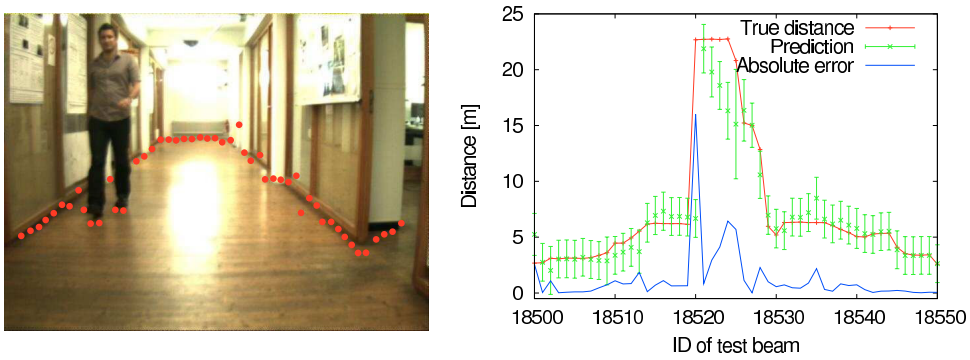


Figure 11: Left: Range predictions of method C05 (Feature-GP) for a single perspective camera image taken from a test sequence. Right: Visualization of the prediction errors for a typical scan from a test sequence. Most of the absolute prediction error is caused by small angular misalignments (here: beam 18520) and for long-range predictions (exceeding 10m).

between laser beams and prediction w.r.t. a fixed beam orientation. For "uncertain aligment", we allow for a small angular misalignment of the laser beams and their projections to the camera images. The relative errors in the last column are computed by dividing the range predictions by the true distances.

As a reference for comparison, Saxena *et al.* [2] reports depth reconstruction errors in indoor environments of 0.084 on a log scale (base 10), which corresponds to 1.21m of mean absolute error. Including stereo information using a second camera, their error drops to 0.079 in log scale, that is, 1.19m on a regular scale.

5.3. Using Perspective Cameras

To show the flexibility of our method, we conducted additional experiments using a single perspective camera (as opposed to an omnidirectional one). In this setting, the correspondence between range observations from the laser scanner (available only during training) and the camera image is not as direct as for omnidirectional, axis-aligned cameras. Nevertheless, the mapping function is bijective in the region observed by the camera and it can be computed analytically using projective geometry. The right image in Fig. 2 shows an example image and the laser beams transformed into image coordinates. The camera has a 50° field of view and it covers approximately 2.2m to 30m in depth. The camera was tilted downwards by 10°. We recorded two data sets containing 600 images each. One set was used for training, the other one for testing.

In this experiment, we additionally evaluated a combination of PCA and LDA, for which we first reduced the dimensionality of the data to 50 using PCA and

then applied LDA to further reduce to 6 dimensions. This is a common approach in face recognition, addressing the concern that LDA might perform poorly due to too little training data. PCA and LDA were learned from 100 images randomly drawn from the training set and the GP models used 300 random images. Test statistics were computed over the whole test set (50 beams per image).

Table 3 gives the quantitative results for this experiment. The observed trend is similar to the one described in Sec. 5.1: The feature-based GP performs best, while the unsupervised methods (LDA and PCA) follow up with a higher mean prediction error. The combination of LDA and PCA did not yield significant advantages in this setting. The absolute prediction errors are higher compared to the omnidirectional setting, mainly because the respective test set contains significantly more long-range predictions due to the heading of the camera towards the long corridor.

Referring to the discussion in the previous section, it should be noted that the distribution of errors is strongly biased towards errors caused by small angular misalignments (see the right diagram in Fig. 11). For most practical applications, for example obstacle avoidance, such small angular misalignments do not have a negative impact. The left image in Fig. 11 shows a typical image from the test sequence including the predictions made using C05 (Feature-GP).

Table 3: Average errors obtained with the different methods on single perspective camera images. The root mean squared errors (RMSE) are calculated relative to the mean predictions for the complete test sets.

	Prediction errors (on single images of a perspective camera)	
Configuration	**Mean absolute error**	**RMSE**
C09: LDA-GP	2.24m	3.52m
C07: PCA-GP	1.87m	3.10m
C11: PCA-LDA-GP	1.85m	3.02m
C05: Feature-GP	1.19m	2.65m

5.4. Non-Linear Dimensionality Reduction

In addition to PCA and LDA, we also considered applying non-linear dimensionality reduction as a preprocessing step. Non-linear dimensionality reduction can be described as seeking a low-dimensional manifold (not necessarily a linear subspace) in which the observed data points can be represented well. Approaches to this problem include local linear embedding (LLE) [32] and ISOMAP [33].

We implemented LLE, due to our positive experience with this technique in the past. In this domain, however, LLE performed significantly worse than all other techniques evaluated in Table 1. An analysis of the constructed manifolds indicated that the low performance may be caused by the significant number of outliers present in our real-world data sets. Similar observations about LLE and the presence of outliers have also been reported by other researcher. Chang and Yeung [34], for example, report that adding between 5% and 10% outliers to perfect data can prevent LLE from finding an appropriate embedding.

5.5. Learning Occupancy Maps from Predicted Scans

Our approach can be applied to a variety of robotics tasks such as obstacle avoidance, localization, or mapping. To illustrate this, we show how to learn a grid map of the environment from the predictive range distributions. Compared to occupancy grid mapping where one estimates for each cell the probability of being occupied or free, we use the so-called *reflection probability maps*. A cell of such a map models the probability that a laser beam passing this cell is reflected or not. Reflection probability maps, which are learned using the so-called *counting model*, have the advantage of requiring no hand-tuned sensor model such as occupancy grid maps (see [35] for further details). The reflection probability m_i of a cell i is given by

$$m_i \;=\; \frac{\alpha_i}{\alpha_i + \beta_i}\,, \tag{6}$$

where α_i is the number of times an observation hits the cell, i.e., ends in it, and β_i is the number of misses, i.e., the number of times a beam has intercepted a cell without ending in it. Since our GP approach does not estimate a single laser end point, but rather a full (normal) distribution $p(z)$ of possible end points, we have to integrate over this distribution (see Figure 12). More precisely, for each grid cell c_i, we update the cell's reflectance values according to the predictive distribution $p(z)$ according to the following formulas:

$$\alpha_i \;\leftarrow\; \alpha_i + \int_{z \in c_i} p(z)\, dz \tag{7}$$

$$\beta_j \;\leftarrow\; \beta_i + \int_{z > c_i} p(z)\, dz\,. \tag{8}$$

Note that for perfectly accurate predictions, the extended update rule is equivalent to the standard formula stated above.

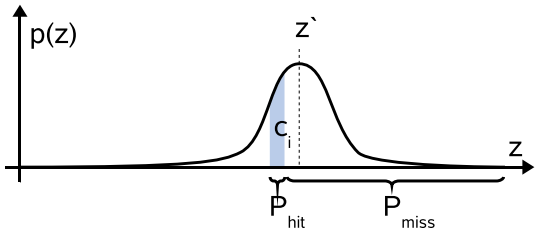


Figure 12: The counting model for reflectance grid maps in conjunction with sensor models that yield Gaussian predictive distributions over ranges.

We applied this extended reflection probability mapping approach to the trajectories and range predictions that resulted from the experiments reported above. Figure 13 presents the laser-based maps using a standard reflection probability mapping system (left column) and our extended variant using the predicted ranges (right column) for the two environments (Freiburg on top and Saarbrücken below). In both cases, it is possible to build an accurate map, which is comparable to maps obtained with infrared proximity sensors [36] or sonars [21].

6. Conclusion

This paper presents a new approach to estimating the free space around a robot based on single images recorded with an omnidirectional camera. The task of estimating the range to the closest obstacle is achieved by applying a Gaussian process model for regression, utilizing edge-based features extracted from the image or, alternatively, using PCA or LDA to find a low-dimensional representation of the visual input in an unsupervised manner. All learned models outperform the optimized individual features.

We furthermore showed in experiments with a real robot that the range predictions are accurate enough to feed them into a mapping algorithm considering predictive range distributions and that the resulting maps are comparable to maps obtained with infrared or sonar sensors.

Acknowledgments

We would like to thank Andrzej Pronobis and Jie Luo for creating the CoSy data sets. This work has partly been supported by the EC under contract number FP7-231888-EUROPA and FP6-004250-CoSy, by the DFG under contract number SFB/TR-8, and by the German Ministry for Education and Research (BMBF) through the DESIRE project.

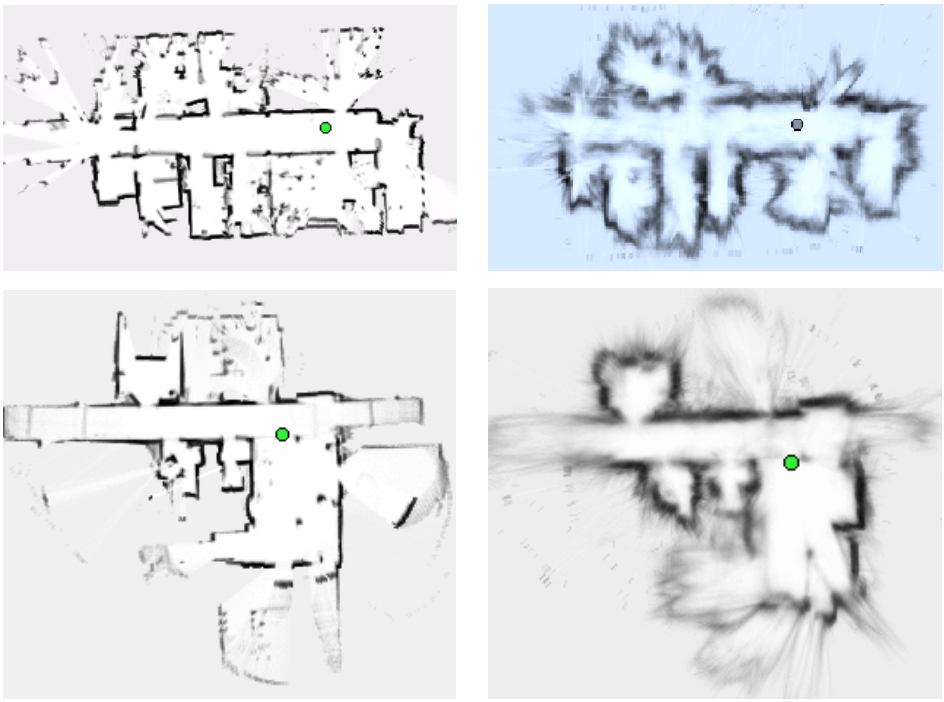


Figure 13: Maps of the Freiburg AIS lab (top row) and DFKI Saarbrücken (bottom row) using real laser data (left) and the predictions of the *Feature-GP* (right).

References

[1] G. Swaminathan, S. Grossberg, Laminar cortical mechanisms for the perception of slanted and curved 3-D surfaces and their 2-D pictorical projections, Journal of Vision 2 (7) (2002) 79–79.

[2] A. Saxena, S. Chung, A. Ng., 3-d depth reconstruction from a single still image, Int. Journal of Computer Vision (IJCV).

[3] C. Plagemann, K. Kersting, P. Pfaff, W. Burgard, Gaussian beam processes: A nonparametric bayesian measurement model for range finders, in: Proc. of Robotics: Science and Systems (RSS), 2007.

[4] F. Sinz, J. Quinonero-Candela, G. Bakir, C. Rasmussen, M. Franz, Learning depth from stereo, in: 26th DAGM Symposium, 2004.

[5] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[6] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, SURF: Speeded up robust features, Computer Vision and Image Understanding (CVIU) 110 (3) (2008) 346–359.

[7] A. Davision, I. Reid, N. Molton, O. Stasse, Monoslam: Real-time single camera slam, IEEE Transaction on Pattern Analysis and Machine Intelligence 29 (6).

[8] H. Strasdat, C. Stachniss, M. Bennewitz, W. Burgard, Visual bearing-only simultaneous localization and mapping with improved feature matching, 2007.

[9] B. Micusik, T. Pajdla, Structure from motion with wide circular field of view cameras, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (7) (2006) 1135–1149.

[10] R. Sim, J. J. Little, Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2006, pp. 2082–2089.

[11] P. Favaro, S. Soatto, A geometric approach to shape from defocus, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (2005) 406–417.

[12] A. Torralba, A. Oliva, Depth estimation from image structure, IEEE Transactions on Pattern Analysis and Machine Learning.

[13] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, G. Bradski, Self-supervised monocular road detection in desert terrain., in: Proc. of Robotics: Science and Systems (RSS), 2006.

[14] J. Michels, A. Saxena, A. Ng, High speed obstacle avoidance using monocular vision and reinforcement learning, in: Int. Conf. on Machine Learning (ICML), 2005, pp. 593–600.

[15] E. Menegatti, A. Pretto, A. Scarpa, E. Pagello, Omnidirectional vision scan matching for robot localization in dynamic environments, IEEE Transactions on Robotics 22 (3) (2006) 523–535.

[16] D. Hoiem, A. Efros, M. Herbert, Recovering surface layout from an image, Int. Journal of Computer Vision (IJCV) 75 (1).

[17] F. Han, S.-C. Zhu, Bayesian reconstruction of 3d shapes and scenes from a single image, in: IEEE Int. Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, Washington, DC, USA, 2003, p. 12.

[18] E. Delage, H. Lee, A. Ng., Automatic single-image 3d reconstructions of indoor manhattan world scenes., in: Proceedings of the 12th International Symposium of Robotics Research (ISRR), 2005.

[19] R. Ewerth, M. Schwalb, B. Freisleben, Using depth features to retrieve monocular video shots, in: Proceedings of the 6th ACM international conference on Image and video retrieval (CIVR), ACM, New York, NY, USA, 2007, pp. 210–217.

[20] H. Moravec, A. Elfes, High resolution maps from wide angle sonar, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), St. Louis, MO, USA, 1985, pp. 116–121.

[21] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, T. Schimdt, Map learning and high-speed navigation in RHINO, in: AI-based Mobile Robots: Case studies of successful robot systems, MIT Press, Cambridge, MA, 1998.

[22] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, T. Yoshigahara, Obstacle avoidance and path planning for humanoid robots using stereo vision, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), New Orleans, LA, USA, 2004.

[23] P. Elinas, R. Sim, J. J. Little, σSLAM: Stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution, in: Proc. of ICRA, 2006.

[24] C. Plagemann, F. Endres, J. Hess, C. Stachniss, W. Burgard, Monocular range sensing: A non-parametric learning approach, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Pasadena, CA, USA, 2008.

[25] E. Alpaydin, Introduction To Machine Learning, MIT Press, 2004.

[26] E. R. Davies, Laws texture energy in texture, in: Machine Vision: Theory, Algorithms, Practicalities, Acedemic Press, 1997.

[27] F. Canny, A computational approach to edge detection, IEEE Trans. Pattern Analysis and Machine Intelligence (1986) 679–714.

[28] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[29] C. Stachniss, C. Plagemann, A. Lilienthal, W. Burgard, Gas distribution modeling using sparse gaussian process mixture models, in: Proc. of Robotics: Science and Systems (RSS), Zurich, Switzerland, 2008.

[30] V. Tresp, Mixtures of gaussian processes, in: Proc. of the Conf. on Neural Information Processing Systems (NIPS), 2000.

[31] EU Project CoSy. [link].
URL `http://www.cognitivesystems.org/`

[32] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[33] J. Tenenbaum, V. de Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction., Science 290 (5500) (2000) 2319–2323.

[34] H. Chang, D.-Y. Yeung, Robust locally linear embedding, Pattern Recognition 39 (6) (2006) 1053–1065.

[35] W. Burgard, C. Stachniss, D. Haehnel, Autonomous Navigation in Dynamic Environments, Vol. 35 of STAR Springer tracts in advanced robotics, Springer Verlag, 2007, Ch. Mobile Robot Map Learning from Range Data in Dynamic Environments.

[36] Y. Ha, H. Kim, Environmental map building for a mobile robot using infrared range-finder sensors, Advanced Robotics 18 (4) (2004) 437–450.

[J2] H. Kretzschmar, G. Grisetti, and C. Stachniss. Life-long map learning for graph-based simultaneous localization and mapping. *KI – Kuenstliche Intelligenz*. In press.

Life-long Map Learning for Graph-based SLAM Approaches in Static Environments

Henrik Kretzschmar, Giorgio Grisetti, Cyrill Stachniss

In this paper, we consider the problem of life-long map learning with mobile robots using the graph-based formulation of the simultaneous localization and mapping problem. To allow for life-long mapping, the memory and computational requirements for mapping should remain bounded when re-traversing already mapped areas. Our method discards observations that do not provide relevant new information with respect to the model constructed so far. This decision is made based on the entropy reduction caused by observations. As a result, our approach scales with the size of the environment and not with the length of the trajectory. The experiments presented in this paper illustrate the advantages of our method using real robot data.

1 Introduction

Maps of the environment are needed for a wide range of robotic applications, including transportation tasks and many service robotic applications. Therefore, learning maps is regarded as one of the fundamental problems in mobile robotics. In the last two decades, several effective approaches for learning maps have been developed. The graph-based formulation of the simultaneous localization and mapping (SLAM) problem models the poses of the robot as nodes in a graph. Spatial constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes. Graph-based approaches such as [Frese *et al.*, 2005; Olson *et al.*, 2006; Grisetti *et al.*, 2007b], which are probably the most efficient techniques at the moment, typically marginalize out the features (or local grid maps) and reduce the mapping problem to trajectory estimation without prior map knowledge. Therefore, the underlying graph structure is often called the pose graph.

The majority of approaches, however, assumes that map learning is carried out as a preprocessing step and that the robot later on uses the model for tasks such as localization or path planning. A robot that is constantly updating the map of its environment has to address the so-called life-long SLAM problem. This problem cannot be handled well by most graph-based techniques since the complexity of these approaches grows with the length of the trajectory. As a result, the memory as well as the computational requirements grow over time and therefore these methods cannot be applied in this context.

The contribution of this paper is a novel approach that enables graph-based SLAM approaches to operate in the context of life-long map learning in static scenes. Our approach is orthogonal to the underlying graph-based mapping technique and applies an entropy-driven strategy to prune the pose graph while minimizing the loss of information. This becomes especially important when re-traversing already mapped areas. As a result, our approach scales with the size of the environment and not with the length of the trajectory. It should be noted that not only long-term mapping systems benefit from our method. Also traditional mapping systems are able to compute a map faster since less resources are claimed and less comparisons between observations are needed to solve the data association problem. We furthermore illustrate that the resulting grid maps are less blurred compared to the maps built without our approach.

2 Related Work

There is a large variety of SLAM approaches available in the robotics community. Common techniques apply extended and unscented Kalman filters [Leonard and Durrant-Whyte, 1991; Julier *et al.*, 1995], sparse extended information filters [Thrun *et al.*, 2004; Eustice *et al.*, 2005], particle filters [Montemerlo and Thrun, 2003; Grisetti *et al.*, 2007a], and graph-based, least square error minimization approaches [Lu and Milios, 1997; Frese *et al.*, 2005; Olson *et al.*, 2006; Grisetti *et al.*, 2007b].

The group of graph-based SLAM approaches for estimating maximum-likelihood maps is often regarded as the most effective means to reduce the error in the pose graph. Lu and Milios [1997] were the first to refine a map by globally optimizing the system of equations to reduce the error introduced by constraints. Since then, a large variety of approaches for minimizing the error in the constraint network have been proposed. Duckett *et al.* [2002] use Gauss-Seidel relaxation. The multi-level relaxation (MLR) approach of Frese *et al.* [2005] apply relaxation at different spatial resolutions. Given a good initial guess, it yields very accurate maps particularly in flat environments. Folkesson and Christensen [2004] define an energy function for each node and try to minimize it. Thrun and Montemerlo [2006] apply variable elimination techniques to reduce the dimensionality of the optimization problem. Olson *et al.* [2006] presented a fast and accurate optimization approach which is based on the stochastic gradient descent (SGD). Compared to approaches such as MLR, it still converges from a worse initial guess. Based on Olson's optimization algorithm, Grisetti *et al.* [2007b] proposed a different parameterization of the nodes in the graph. The tree-based parameterization yields a significant boost in performance. In addition to that, the approach can deal with arbitrary graph topologies. The approach presented in this paper is built upon the work of Grisetti *et al.* [2007b].

Most graph-based approaches available today do not provide means to efficiently prune the pose graph, that has to be corrected by the underlying optimization framework. Most approaches can only add new nodes or apply a rather simple decision whether to add a new node to the pose graph or not (such as the question of how spatially close a node is to an existing one). However, there are some notable exceptions: Folkesson and Christensen [2004] combine nodes into so-called star nodes which then define rigid local submaps. The method applies de-

layed linearization of local subsets of the graph, permanently combining a set of nodes in a relative frame. Related to that, Konolige and Agrawal [2008] subsample nodes for the global optimization and correct the other nodes locally after global optimization.

Other authors considered the problem of updating a map upon changes in the environment. For example, Biber and Duckett [2005] propose an approach to update an existing model of the environment. They use five maps on different time scales and incorporate new information by forgetting old information. Related to that, Stachniss and Burgard [2005] learn clusters of local map models to identify typical states of the environment. Both approaches focus on modeling changes in the environment but do not address the full SLAM problem since they require an initial map to operate. The approach presented in this paper further improves the node reduction techniques for graph-based SLAM by estimating how much the new observation will change the map. It explicitely considers the information gain of observations to decide whether the corresponding nodes should be removed from the graph or not.

In the remainder of this paper, we will first introduce the basic concept of pose graph optimization originally presented in [Grisetti *et al.*, 2007b]. In Section 4, we describe our contribution to information-driven node reduction. In Section 5, we finally presents our experimental evaluation.

3 Map Learning using Pose Graphs

The graph-based formulation of the SLAM problem models the poses of the robot as nodes in a graph (a so-called pose graph). Spatial constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes. Most approaches to graph-based SLAM focus on estimating the most-likely configuration of the nodes and are therefore referred to as maximum-likelihood (ML) techniques. The approach presented in this paper also applies an optimization framework that belongs to this class of methods.

3.1 Problem Formulation

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. Let $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ be a vector of parameters which describes a configuration of the nodes. Let δ_{ji} and Ω_{ji} be respectively the mean and the information matrix of an observation of node j seen from node i. Let $f_{ji}(\mathbf{x})$ be a function that computes a zero noise observation according to the current configuration of the nodes j and i.

Given a constraint between node j and node i, we can define the *error* e_{ji} introduced by the constraint as

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

as well as the residual $r_{ji} = -e_{ji}(\mathbf{x})$. Let $\mathcal{C}$ be the set of pairs of indices for which a constraint δ exists. The goal of a ML approach is to find the configuration of the nodes that minimizes the negative log likelihood of the observations. Assuming the constraints to be independent, this can be written as

$$\mathbf{x}^* \quad = \quad \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{2}$$

3.2 Map Optimization

To solve Eq. (2), different techniques can be applied. Our work applies the approach of Grisetti *et al.* [2007b] which is an extention of the work of Olson *et al.* [2006]. Olson *et al.* [2006] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to compute the most likely configuration of the nodes in the network. The approach minimizes Eq. (2) by iteratively selecting a constraint and by moving the nodes of the pose graph in order to decrease the error introduced by the selected constraint. Compared to the standard formulation of gradient descent, the constraints are not optimized as a whole but individually. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji} \tag{3}$$

Here, $\mathbf{x}$ is the set of variables describing the locations of the poses in the network and $\mathbf{H}^{-1}$ is a preconditioning matrix. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, r_{ji} is the residual, and λ is the learning rate which decreases with the iteration. For a detailed explanation of Eq. (3), we refer the reader to [Grisetti *et al.*, 2007b] or [Olson *et al.*, 2006].

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing subsets of nodes, one subset for each constraint. Whenever a solution for one of these subproblems is found, the network is updated accordingly. Obviously, updating the constraints one after each other can have antagonistic effects on the corresponding subsets of variables. To avoid infinite oscillations, one uses the learning rate λ to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different subproblems converge asymptotically to an equilibrium point which is the solution reported by the algorithm.

3.3 Tree Parameterization for Efficient Map Optimization

The poses $\mathbf{p} = \{p_1, \ldots, p_n\}$ of the nodes define the configuration of the graph. The poses can be described by a vector of *parameters* $\mathbf{x}$ such that a bidirectional mapping between $\mathbf{p}$ and $\mathbf{x}$ exists. The parameterization defines the subset of variables that are modified when updating a constraint in SGD. An efficient way of parameterizing the nodes is to use a tree. To obtain that tree, we compute a spanning tree from the pose graph. Given such a tree, one can define the parameterization for a node as

$$x_i \quad = \quad p_i - p_{\mathrm{parent}(i)}, \tag{4}$$

where $p_{\mathrm{parent}(i)}$ refers to the parent of node i in the spanning tree. As shown in [Grisetti *et al.*, 2007b] this approach can dramatically speed up the convergence rate compared to the method of Olson *et al.* [2006].

The technique described so far is typically executed as a batch process but there exists also an incremental variant [Grisetti *et al.*, 2008] which performs the optimization only on the portions of the graph which are affected by the introduction of new constraints. It is thus able to re-use the previously generated solutions to compute the new one.

3.4 The SLAM Front-end

The approach briefly described above only focuses on correcting a pose graph *given* all constraints and is often referred to as the SLAM back-end. In contrast to that, the SLAM front-end aims at extracting the constraints from sensor data. In this paper, we build our work upon the SLAM front-end described in the Ph.D. thesis of Olson [2008]. We refer to this technique as "without graph reduction". The front-end generates a new node every time the robot travels a minimum distance. Every node is labeled with the laser-scan acquired at that position. Constraints between subsequent nodes are generated by pairwise scan-matching. Every time a new node is added, the front-end seeks for loop closures with other nodes. It therefore computes the marginal covariances of all nodes with respect to the newly added one. This is done by using the approximate approach of Tipaldi *et al.* [2007]. Once these covariances are computed, the approach selects a set of candidate loop closures using the χ^2 test. The corresponding edges are then created by aligning the current observations with the ones stored in each candidate loop closing node determined by the previous step. In addition to that, an outlier rejection technique based on spectral clustering is applied to reduce the risk of wrong matches.

The contribution of this paper is a technique that "sits" between the SLAM front-end and the optimizer, the SLAM back-end, in order to enable a robot to perform life-long map learning in static worlds. It allows for removing redundant nodes by considering the information gain of the observations. As we will show in the remainder of this paper, our method allows for efficient map learning especially in the context of frequent re-traversals of previously mapped areas. In addition to that, we illustrate that this technique improves the map quality when learning grid maps and that it generates sharp boundaries between free and occupied spaces.

4 Our Approach for Life-Long Map Learning

For life-long map learning, a robot cannot add new nodes to the graph whenever it is re-entering already visited terrain. The key idea of our approach is to prune the graph structure to limit the number of nodes. Most existing approaches to graph reduction simply consider the position of a potential new node and do not integrate it into the graph if it is spatially close to an existing node. In this paper, we propose a different, information-driven approach to node reduction. In contrast to considering poses only, we measure the amount of information an observation contributes to the belief of the robot.

4.1 Information-Theoretic Node Reduction

The key idea of our approach is to measure the information gain which is defined as the reduction of uncertainty caused by an observation. To measure the uncertainty of the current belief of the robot, we use the entropy. Entropy H is a general measure for uncertainty in the belief over a random variable x. For a discrete probability distribution, entropy is given by

$$H\left(p(x)\right) = -\sum_{x} p(x) \log_2 p(x). \tag{5}$$

In theory, the entropy has to be computed over the full SLAM posterior, thus taking into account the pose and the map uncertainty. In our case, however, we seek for discarding observations when re-traversing a known area. If the robot performs such a re-traversal, it has already identified loop-closing constraints that caused a pose correction carried out by the optimization framework. Thus, the nodes which are discarded typically have a minor effect on the pose uncertainty itself. Therefore, the relevant part of the uncertainty is given by the map uncertainty. To put this in other words, the optimization framework is a maximum likelihood estimator and its estimate is the node arrangement. Based on this arrangement, the model can be computed and thus the entropy. For a grid map, the entropy is

$$\begin{aligned} H(p(m)) &= \sum_{c \in m} H(p(c)) && (6) \\ &= -\sum_{c \in m} \big(p(c) \log p(c) + p(\neg c) \log p(\neg c)\big), && (7) \end{aligned}$$

where c refers to the individual cells of the grid map m. The idea of our approach is to discard all observations and the corresponding nodes in the graph which do not reduce the entropy. Based on the entropy, we can define the information gain I of an observation z by summing over all cells c in the map m as

$$I(z) = \sum_{c \in m} H(p(c)) - H(p(c|z)). \tag{8}$$

Note that our approach does not perform its decision for the most recent observation only. In contrast, it also considers old observations for removal. In each step, our algorithm considers the information gain for all measurements z_i for $i = 1, \ldots, n$

$$I(z_i) = \sum_{c \in m} H(p(c \mid z_{1:i-1,i+1:n})) - H(p(c|z_{1:n})). \tag{9}$$

To determine if an observation z_i should be discarded, we check if $I(z_i)$ is smaller than or equal to zero. Zero means that the observation does not contain novel information that would reduce the uncertainty in the belief of the robot.

In practice, the sensor of the robot has only a limited range. Therefore, only spatially close nodes (an thus measurements) need to be considered here and not all scans $z_{1:n}$ in the map. We compute the information gain according to Eq. (9) for all observations in the vicinity of the robot according to a Dijkstra expansion. Then, we remove observations until the one with the lowest information gain has a value greater than zero.

In this section, we described how to identify nodes in the graph corresponding to observations that can be discarded without losing relevant information. In the next section, we describe how to prune the graph while preserving most of the information encoded in the edges.

4.2 Updating the Graph

To remove a node from the graph structure, the information encoded in the edges between the node and its neighbors should not be discarded along with the node since it encodes relevant spatial information. Simply removing all adjacent edges can even lead to several disconnected graphs which is clearly undesirable since in this case, the overall map would break apart into unconnected submaps.

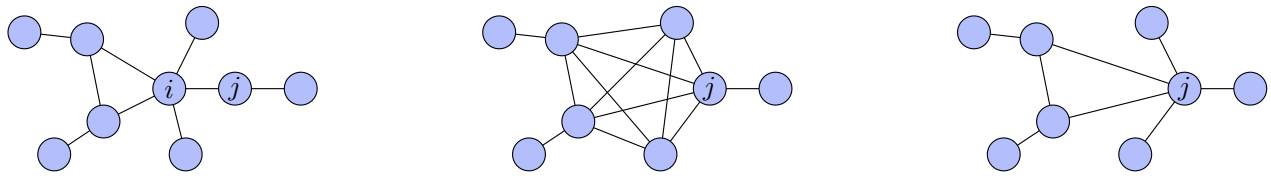


Figure 1: Left: Graph built according to the observations. Middle: Exactly marginalized but densely connected graph after removing node i. Right: Approximate solution obtained by collapsing the edges at i into node j.

In theory, one needs to connect all pairs of neighbors of the node to be removed to preserve the error function used for graph optimization. This, however, can lead to a fully connected graph which results in a quadratic number of edges. When removing a node which has k neighbors, the total number of edges can grow by up to $\frac{1}{2}k(k-1)-k$. This introduces a complexity which is not suited for life-long map learning. An illustration of the graph update is depicted in the left and middle illustration of Figure 1.

Therefore, we propose an alternative way of removing a node which is an approximate solution but which does not increase the computational cost. Let i be the node to be removed and N be the set of neighbors of i. The key idea of our method is to select one node $j \in N$ and merge the information of all edges connecting i into existing as well as new edges connecting j. This results in removing all k edges at i and in adding $(k-1)$ edges between j and $N \backslash \{j\}$. If this results in two edges connecting the same two nodes j and one of its neighbors, these edges can directly be merged (see also Figure 1, right). Consequently, the overall number of edges always decreases at least by 1.

Merging the information encoded in the edges can be done in a straight forward manner since the edges encode Gaussian constraints. Thus, concatenating two constraints with means δ_{ji} and δ_{kj} and information matrices Ω_{ji} and Ω_{kj} is done by

$$\delta_{ki} = \delta_{ji} \oplus \delta_{kj} \tag{10}$$

$$\Omega_{ki} = \left(\Omega_{ji}^{-1} + \Omega_{kj}^{-1}\right)^{-1}. \tag{11}$$

Similar to that, merging two constraints δ_{ij}^1 and δ_{ij}^2 between nodes j and i is done by

$$\delta_{ji} = \Omega_{ji}^{-1}(\Omega_{ji}^{(1)}\delta_{ji}^{(1)} + \Omega_{ji}^{(2)}\delta_{ji}^{(2)}) \tag{12}$$

$$\Omega_{ji} = \Omega_{ji}^{(1)} + \Omega_{ji}^{(2)}. \tag{13}$$

To collapse a node i into one of its neighbors, one could select, in theory, an arbitrary node $j \in N$. However, the selection of the node j can have a significant influence on the resulting map that will be obtained. The reason for that is the underlying optimization framework. Most existing approaches assume Gaussian observations (the edges represent Gaussians) although this assumption may not hold in practice. In addition to that, some optimization systems assume roughly spherical covariances to exhibit maximum performance. Thus, it is desirable to avoid long edges to limit the effect of linearization errors. Our approach therefore considers all neighbors $j \in N$ and selects the one such that the sum of the lengths of all edges in the resulting graph is minimized:

$$j^* = \operatorname*{argmin}_{j \in N} \sum_{e \in \mathcal{G}(i \to j)} \mathit{length}(e) \tag{14}$$

In Eq. (14), $\mathcal{G}(i \to j)$ refers to the graph that results when collapsing node i into node j. Note that in practice, only the edges at i and those at j need to be involved in the computation and not the entire graph.

4.3 Gamma Index

To evaluate how densely connected a pruned pose graph is in practice, we will evaluate real world data sets using the so-called gamma index [Garrison and Marble, 1965]. The gamma index (γ) is a measure of connectivity of a graph. It is defined as the ratio between the number of existing edges and the maximum number of possible edges. It is given by

$$\gamma = \frac{e}{\frac{1}{2}v(v-1)}, \tag{15}$$

where e is the number of edges and v is the number of nodes in the graph. The gamma index varies from 0 to 1, where 0 means that the nodes are not connected at all and 1 means that the graph is complete.

As we will show in the experiments, our approach leads to a small and more or less constant gamma index, below 0.01 (in all our datasets). In contrast to that, the sound pruning strategy tends towards much higher gamma values.

5 Experimental Evaluation

The experimental evaluation of this work was carried out at the University of Freiburg using a real ActivMedia Pioneer-2 robot equipped with a SICK laser range finder and running CARMEN. In addition to that, we considered a dataset that is frequently used in the robotics community to analyze SLAM algorithms, namely the Intel Research dataset provided by Dirk Hähnel.

5.1 Runtime

The experiments in this section are designed to show that our approach to informed graph pruning is well suited for robot mapping – for life-long map learning as well as for standard SLAM. In the first set of experiments, we present an analysis of how the graph structure and thus the runtime increases when a robot constantly re-traverses already mapped areas. We compare the results of a graph-based optimization approach [Grisetti *et al.*, 2007b] in combination with a re-implementation of the SLAM front-end of Olson [2008] to our novel method (using the same front- and back-end).

In the experiment presented in Figure 2, the robot was constantly re-visiting already known areas. It traveled forth and

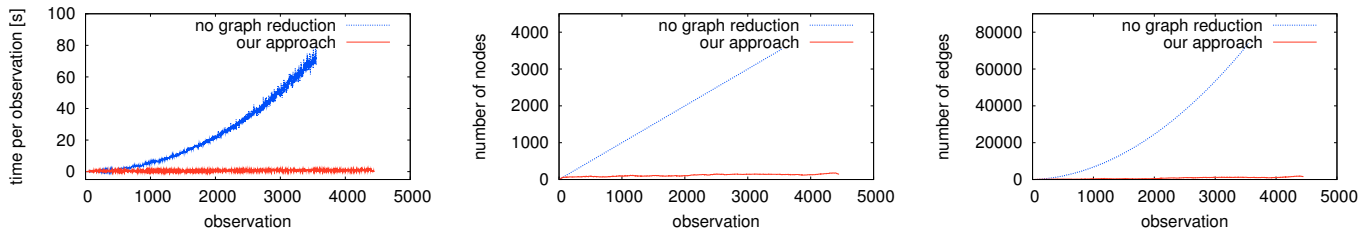


Figure 2: Typical results of an experiment in which the robot moves in already visited areas. Left: Runtime per observation for the standard approach (blue) and for our method (red). Middle and right: Number of nodes and edges in the graph for both methods. Due to time reasons, the experiment for the standard approach was aborted after around 3500 observations.

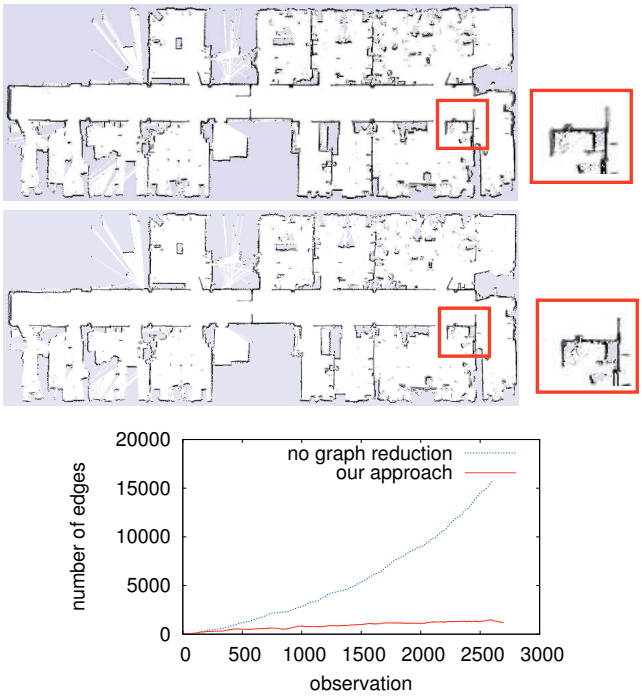


Figure 3: Results obtained by a robot moving at Freiburg University, building 079, repeatedly visiting the individual rooms and the corridor. Top map image: standard approach without graph pruning. Bottom map image: our approach.

back 100 times in an approximately 20 m long corridor in our lab environment. This behavior leads to a runtime explosion for the standard approach. Please note that this is not caused by the underlying optimization framework (which is executed in a few milliseconds) but by the SLAM front-end that looks for constraints between the nodes in the graph considering all previously recorded scans. In contrast to that, our approach keeps the number of nodes in the graph more or less constant and thus avoids the runtime explosion.

In an additional experiment, the robot repeatedly visited different rooms and the corridor in our lab. Figure 3 shows the resulting maps as well as the effect of the graph sparsification. In sum, this experiment yields similar results than in the previous experiment.

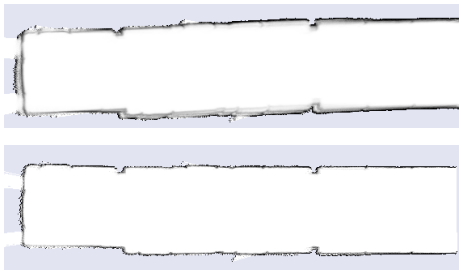


Figure 4: Robot moving 100 times forth and back in the corridor. Standard grid-based mapping approaches (top) tend to generate thick and blurred walls whereas our graph sparsification (bottom) does not suffer from this issue.

5.2 Improved Grid Map Quality by Information-driven Graph Sparsification

The second experiment is designed to illustrate that our approach has a positive influence on the quality of the resulting grid map. In contrast to feature-based approaches, grid maps have one significant disadvantage when it comes to life-long map learning. Whenever a robot re-enters a known region and uses scan-matching, the chance of making a small alignment error is nonzero. After the first error, the probability of making further errors increases since the map the robot aligns its observation with already has a (small) error. In the long run, this is likely to lead to divergence or at least to artificially thick walls and obstacles.

This effect, however, is significantly reduced when applying our graph sparsification technique since scans are only maintained as long as they provide relevant information, otherwise, they are discarded. To illustrate this effect, consider Figure 4. The top image shows the result of 100 corridor traversals without graph sparsification. The thick and blurred walls as a result of the misaligned poses are clearly visible. In contrast to that, the result obtained by our approach does not suffer from this problem (bottom image). The same effect can also be observed in the magnified areas in Figure 3.

5.3 Approximate Graph Update

We furthermore compared the effect of our approximate graph update routine versus full marginalization of the nodes (see Figure 1 for an illustration). As discussed in Section 4.2, our node

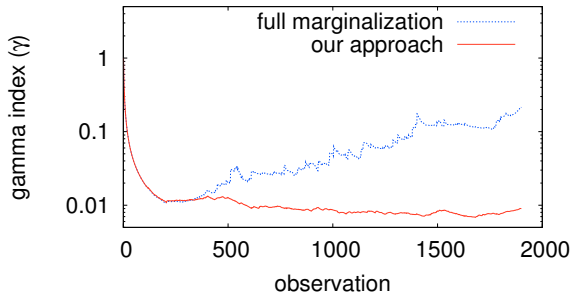


Figure 5: Evolution of the gamma index (Intel Research Lab).

removal technique is guaranteed to also decrease the number of edges in the graph. In contrast to this, full marginalization typically leads to densely connected graphs.

This effect can be observed in real world data such as the Intel Research Lab, see Figure 5. This figure shows the gamma indices of both graphs. As can be seen, the graph structure obtained with our approach is and stays comparably sparse. The corresponding graphs as well as the graph obtained by the standard approach are depicted in Figure 6.

This sparsity achieved by our approach has two advantages: First, the underlying optimization method depends linearly on the number of edges in the graph. Thus, having less edges results in a faster optimization. Second, after multiple node removals using full marginalization, it is likely that also spatially distant nodes are connected via an edge. As mentioned in Section 5.2, these long distant edges can be suboptimal for the underlying optimization engine (as this is the case for [Grisetti *et al.*, 2007b]).

6 Conclusion

In this paper, we presented a novel approach that enables life long map learning in static scenes. It is designed for mobile robots that use a graph-based framework to solve the simultaneous localization and mapping problem. By considering the information gain of observations, our method removes redundant information from the graph and in this way keeps the size of the pose-constraint network constant as long as the robot traverses already mapped areas. We introduce an approximate way to prune the graph structure that enables us to limit the complexity and allows for highly efficient robotic map learning. The approach has been implemented and thoroughly tested with real robot data. We provided real world experiments and considered standard benchmark datasets used in the SLAM community to illustrate the advantages of our methods.

Note that even though this paper describes only 2D experiments generated based on a 2D implementation of the work. The extension to 3D, however, should be straightforward. Given a local 3D grid (or more efficient representations such as octrees), the entropy and thus the information gain can be computed in the same way. Furthermore, the approximate marginalization is directly applicable to any kind of constraint network. Therefore, we believe that the approach can be directly applied to 3D data even though we have not done this so far.

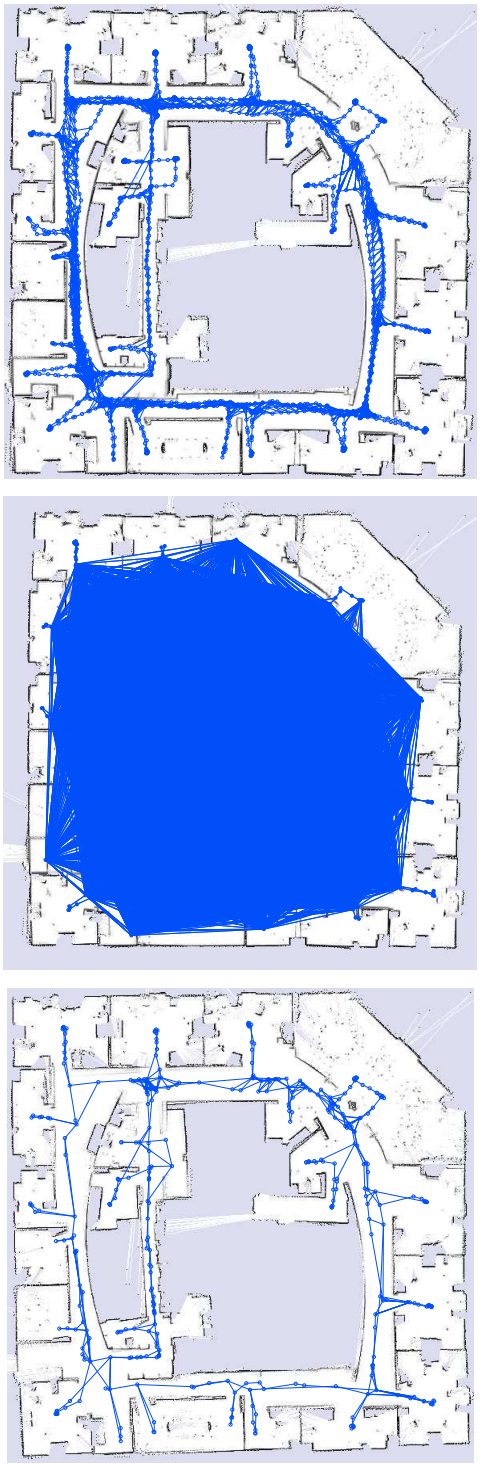


Figure 6: Map and graph obtained from the Intel Research Lab dataset by using the standard (top) as well as by full marginalization (middle) and by using our approach (bottom). Standard approach: 1802 nodes, 3916 edges, full marginalization: 349 nodes, 13052 edges, our approach with approximate marginalization: 354 nodes, 559 edges.

Acknowledgment

We would like to thank Dirk Hähnel for providing the Intel Research Lab dataset. This work has partly been supported by the DFG under contract number SFB/TR-8 and by the EC under contract number FP7-ICT-231888-EUROPA.

References

[Biber and Duckett, 2005] Biber, P., and Duckett, T. *Dynamic Maps for Long-Term Operation of Mobile Service Robots*. In *Proc. of Robotics: Science and Systems (RSS)*, pages 17–24, 2005.

[Duckett *et al.*, 2002] Duckett, T., Marsland, S., and Shapiro, J. *Fast, On-line Learning of Globally Consistent Maps*. *Autonomous Robots*, 12(3):287 – 300, 2002.

[Eustice *et al.*, 2005] Eustice, R., Singh, H., and Leonard, J. *Exactly Sparse Delayed-State Filters*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, 2005.

[Folkesson and Christensen, 2004] Folkesson, J., and Christensen, H. *Graphical SLAM - A Self-Correcting Map*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[Frese *et al.*, 2005] Frese, U., Larsson, P., and Duckett, T. *A Multilevel Relaxation Algorithm for Simultaneous Localisation and Mapping*. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[Garrison and Marble, 1965] Garrison, W. L., and Marble, D. F. *A Prolegomenon to the Forecasting of Transportation Devel.*, 1965.

[Grisetti *et al.*, 2007a] Grisetti, G., Stachniss, C., and Burgard, W. *Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters*. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[Grisetti *et al.*, 2007b] Grisetti, G., Stachniss, C., Grzonka, S., and Burgard, W. *A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent*. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[Grisetti *et al.*, 2008] Grisetti, G., Rizzini, D. L., Stachniss, C., Olson, E., and Burgard, W. *Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[Julier *et al.*, 1995] Julier, S., Uhlmann, J., and Durrant-Whyte, H. *A new Approach for Filtering Nonlinear Systems*. In *Proc. of the American Control Conference*, pages 1628–1632, 1995.

[Konolige and Agrawal, 2008] Konolige, K., and Agrawal, M. *FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping*. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[Leonard and Durrant-Whyte, 1991] Leonard, J., and Durrant-Whyte, H. *Mobile robot localization by tracking geometric beacons*. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[Lu and Milios, 1997] Lu, F., and Milios, E. *Globally Consistent Range Scan Alignment for Environment Mapping*. *Autonomous Robots*, 4:333–349, 1997.

[Montemerlo and Thrun, 2003] Montemerlo, M., and Thrun, S. *Simultaneous Localization and Mapping with Unknown Data Association using FastSLAM*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1985–1991, 2003.

[Olson *et al.*, 2006] Olson, E., Leonard, J., and Teller, S. *Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[Olson, 2008] Olson, E. *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, USA, 2008.

[Stachniss and Burgard, 2005] Stachniss, C., and Burgard, W. *Mobile Robot Mapping and Localization in Non-Static Environments*. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1324–1329, 2005.

[Thrun and Montemerlo, 2006] Thrun, S., and Montemerlo, M. *The graph SLAM algorithm with applications to large-scale mapping of urban structures*. *The International Journal of Robotics Research*, 25(5-6):403, 2006.

[Thrun *et al.*, 2004] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H. *Simultaneous Localization and Mapping With Sparse Extended Information Filters*. *Int. Journal of Robotics Research*, 23(7/8):693–716, 2004.

[Tipaldi *et al.*, 2007] Tipaldi, G. D., Grisetti, G., and Burgard, W. *Approximated Covariance Estimation in Graphical Approaches to SLAM*. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

Contact

Cyrill Stachniss
Email: stachnis@informatik.uni-freiburg.de

Bild

Henrik Kretzschmar is a PhD student at the Department of Computer Science at Freiburg University. He recently received his MSc degree in computer science from the University of Freiburg in 2009. His research interests lie in the area of simultaneous localization and mapping and probabilistic inference.

Bild

Giorgio Grisetti is working as a Post-doc at the Autonomous Intelligent Systems Lab at Freiburg University headed by Wolfram Burgard. Before, he was a PhD student at University of Rome La Sapienza were he received his PhD degree. His research interests focus providing effective solutions to mobile robot navigation in all its aspects: SLAM, localization and path planning.

Bild

Cyrill Stachniss received his PhD degree from the University of Freiburg in 2006. He then was with the Swiss Federal Institute of Technology in Zurich as a senior researcher. Since 2007, he is an academic advisor at the University of Freiburg. His research areas are mobile robot navigation, exploration, SLAM as well as learning approaches in the context of robotics. He is an associate editor of the IEEE Transactions on Robotics.

[J3] R. Kuemmerle, B. Steder, M. Ruhnke, G. Grisetti, C. Stachniss, C. Dornhege, and A. Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27:387-407, 2009.

Autonomous Robots manuscript No.
(will be inserted by the editor)

On Measuring the Accuracy of SLAM Algorithms

Rainer Kümmerle · **Bastian Steder** ·
Christian Dornhege · **Michael Ruhnke** ·
Giorgio Grisetti · **Cyrill Stachniss** ·
Alexander Kleiner

Received: date / Accepted: date

Abstract In this paper, we address the problem of creating an objective benchmark for evaluating SLAM approaches. We propose a framework for analyzing the results of a SLAM approach based on a metric for measuring the error of the corrected trajectory. This metric uses only relative relations between poses and does not rely on a global reference frame. This overcomes serious shortcomings of approaches using a global reference frame to compute the error. Our method furthermore allows us to compare SLAM approaches that use different estimation techniques or different sensor modalities since all computations are made based on the corrected trajectory of the robot.

We provide sets of relative relations needed to compute our metric for an extensive set of datasets frequently used in the robotics community. The relations have been obtained by manually matching laser-range observations to avoid the errors caused by matching algorithms. Our benchmark framework allows the user to easily analyze and objectively compare different SLAM approaches.

Keywords SLAM; mapping accuracy; benchmarking

All authors are with the
University of Freiburg, Dept. of Computer Science, Georges Köhler Allee 79, 79110 Freiburg, Germany
Tel.: +49-761-203-8006, Fax: +49-761-203-8007
E-mail: {kuemmerl,steder,dornhege,ruhnke,grisetti,stachnis,kleiner}@informatik.uni-freiburg.de

This is a preprint of an article published in Journal of Autonomous Robots. The original publication is available at www.springerlink.com

1 Introduction

Models of the environment are needed for a wide range of robotic applications including transportation tasks, guidance, and search and rescue. Learning maps has therefore been a major research focus in the robotics community in the last decades. Robots that are able to acquire an accurate model of their environment are regarded as fulfilling a major precondition of truly autonomous agents.

In the literature, the mobile robot mapping problem under pose uncertainty is often referred to as the *simultaneous localization and mapping* (SLAM) or *concurrent mapping and localization* (CML) problem [Smith and Cheeseman, 1986; Dissanayake *et al.*, 2000; Gutmann and Konolige, 1999; Hähnel *et al.*, 2003; Montemerlo *et al.*, 2003; Thrun, 2001; Leonard and Durrant-Whyte, 1991]. SLAM is considered to be a complex problem because to localize itself a robot needs a consistent map and for acquiring the map the robot requires a good estimate of its location. This mutual dependency among the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space.

Whereas dozens of different techniques to tackle the SLAM problem have been presented, there is no gold standard for comparing the results of different SLAM algorithms. In the community of feature-based estimation techniques, researchers often measure the distance or Mahalanobis distance between the estimated landmark location and the true location (if this information is available). As we will illustrate in this paper, comparing results based on an absolute reference frame can have shortcomings. In the area of grid-based estimation techniques, people often use visual inspection to compare maps or overlays with blueprints of buildings. This kind of evaluation becomes more and more difficult as new SLAM approaches show increasing capabilities and thus large scale environments are needed for evaluation. In the community, there is a strong need for methods allowing meaningful comparisons of different approaches. Ideally, such a method is capable of performing comparisons between mapping systems that apply different estimation techniques and operate on different sensing modalities. We argue that meaningful comparisons between different SLAM approaches require a common performance measure (metric). This metric should enable the user to compare the outcome of different mapping approaches when applying them on the same dataset.

In this paper, we propose a novel technique for comparing the output of SLAM algorithms. We aim to establish a benchmark that allows for objectively measuring the performance of a mapping system. We propose a metric that operates only on relative geometric relations between poses along the trajectory of the robot. Our approach allows for making comparisons even if a perfect ground truth information is not available. This enables us to present benchmarks based on frequently used datasets in the robotics community such as the MIT Killian Court or the Intel Research Lab dataset. The disadvantage of our method is that it requires manual work to be carried out by a human that knows the topology of the environment. The manual work, however, has to be done only once for a dataset and then allows other researchers to evaluate their methods easily. In this paper, we present manually obtained relative relations for different datasets that can be used for carrying out comparisons. We furthermore provide evaluations for the results of three different mapping techniques, namely scan-matching, SLAM using Rao-Blackwellized particle filter [Grisetti *et al.*, 2007b; Stachniss *et al.*, 2007b], and a maximum likelihood SLAM approach based on the graph formulation [Grisetti *et al.*, 2007c; Olson, 2008].

The remainder of this paper is organized a follows. First, we discuss related work in Section 2 and present the proposed metric based on relative relations between poses along

the trajectory of the robot in Section 3. Then, in Section 4 and Section 5 we explain how to obtain such relations in practice. In Section 6, we briefly discuss how to benchmark if the tested SLAM system does not provide pose estimates. Next, in Section 7 we provide a brief overview of the datasets used for benchmarking and in Section 8 we present our experiments which illustrate different properties of our method and we give benchmark results for three existing SLAM approaches.

2 Related Work

Learning maps is a frequently studied problem in the robotics literature. Mapping techniques for mobile robots can be classified according to the underlying estimation technique. The most popular approaches are extended Kalman filters (EKFs) [Leonard and Durrant-Whyte, 1991; Smith *et al.*, 1990], sparse extended information filters [Eustice *et al.*, 2005a; Thrun *et al.*, 2004], particle filters [Montemerlo *et al.*, 2003; Grisetti *et al.*, 2007b], and least square error minimization approaches [Lu and Milios, 1997; Frese *et al.*, 2005; Gutmann and Konolige, 1999; Olson *et al.*, 2006]. For some applications, it might even be sufficient to learn local maps only [Hermosillo *et al.*, 2003; Thrun and colleagues, 2006; Yguel *et al.*, 2007].

The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. If these assumptions are violated the filter is likely to diverge [Julier *et al.*, 1995; Uhlmann, 1995].

Thrun *et al.* [2004] proposed a method to correct the poses of a robot based on the inverse of the covariance matrix. The advantage of sparse extended information filters (SEIFs) is that they make use of the approximative sparsity of the information matrix. Eustice *et al.* [2005a] presented a technique that more accurately computes the error-bounds within the SEIF framework and therefore reduces the risk of becoming overly confident.

Dellaert and colleagues proposed a smoothing method called square root smoothing and mapping (SAM) [Dellaert, 2005; Kaess *et al.*, 2007; Ranganathan *et al.*, 2007]. It has several advantages compared to EKF-based solutions since it better covers the non-linearities and is faster to compute. In contrast to SEIFs, it furthermore provides an exactly sparse factorization of the information matrix. In addition to that, SAM can be applied in an incremental way [Kaess *et al.*, 2007] and is able to learn maps in 2D and 3D.

Frese's TreeMap algorithm [Frese, 2006] can be applied to compute nonlinear map estimates. It relies on a strong topological assumption on the map to perform sparsification of the information matrix. This approximation ignores small entries in the information matrix. In this way, Frese is able to perform an update in $\mathcal{O}(\log n)$ where n is the number of features.

An alternative approach to find maximum likelihood maps is the application of least square error minimization. The idea is to compute a network of constraints given the sequence of sensor readings. It should be noted that our approach for evaluating SLAM methods presented in this paper is highly related to this formulation of the SLAM problem.

Lu and Milios [1997] introduced the concept of graph-based or network-based SLAM using a kind of brute force method for optimization. Their approach seeks to optimize the whole network at once. Gutmann and Konolige [1999] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Duckett *et al.* [2002] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of relations. To make the problem linear, they assume

knowledge about the orientation of the robot. Frese *et al.* [2005] propose a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions. MLR is reported to provide very good results in flat environments especially if the error in the initial guess is limited.

Olson *et al.* [2006] presented an optimization approach that applies stochastic gradient descent for resolving relations in a network efficiently. Extensions of this work have been presented by Grisetti *et al.* [2007c; 2007a] Most approaches to graph-based SLAM such as the work of Olson *et al.*, Grisetti *et al.*, Frese *et al.*, and others focus on computing the best map and assume that the relations are given. The ATLAS framework [Bosse *et al.*, 2003], hierarchical SLAM [Estrada *et al.*, 2005], or the work of Nüchter *et al.* [2005], for example, can be used to obtain the constraints. In the graph-based mapping approach used in this paper, we followed the work of Olson [2008] to extract constraints and applied [Grisetti *et al.*, 2007c] for computing the minimal error configuration.

Activities related to performance metrics for SLAM methods, such as the work described in this paper, can roughly be divided into three major categories: First, competition settings where robot systems are competing within a defined problem scenario, such as playing soccer, navigating through a desert, or searching for victims. Second, collections of publicly available datasets that are provided for comparing algorithms on specific problems. Third, related publications that introduce methodologies and scoring metrics for comparing different methods.

The comparison of robots within benchmarking scenarios is a straight-forward approach for identifying specific system properties that can be generalized to other problem types. For this purpose numerous robot competitions have been initiated in the past, evaluating the performance of cleaning robots [EPFL and IROS, 2002], robots in simulated Mars environments [ESA, 2008], robots playing soccer or rescuing victims after a disaster [RoboCup Federation, 2009], and cars driving autonomously in an urban area [Darpa, 2007]. However, competition settings are likely to generate additional noise due to differing hardware and software settings. For example, when comparing mapping solutions in the RoboCup Rescue domain, the quality of maps generated using climbing robots can greatly differ from those generated on wheel-based robots operating in the plane. Furthermore, the approaches are often tuned to the settings addressed in the competitions.

Benchmarking of systems from datasets has reached a rather mature level in the vision community. There exist numerous data bases and performance measures, which are available via the Internet. Their purpose is to validate, for example, image annotation [Torralba *et al.*, 2007], range image segmentation [Hoover *et al.*, 1996], and stereo vision correspondence algorithms [Scharstein and Szeliski, 2002]. These image databases provide ground truth data [Torralba *et al.*, 2007; Scharstein and Szeliski, 2002], tools for generating ground truth [Torralba *et al.*, 2007] and computing the scoring metric [Scharstein and Szeliski, 2002], and an online ranking of results from different methods [Scharstein and Szeliski, 2002] for direct comparison.

In the robotics community, there are some well-known web sites providing datasets [Howard and Roy, 2003; Bonarini *et al.*, 2006] and algorithms [Stachniss *et al.*, 2007a] for mapping. However, they neither provide ground truth data nor recommendations on how to compare different maps in a meaningful way.

Some preliminary steps towards benchmarking navigation solutions have been presented in the past. Amigoni *et al.* [2007] presented a general methodology for performing experimental activities in the area of robotic mapping. They suggested a number of issues that should be addressed when experimentally validating a mapping method. For example, the mapping system should be applied to publicly available data, parameters of the algorithm

should be clearly indicated (and also effects of their variations presented), as well as parameters of the map should be explained. When ground truth data is available, they suggest to utilize the Hausdorff metric for map comparison.

Wulf *et al.* [2008] proposed the idea of using manually supervised Monte Carlo Localization (MCL) for matching 3D scans against a reference map. They suggested that a reference map be generated maps from independently created CAD data, which can be obtained from the land registry office. The comparison between generated map and ground truth has been carried out by computing the Euclidean distance and angle difference of each scan, and plotting these over time. Furthermore, they provided standard deviation and maximum error of the track for comparisons. We argue that comparing the absolute error between two tracks might not yield a meaningful assertion in all cases as illustrated in the initial example in Section 3. This effect gets even stronger when the robot makes a small angular error especially in the beginning of the dataset (and when it does not return to this place again). Then, large parts or the overall map are likely to be consistent, the error, however, will be huge. Therefore, the method proposed in this paper favors comparisons between relative poses along the trajectory of the robot. Based on the selection between which pose relations are considered, different properties can be highlighted.

Balaguer *et al.* [2007] utilize the USARSim robot simulator and a real robot platform for comparing different open source SLAM approaches. They demonstrated that maps resulting from processing simulator data are very close to those resulting from real robot data. Hence, they concluded that the simulator engine could be used for systematically benchmarking different approaches of SLAM. However, it has also been shown that noise is often but not always Gaussian in the SLAM context [Stachniss *et al.*, 2007b]. Gaussian noise, however, is typically used in most simulation systems. In addition to that, Balaguer *et al.* do not provide a quantitative measure for comparing generated maps with ground truth. As with many other approaches, their comparisons were carried out by visual inspection.

The paper presented here extends our work [Burgard *et al.*, 2009] with a more detailed description of the approach, a technique for extracting relations from aerial images, and a significantly extended experimental evaluation.

3 Metric for Benchmarking SLAM Algorithms

In this paper, we propose a metric for measuring the performance of a SLAM algorithm *not by comparing the map itself but by considering the poses of the robot during data acquisition*. In this way, we gain two important properties: First, it allows us to compare the result of algorithms that generate different types of metric map representations, such as feature-maps or occupancy grid maps. Second, the method is invariant to the sensor setup of the robot. Thus, a result of a graph-based SLAM approach working on laser range data can be compared, for example, with the result of vision-based FastSLAM. The only property we require is that the SLAM algorithm estimates the trajectory of the robot given by a set of poses at which observations are made. All benchmark computations will be performed on this set.

3.1 A Measure for Benchmarking SLAM Results

Let $x_{1:T}$ be the poses of the robot estimated by a SLAM algorithm from time step 1 to T. Let $x^*_{1:T}$ be the reference poses of the robot, ideally the true locations. A straightforward error

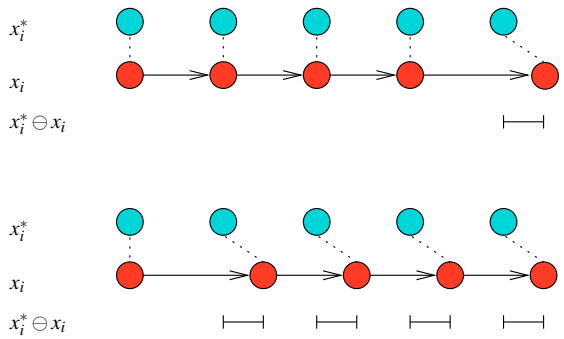


Fig. 1 This figure illustrates a simple example where the metric in Eq. 1 fails. The light blue circles show the reference positions of the robot $\{x_i^*\}$ while the dark red circles show the estimated positions of the robot $\{x_i\}$. The correspondence between the estimated locations and the ground truth is shown with dashed lines, and the direction of motion of the robot is highlighted with arrows. In the situation shown in the upper part, the robot makes a small mistake at the end of the path. This results in a small error. Conversely, in the situation illustrated on the bottom part of the figure the robot makes a small error of the same entity, but at the beginning of the travel, thus resulting in a much bigger global error.

metric could be defined as

$$\varepsilon(x_{1:T}) = \sum_{t=1}^{T} (x_t \ominus x_t^*)^2, \tag{1}$$

where $\oplus$ is the standard motion composition operator and $\ominus$ its inverse. Let $\delta_{i,j} = x_j \ominus x_i$ be the relative transformation that moves the node x_i onto x_j and accordingly $\delta_{i,j}^* = x_j^* \ominus x_i^*$. Eq. 1 can be rewritten as

$$\varepsilon(x_{1:T}) = \sum_{t=1}^{T} \left((x_1 \oplus \delta_{1,2} \oplus \ldots \oplus \delta_{t-1,t}) \ominus (x_1^* \oplus \delta_{1,2}^* \oplus \ldots \oplus \delta_{t-1,t}^*) \right)^2 \tag{2}$$

We claim that this metric is suboptimal for comparing the result of a SLAM algorithm. To illustrate this, consider the following 1D example in which a robot travels along a straight line. Let the robot make a translational error of e during the first motion, $\delta_{1,2} = \delta_{1,2}^* + e$, and perfect estimates at all other points in time $\delta_{t,t+1} = \delta_{t,t+1}^*$ for $t > 1$. Thus, the error according to Eq. 2, will be $T \cdot e$, since $\delta_{1,2}$ is contained in every pose estimate for $t > 1$. If, however, we estimate the trajectory backwards starting from x_T to x_1 or alternatively by shifting the whole map by e, we obtain an error of e only. This indicates, that such an error estimate is suboptimal for comparing the results of a SLAM algorithm. See also Figure 1 for an illustration.

In the past, the so-called NEES measure proposed in [Bar-Shalom *et al.*, 2001] as

$$\varepsilon(x_{1:T}) = \sum_{t=1}^{T} (x_t - x_t^*)^T \Omega_t (x_t - x_t^*), \tag{3}$$

has often been used to evaluate the results of a SLAM approach (e.g., [Eustice *et al.*, 2005b]). Here Ω_t represents the information matrix of the pose x_t. The NEES measure, however, suffers from a similar problem as Eq. 1 when computing ε. In addition to that, not all SLAM algorithms provide an estimate of the information matrix and thus cannot be compared based on Eq. 3.

Based on this experience, we propose a measure that considers the deformation energy that is needed to transfer the estimate into the ground truth. This can be done — similar to the ideas of the graph mapping introduced by Lu and Milios [1997] — by considering the nodes as masses and connections between them as springs. Thus, our metric is based on the *relative* displacement between robot poses. Instead of comparing x to x^* (in the global reference frame), we do the operation based on δ and δ^* as

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \tag{4}$$

where N is the number of relative relations and $trans(\cdot)$ and $rot(\cdot)$ are used to separate and weight the translational and rotational components. We suggest that both quantities be evaluated individually. In this case, the error (or transformation energy) in the above-mentioned example will be consistently estimated as the single rotational error no matter where the error occurs in the space or in which order the data is processed.

Our error metric, however, leaves open which relative displacements $\delta_{i,j}$ are included in the summation in Eq. 4. Using the metric and selecting relations are two related but different problems. Evaluating two approaches based on a different set of relative pose displacements will obviously result in two different scores. As we will show in the remainder of this section, the set δ and thus δ^* can be defined to highlight certain properties of an algorithm.

Note that some researchers prefer the absolute error (absolute value, not squared) instead of the squared one. We prefer the squared one since it derives from the motivation that the metric measures the energy needed to transform the estimated trajectory into ground truth. However, one can also use the metric using the non-squared error instead of the squared one. In the experimental evaluation, we actually provide both values.

3.2 Selecting Relative Displacements for Evaluation

Benchmarks are designed to compare different algorithms. In the case of SLAM systems, however, the task the robot finally has to solve should define the required accuracy and this information should be considered in the measure.

For example, a robot generating blueprints of buildings should reflect the geometry of a building as accurately as possible. In contrast to that, a robot performing navigation tasks requires a map that can be used to robustly localize itself and to compute valid trajectories to a goal location. To carry out this task, it is sufficient in most cases that the map is topologically consistent and that its observations can be locally matched to the map, i.e. its spatial structure is correctly representing the environment. We refer to a map having this property as being locally consistent. Figure 3 illustrates the concept of locally consistent maps which are suited for a robot to carry out navigation tasks.

By selecting the relative displacements $\delta_{i,j}$ used in Eq. 4 for a given dataset, the user can highlight certain properties and thus design a measure for evaluating an approach given the application in mind.

For example, by adding only known relative displacements between nearby poses based on visibility, a local consistency is highlighted. In contrast to that, by adding known relative displacements of far away poses, for example, provided by an accurate external measurement device or by background knowledge, the accuracy of the overall geometry of the mapped environment is enforced. In this way, one can incorporate additional knowledge (for example, that a corridor has a certain length and is straight) into the benchmark.

4 Obtaining Reference Relations in Indoor Environments

In practice, the key question regarding Eq. 4 is how to determine the *true relative displacements* between poses. Obviously, the true values are not available. However, we can determine close-to-true values by using the information recorded by the mobile robot and the background knowledge of the human recording the datasets, which, of course, involves manual work.

Please note, that the metric presented above is independent of the actual sensor used. In the remainder of this paper, however, we will concentrate on robots equipped with a laser range finders, since they are probably the most popular sensors in robotics at the moment.To evaluate an approach operating on a different sensor modality, one has two possibilities to generate relations. One way would be to temporarily mount a laser range finder on the robot and calibrate it in the robot coordinate frame. If this is not possible, one has to provide a method for accurately determining the relative displacements between two poses from which an observation has been taken that observes the same part of the space.

4.1 Initial Guess

In our work, we propose the following strategy. First, one tries to find an initial guess about the relative displacement between poses. Based on the knowledge of the human, a wrong initial guess can be easily discarded since the human "knows" the structure of the environment. In a second step, a refinement is proposed based on manual interaction.

4.1.1 Symeo System

One way for obtaining good initial guesses with no or only very few interactions can be the use of the Symeo Positioning System LPR-B [Symeo GmbH, 2008]. It works similar to a local GPS system but indoors and can achieve a localization accuracy of around 5 cm to 10 cm. The problem is that such a system designed for industrial applications is typically not present at most robotics labs. If available, however, it is well suited for a rather accurate initial guess of the robot's position.

4.1.2 Initial Estimate via SLAM Approaches

In most cases, however, researchers in robotics will have SLAM algorithms at hand that can be used to compute an initial guess about the poses of the robot. In the recent years, several accurate methods have been proposed to serve as such a guess (see Section 2). By manually inspecting the estimates of the algorithm, a human can accept, refine, or discard a match and also add missing relations.

It is important to note that the output is not more than an initial guess and it is used to estimate the visibility constraints which will be used in the next step.

4.2 Manual Matching Refinement and Rejection

Based on the initial guess about the position of the robot for a given time step, it is possible to determine which observations in the dataset should have covered the same part of the

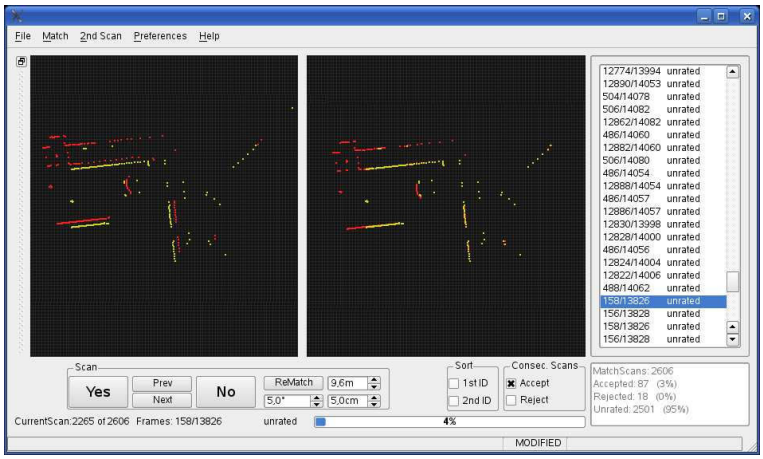


Fig. 2 User interface for matching, accepting, and discarding pairs of observations.

space or the same objects. For a laser range finder, this can easily be achieved. Between each visible pair of poses, one adds a relative displacement into a candidate set.

In the next step, a human processes the candidate set to eliminate wrong hypotheses by visualizing the observation in a common reference frame. This requires manual interaction but allows for eliminating wrong matches and outliers with high precision, since the user is able to incorporate his background knowledge about the environment.

Since we aim to find the best possible relative displacement, we perform a pair-wise registration procedure to refine the estimates of the observation registration method. It furthermore allows the user to manually adjust the relative offset between poses so that the pairs of observations fit better. Alternatively, the pair can be discarded.

This approach might sound labor-intensive but with an appropriate user interface, this task can be carried out without a large waste of resources. For example, for a standard dataset with 1,700 relations, it took an unexperienced user approximately four hours to extract the relative translations that then served as the input to the error calculation. Figure 2 shows a screen-shot of the user interface used for evaluation.

It should be noted that for the manual work described above some kind of structure in the environment is required. The manual labor might be very hard in highly unstructured scenes.

4.3 Other Relations

In addition to the relative transformations added upon visibility and matching of observations, one can directly incorporate additional relations resulting from other sources of information, for example, given the knowledge about the length of a corridor in an environment. By adding a relation between two poses — each at one side of the corridor — one can incorporate knowledge about the global geometry of an environment if this is available. This fact is, for example, illustrated by the black dashed line in Figure 3 that implies a known distance between two poses in a corridor that are not adjacent. Figure 4 plots the corresponding error identified by the relation.

In the experimental evaluation, we will show one example for such additional relations used in real world datasets. In this example, we utilize relations derived from satellite image data.

5 Obtaining Reference Relations in Outdoor Environments

The techniques described in the previous section can be used to obtain a close-to-ground-truth for indoor environments. In outdoor scenarios however, the manual validation of the data is usually less practical due to the reduced structure and the large size. In wide open areas it may be difficult for a human operator to determine whether a potential alignment between laser scans is good or not due to the limited range of the scanner. Furthermore the poor structure of the environment makes this procedure hard even when the laser senses a high number of obstacles.

GPS is commonly used to bound the global uncertainty of a vehicle moving outdoors. Unfortunately, GPS suffers from outages or occlusions so that a robot relying on GPS might encounter substantial positioning errors. Especially in urban environments, GPS is known to be noisy. Even sophisticated SLAM algorithms cannot fully compensate for these errors as there still might be lacking relations between observations combined with large odometry errors that introduce a high uncertainty in the current position of the vehicle.

As an alternative to GPS, it is possible to use aerial images to determine relations close to the ground truth. We investigated this approach in our previous work [Kümmerle *et al.*, 2009] and we show that this solution yields a better global consistency of the resulting map, if we consider the prior information. Satellite images of locations are widely available on the web by popular tools like Google-Earth or Microsoft Live-Earth. This data can be used as prior information to localize a robot equipped with a 3D laser range finder.

The overall approach is based on the Monte-Carlo localization framework [Dellaert *et al.*, 1998]. The satellite images are captured from a viewpoint significantly different from the one of the robot. However, by using 3D scans we can extract 2D information which is more likely to be consistent with the one visible in the reference map. In this way, we can prevent the system from introducing inconsistent prior information.

In the following, we explain how we adapted Monte Carlo Localization (MCL) to operate on aerial images and how to select points from 3D scans to be considered in the observation model of MCL. This procedure returns a set of candidate robot locations $\hat{x}_i$. From those positions, we then select a subset of pairs of locations from which to compute the reference displacements $\hat{\hat{\delta}}_{i,j}$ to be used in the metric.

5.1 Monte Carlo Localization

To estimate the pose x of the robot in its environment, we consider probabilistic localization, which follows the recursive Bayesian filtering scheme. The key idea of this approach is to maintain a probability density $p(x_t \mid z_{1:t}, u_{0:t-1})$ of the location x_t of the robot at time t given all observations $z_{1:t}$ and all control inputs $u_{0:t-1}$. This posterior is updated as follows:

$$p(x_t \mid z_{1:t}, u_{0:t-1}) = \\ \alpha \cdot p(z_t \mid x_t) \cdot \int p(x_t \mid u_{t-1}, x_{t-1}) \cdot p(x_{t-1})\, dx_{t-1}. \tag{5}$$

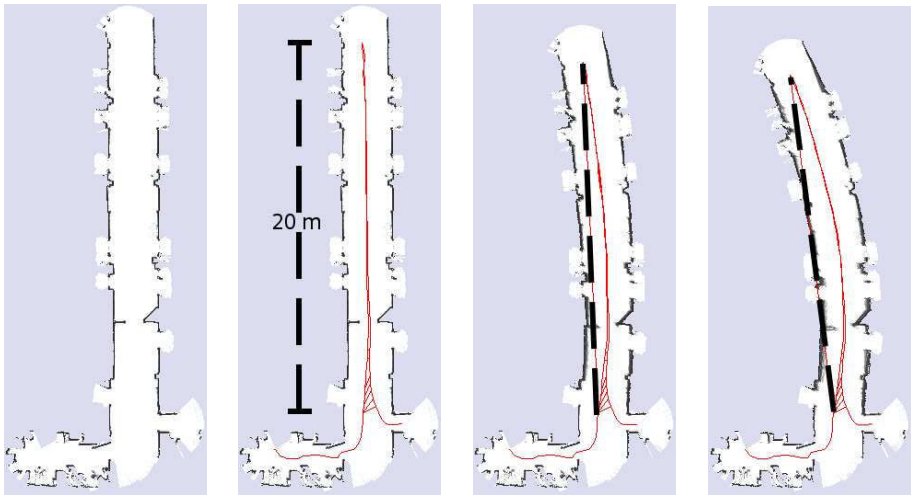


Fig. 3 Example of the performance measure on maps generated using different sensor setups. The relations between close-by positions are determined by a human assisted scan-alignment procedure performed on scans acquired at close-by locations. The long dashed line represents a relation added by manually measuring the relative distance at two locations of the robot: (a) the reference map obtained from the relative measurements, (b) the reference map superimposed with the network of relative measurements, (c) a map obtained by scan matching using a 4 meters range sensor, with the superimposed relation (this map is still usable for navigating a robot), (d) a map obtained by cropping the range of the sensor to 3 meters. Whereas the quality of the rightmost map is visibly decreased, it is also adequate for robot navigation since it preserves a correct topology of the environment (all doorways are still visible) and it correctly reflects the local spatial structure of the corridor. Therefore, it is locally consistent, but not globally consistent as (a). See also Figure 4 for corresponding error plots.

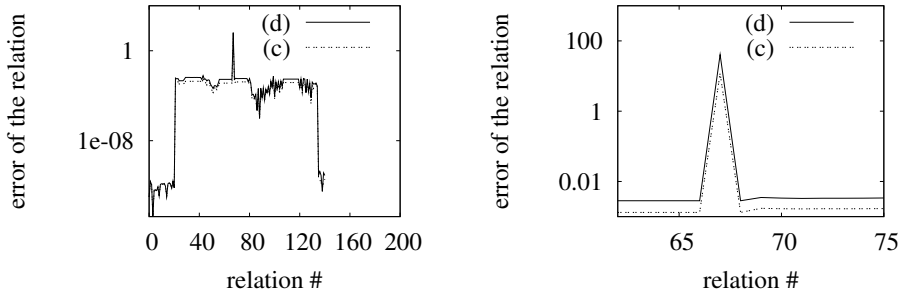
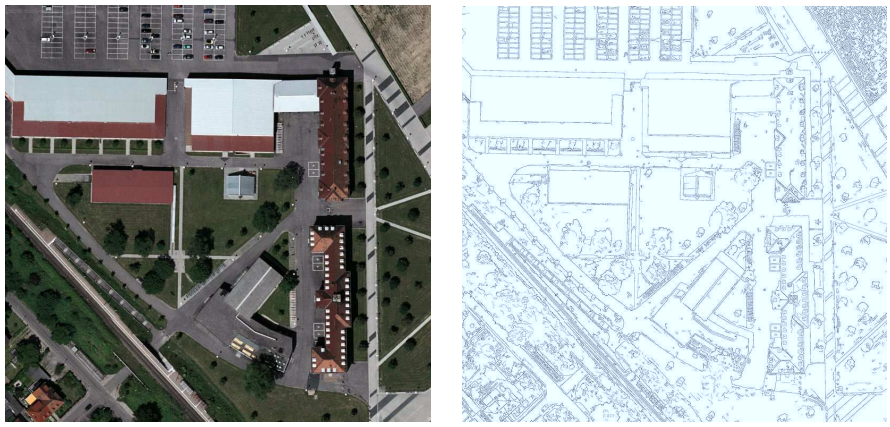


Fig. 4 This figure shows the behavior of the error metric for the maps (c) and (d) in Figure 3. On the left we plot the error introduced by the individual relations. The right plot is a magnification of the left one in the region corresponding to the manually introduced relations marked on the images with the dashed line. This results in a substantial increase of the global ε of SLAM results under comparison.



(a) (b)

Fig. 5 **(a)** A Google Earth image of the Freiburg campus. **(b)** The corresponding Canny image. Despite the considerable clutter, the structure of the buildings and the vertical elements are clearly visible.

Here, α is a normalization constant which ensures that $p(x_t \mid z_{1:t}, u_{0:t-1})$ sums up to one over all x_t. The terms to be described in Eq. 5 are the prediction model $p(x_t \mid u_{t-1}, x_{t-1})$ and the sensor model $p(z_t \mid x_t)$. One contribution of this work is an appropriate computation of the sensor model in the case that a robot equipped with a 3D range sensor operates in a given birds-eye map.

MCL is a variant of particle filtering [Doucet *et al.*, 2001] where each particle corresponds to a possible robot pose and has an assigned weight $w^{[i]}$. The belief update from Eq. 5 is performed according to the following two alternating steps:

1. In the prediction step, we draw for each particle with weight $w^{[i]}$ a new particle according to $w^{[i]}$ and to the prediction model $p(x_t \mid u_{t-1}, x_{t-1})$.
2. In the correction step, a new observation z_t is integrated. This is done by assigning a new weight $w^{[i]}$ to each particle according to the sensor model $p(z_t \mid x_t)$.

Furthermore, the particle set needs to be re-sampled according to the assigned weights to obtain a good approximation of the pose distribution with a finite number of particles.

So far, we have described the general framework of MCL. In the next section, we will describe the sensor model for determining the likelihood $p(z_t \mid x_t)$ of perceiving the 3D scan z_t from a given robot position x_t within an aerial image. For convenience, we will drop the time index t in the remainder of this section.

5.2 Sensor Model for 3D Range Scans in Aerial Images

The task of the sensor model is to determine the likelihood $p(z \mid x)$ of a reading z given the robot is at pose x. In our current system, we apply the so called endpoint model or likelihood fields [Thrun *et al.*, 2005]. Let z^k be the endpoints of a 3D scan z. The endpoint model computes the likelihood of a reading based only on the distances between a scan point $\hat{z}^k$ re-projected onto the map according to the pose x of the robot and the point in the map $\hat{d}^k$ which is closest to $\hat{z}^k$ as:

$$p(z \mid x) = f(\|\hat{z}^1 - \hat{d}^1\|, \ldots, \|\hat{z}^k - \hat{d}^k\|). \tag{6}$$

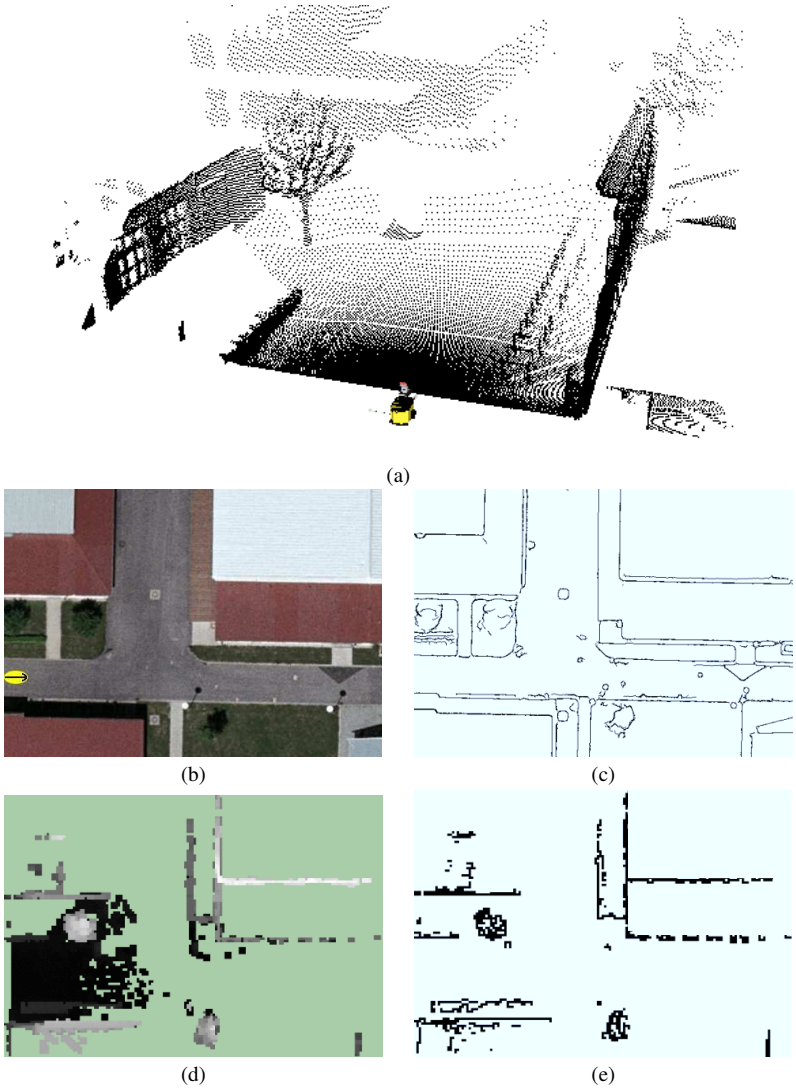


Fig. 6 **(a)** A 3D scan represented as a point cloud. **(b)** The aerial image of the scene. **(c)** The Canny edges extracted from (b). **(d)** A view from the top, where the gray value represents the maximal height per cell. The darker the color the lower the height. **(e)** Extracted height variations from (d).

If we assume that the beams are independent and the sensor noise is normally distributed we can rewrite Eq. 6 as

$$f(\|\hat{z}^1 - \hat{d}^1\|, \ldots, \|\hat{z}^k - \hat{d}^k\|) \propto \prod_j e^{\frac{(\hat{z}^j - \hat{d}^j)^2}{\sigma^2}}. \tag{7}$$

Since the aerial image only contains 2D information about the scene, we need to select a set of beams from the 3D scan, which are likely to result in structures that can be identified

and matched in the image. In other words, we need to transform both the scan and the image into a set of 2D points which can be compared via the function $f(\cdot)$.

To extract these points from the image we employ the standard Canny edge extraction procedure [Canny, 1986]. The idea behind this is that if there is a height gap in the aerial image, there will often also be a visible change in intensity in the aerial image and we assume that this intensity change is detected by the edge extraction procedure. In an urban environment, such edges typically correspond to borders of roofs, trees, fences or other structures. Of course, the edge extraction procedure returns a lot of false positives that do not represent any actual 3D structure, like street markings, grass borders, shadows, and other flat markings. All these aspects have to be considered by the sensor model.

A straightforward way to address this problem is to select a subset of beams z^k from the 3D scan which will then be used to compute the likelihood. The beams which should be considered are the ones which correspond to significant variations along the z direction of the 3D scan. For vertical structures, a direct matching between the extracted edges and the measurements of a horizontal 2D laser range scanner can be performed, as discussed by Früh and Zakhor [2004]. If a 3D laser range finder is available, we also attempt to match variations in height that are not purely vertical structures, like trees or overhanging roofs. This procedure is illustrated by the sequence of images in Figure 6.

In the current implementation, we considered variations in height of 0.5 m and above as possible positions of edges that could also be visible in the aerial image. The positions of these variations relative to the robot can then be matched against the Canny edges of the aerial image in a point-by-point fashion, similar to the matching of 2D-laser scans against an occupancy grid map. Additionally, we employ a heuristic to detect when the prior is not available, i.e., when the robot is inside of a building or under overhanging structures. This is based on the 3D perception. If there is a ceiling which leads to range measurements above the robot no global relations from the localization are integrated, since we assume that the area the robot is sensing is not visible in the aerial image.

Figure 7 shows an example trajectory estimated with this technique (in red) and the GPS positions (in blue). As can be seen, the estimates are more accurate than the GPS data. Thus the improved guess facilitates the manual verification of the data. Note that the approach presented here is used to obtain the candidate relations for outdoor datasets. A human operator has to accept or decline all relations found by the approach.

6 Benchmarking for Algorithms without Trajectory Estimates

A series of SLAM approaches estimate the trajectory of the robot as well as a map. However, in the context of the EKF, researchers often exclude an estimate of the full trajectory to lower the computational load. To facilitate evaluation one could store the current pose for each processing step and use it to build a trajectory. This would lead to good results if only local accuracy is considered. However, global corrections appearing later in run-time are not represented correctly.

We see two solutions to overcome this problem: (a) depending on the capabilities of the sensor, one can recover the trajectory as a post processing step given the feature locations and the data association estimated by the approach. This procedure could be quite easily realized by a localization run in the built map with given data association (the data association of the SLAM algorithm). (b) in some settings this strategy can be difficult and one might argue that a comparison based on the landmark locations is more desirable. In this case, one can apply our metric operating on the landmark locations instead of based on the

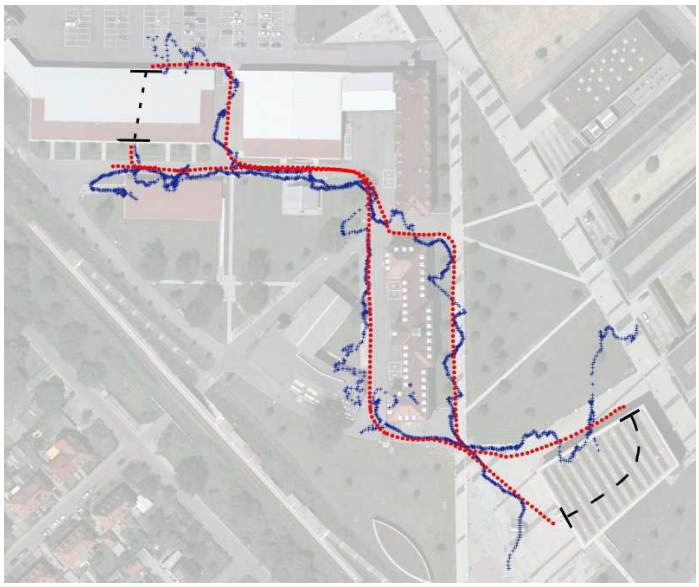


Fig. 7 Trajectory estimated using satellite images versus GPS data overlaid on the image of the ALU-FR campus.

poses of the robot. In this case, the relations $\delta^*_{i,j}$ can be determined by measuring the relative distances between landmarks using, for example, a highly accurate measurement device.

The disadvantage of this approach is that the data association between estimated landmarks and ground truth landmarks is not given. Depending on the kind of observations, a human can manually determine the data association for each observation of an evaluation datasets as done by Frese [2008]. This, however, might get intractable for SIFT-like features obtained with high frame rate cameras. Note that all metrics measuring an error based on landmark locations require such a data association as given. Furthermore, it becomes impossible to compare significantly different SLAM systems using different sensing modalities. Therefore, we would recommend the first option to evaluate techniques such as EKF.

7 Datasets for Benchmarking

To validate the metric, we selected a set of datasets representing different kinds of environments from the publicly available datasets. We extracted relative relations between robot poses using the methods described in the previous sections by manually validating every single observation between pairs of poses.

As a challenging indoor corridor-environment with a non-trivial topology including nested loops, we selected the MIT Killian Court dataset [1] (Infinite Corridor) and the dataset of the ACES building at the University of Texas, Austin [2]. As a typical office environment with a significant level of clutter, we selected the dataset of building 079 at the University of Freiburg, the Intel Research Lab dataset [3], and a dataset acquired at the CSAIL at

[1] Courtesy of Mike Bosse

[2] Courtesy of Patrick Beeson

[3] Courtesy of Dirk Haehnel

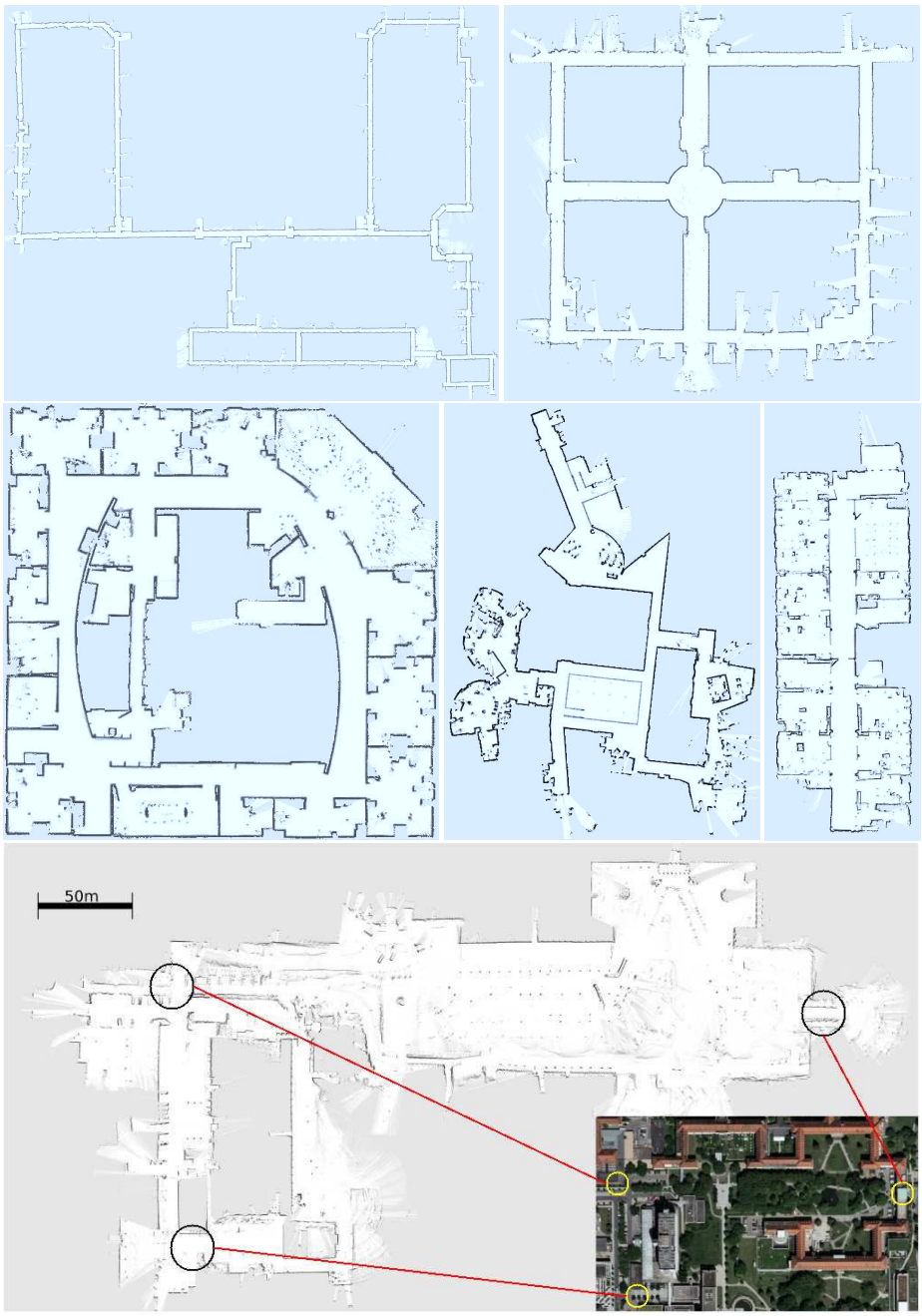


Fig. 8 Maps obtained by the reference datasets used to validate our metric. From top to bottom and left to right: MIT Killian Court (Boston), ACES Building (Austin), Intel Research Lab (Seattle), MIT CS Building (Boston), building 079 University of Freiburg, and the University Hospital in Freiburg. The depicted map of the University Hospital was obtained by using the background information extracted from the satellite images.

MIT. For addressing outdoor environments, we recorded a new dataset at the park area of the University Hospital, Freiburg. To give a visual impression of the scanned environments, Figure 8 illustrates maps obtained by executing state-of-the-art SLAM algorithms [Grisetti *et al.*, 2007b; 2007c; Olson, 2008]. All datasets, the manually verified relations, and map images are available online at:
`http://ais.informatik.uni-freiburg.de/slamevaluation/`

8 Experimental Evaluation

This evaluation is designed to illustrate the properties of our method. We selected three popular mapping techniques, namely scan matching, a Rao-Blackwellized particle filter-based approach, and a graph-based solution to the SLAM problem and processed the datasets discussed in the previous section.

We provide the scores obtained from the metric for all combinations of SLAM approach and dataset. This will allow other researchers to compare their own SLAM approaches against our methods using the provided benchmark datasets. In addition, we also present sub-optimally corrected trajectories in this section to illustrate how inconsistencies affect the score of the metric. We will show that our error metric is well-suited for benchmarking and this kind of evaluation.

8.1 Evaluation of Existing Approaches using the Proposed Metric

In this evaluation, we considered the following mapping approaches:

Scan Matching: Scan matching is the computation of the incremental, open loop maximum likelihood trajectory of the robot by matching consecutive scans [Lu and Milios, 1994; Censi, 2006]. In small environments, a scan matching algorithm is generally sufficient to obtain accurate maps with a comparably small computational effort. However, the estimate of the robot trajectory computed by scan matching is affected by an increasing error which becomes visible whenever the robot reenters in known regions after visiting large unknown areas (loop closing or place revisiting).

Grid-based Rao-Blackwellized Particle Filter (RBPF) for SLAM: We use the RBPF implementation described in [Grisetti *et al.*, 2007b; Stachniss *et al.*, 2007b] which is available online [Stachniss *et al.*, 2007a]. It estimates the posterior over maps and trajectories by means of a particle filter. Each particle carries its own map and a hypothesis of the robot pose within that map. The approach uses an informed proposal distribution for particle generation that is optimized to laser range data. In the evaluation presented here, we used 50 particles. Note that a higher number of samples may improve the performance of the algorithm.

Graph Mapping: This approach computes a map by means of graph optimization [Grisetti *et al.*, 2007c]. The idea is to construct a graph out of the sequence of measurements. Every node in the graph represents a pose along the trajectory taken by the robot and the corresponding measurement obtained at that pose. Then, a least square error minimization approach is applied to obtain the most-likely configuration of the graph. In general, it is non-trivial to find the constraints, often referred to as the data association problem. Especially in symmetric environments or in situations with large noise, the edges in the

Table 1 Quantitative results of different approaches/datasets on the translation error as well as the corresponding standard deviation and the maximum error. [1] scan matching has been applied as a preprocessing step to improve the odometry.

Translational error m (abs) / m^2 (sqr)	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces			
Eq. 4 using absolute errors	0.173 ± 0.614	0.060 ± 0.049	0.044 ± 0.044
Eq. 4 using squared errors	0.407 ± 2.726	0.006 ± 0.011	0.004 ± 0.009
Maximum absolute error of a relation	4.869	0.433	0.347
Intel			
Eq. 4 using absolute errors	0.220 ± 0.296	0.070 ± 0.083	0.031 ± 0.026
Eq. 4 using squared errors	0.136 ± 0.277	0.011 ± 0.034	0.002 ± 0.004
Maximum absolute error of a relation	1.168	0.698	0.229
MIT Killian Court			
Eq. 4 using absolute errors	1.651 ± 4.138	0.122 ± 0.386[1]	0.050 ± 0.056
Eq. 4 using squared errors	19.85 ± 59.84	0.164 ± 0.814[1]	0.006 ± 0.029
Maximum absolute error of a relation	19.467	2.513[1]	0.765
MIT CSAIL			
Eq. 4 using absolute errors	0.106 ± 0.325	0.049 ± 0.049[1]	0.004 ± 0.009
Eq. 4 using squared errors	0.117 ± 0.728	0.005 ± 0.013[1]	0.0001 ± 0.0005
Maximum absolute error of a relation	3.570	0.508[1]	0.096
Freiburg bldg 79			
Eq. 4 using absolute errors	0.258 ± 0.427	0.061 ± 0.044[1]	0.056 ± 0.042
Eq. 4 using squared errors	0.249 ± 0.687	0.006 ± 0.020[1]	0.005 ± 0.011
Maximum absolute error of a relation	2.280	0.856[1]	0.459
Freiburg Hospital			
Eq. 4 using absolute errors	0.434 ± 1.615	0.637 ± 2.638	0.143 ± 0.180
Eq. 4 using squared errors	2.79 ± 18.19	7.367 ± 38.496	0.053 ± 0.272
Maximum absolute error of a relation	15.584	15.343	2.385
Freiburg Hospital, only global relations (see text)			
Eq. 4 using absolute errors	13.0 ± 11.6	12.3 ± 11.7	11.6 ± 11.9
Eq. 4 using squared errors	305.4 ± 518.9	288.8 ± 626.3	276.1 ± 516.5
Maximum absolute error of a relation	70.9	65.1	66.1

graph may be wrong or imprecise and thus the resulting map may yields inconsistencies. In our current implementation of the graph mapping system, we followed the approach of Olson [2008] to compute constraints.

For our evaluation, we manually extracted the relations for all datasets mentioned in the previous section. The manually extracted relations are available online, see Section 7. We then carried out the mapping approaches and used the corrected trajectory for computing the error according to our metric. Please note, that the error computed according to our metric (as well as for most other metrics too) can be separated into two components: a translational error and a rotational error. Often, a "weighting-factor" is used to combine both error terms into a single number, see, for example, [Pfaff *et al.*, 2006]. In our evaluation, however, we provide both terms separately for a better transparency of the results.

We processed all benchmark datasets from Section 7 using the algorithms listed above. A condensed view of each algorithm's performance is given by the averaged error over all relations. In Table 1, we give an overview on the translational error of the various algorithms, while Table 2 shows the rotational error. Comparing two algorithms can be done by comparing the values given in the tables, namely the maximum error as well as the average error. It can be seen that the more advanced algorithms (Rao-Blackwellized particle filter

Table 2 Quantitative results of different approaches/datasets on the rotational error as well as the corresponding standard deviation and the maximum error. [1] scan matching has been applied as a preprocessing step to improve the odometry.

Rotational error *deg* (abs) / deg^2 (sqr)	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces			
Eq. 4 using absolute errors	1.2 ± 1.5	1.2 ± 1.3	0.4 ± 0.4
Eq. 4 using squared errors	3.7 ± 10.7	3.1 ± 7.0	0.3 ± 0.8
Maximum absolute error of a relation	12.1	7.9	3.5
Intel			
Eq. 4 using absolute errors	1.7 ± 4.8	3.0 ± 5.3	1.3 ± 4.7
Eq. 4 using squared errors	25.8 ± 170.9	36.7 ± 187.7	24.0 ± 166.1
Maximum absolute error of a relation	4.5	34.7	6.4
MIT Killian Court			
Eq. 4 using absolute errors	2.3 ± 4.5	0.8 ± 0.8[1]	0.5 ± 0.5
Eq. 4 using squared errors	25.4 ± 65.0	0.9 ± 1.7[1]	0.9 ± 0.9
Maximum absolute error of a relation	21.6	7.4[1]	5.4
MIT CSAIL			
Eq. 4 using absolute errors	1.4 ± 4.5	0.6 ± 1.2[1]	0.05 ± 0.08
Eq. 4 using squared errors	22.3 ± 111.3	1.9 ± 17.3[1]	0.01 ± 0.04
Maximum absolute error of a relation	26.3	18.2[1]	0.8
Freiburg bldg 79			
Eq. 4 using absolute errors	1.7 ± 2.1	0.6 ± 0.6[1]	0.6 ± 0.6
Eq. 4 using squared errors	7.3 ± 14.5	0.7 ± 2.0[1]	0.7 ± 1.7
Maximum absolute error of a relation	9.9	6.4[1]	5.4
Freiburg Hospital			
Eq. 4 using absolute errors	1.3 ± 3.0	1.3 ± 2.3	0.9 ± 2.2
Eq. 4 using squared errors	10.9 ± 50.4	7.1 ± 42.2	5.5 ± 46.2
Maximum absolute error of a relation	27.4	28.0	29.6
Freiburg Hospital, only global relations (see text)			
Eq. 4 using absolute errors	6.3 ± 5.2	5.5 ± 5.9	6.3 ± 6.2
Eq. 4 using squared errors	66.1 ± 101.4	64.6 ± 144.2	77.2 ± 154.8
Maximum absolute error of a relation	27.3	35.1	38.6

and graph mapping) usually outperform scan matching. This is mainly caused by the fact that scan matching only locally optimizes the result and will introduce topological errors in the maps, especially when large loops have to be closed. A distinction between RBPF and graph mapping seems difficult as both algorithms perform well in general. On average, graph mapping seems to be slightly better than a RBPF for mapping. It should also be noted that for the outdoor dataset (Freiburg hospital), the RBPF mapper was not able to close the large loop and therefore was substantially worse than the graph mapper.

To visualize the results and to provide more insights about the metric, we do not provide the scores only but also plots showing the error of each relation. In case of high errors in a block of relations, we label the relations in the maps. This enables us to see not only where an algorithm fails, but can also provide insights as to why it fails. Inspecting those situations in correlation with the map helps to understand the properties of algorithms and gives valuable insights on its capabilities. For three datasets, a detailed analysis using these plots is presented in Section 8.2 to Section 8.4. The overall analysis provides the intuition that our metric is well-suited for evaluating SLAM approaches.

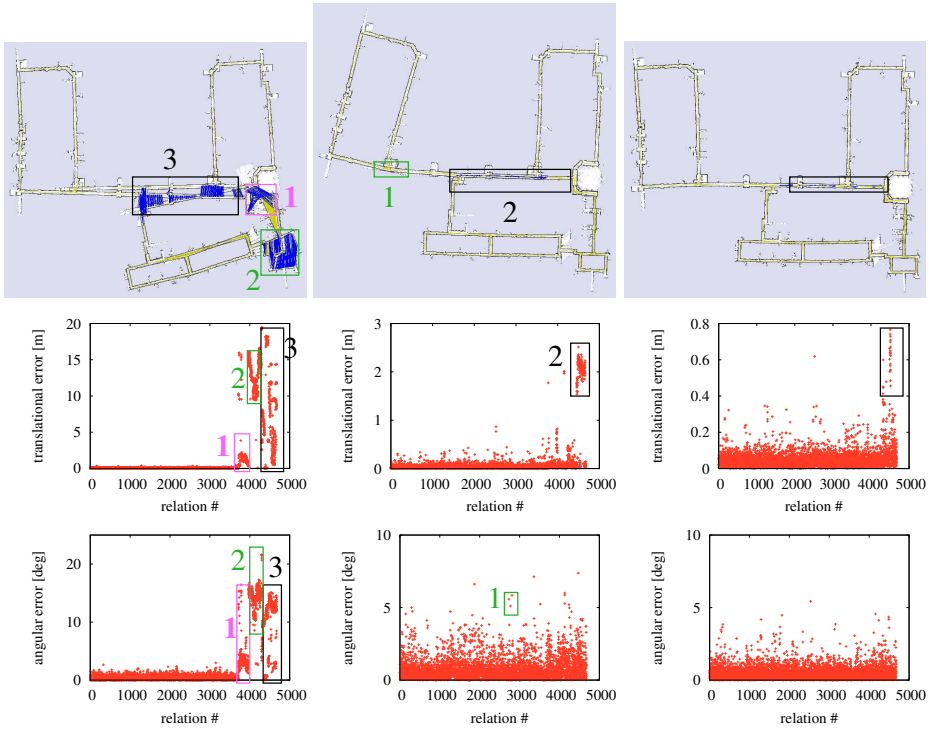


Fig. 9 This figure illustrates our metric applied to the MIT Killian Court dataset. The reference relations are depicted in light yellow, while the relations marked in the plots are shown in dark blue. The left column shows the results of pure scan-matching, the middle column the result of a RBPF-based technique with 50 samples, and the right column shows the result of a graph-based approach. The regions marked in the map correspond to regions in the error plots having high error. Due to its inability of dealing with loop closures scan matching has a high error when revisiting known regions. However, the absence of significant structure along the corridors for scan registration is an issue for both the graph-based and the RBPF approach. All in all, the graph-based approach outperforms the other methods.

8.2 MIT Killian Court

The MIT Killian Court dataset has been acquired in a large indoor environment, where the robot mainly observed corridors lacking structures that support accurate pose correction. The robot traverses multiple nested loops – a challenge especially for the RBPF-based technique. We extracted close to 5,000 relations between nearby poses that are used for evaluation. Figure 9 shows three different results and the corresponding error distributions to illustrate the capabilities of our method. Regions in the map with high inconsistencies correspond to relations having a high error. The absence of significant structure along the corridors results in a small or medium re-localization error of the robot in all compared approaches. In sum, we can say the graph-based approach outperforms the other methods and that the score of our metric reflects the impression of a human about map quality obtained by visually inspecting the mapping results (the vertical corridors in the upper part are supposed to be parallel).

8.3 Freiburg Indoor Building 079

The building 079 of the University of Freiburg is an example for an indoor office environment. The building consists of one corridor which connects the individual rooms. Figure 10 depicts the results of the individual algorithms (scan matching, RBPF, graph-based). In the first row of Figure 10, the relations having a translational error greater than 0.15 m are highlighted in blue.

In the left plot showing the scan matching result, the relations plotted in blue are generated when the robot revisits an already known region. These relations are visible in the corresponding error plots (Figure 10 first column, second and third row). As can be seen from the error plots, the relations with a number greater than 1,000 have a larger error than the rest of the dataset. The fact that the pose estimate of the robot is sub-optimal and that the error accumulates can also be seen by the rather blurry map and that some walls occur twice. In contrast to that, the more sophisticated algorithms, namely RBPF and graph mapping, are able to produce consistent and accurate maps in this environment. Only very few relations show an increased error (illustrated by dark blue relations).

8.4 Freiburg University Hospital

This dataset consists of 2D and 3D laser range data obtained with one statically mounted SICK scanner and one mounted on a pan-tilt unit. The robot was steered through a park area that contains a lot of bushes and which is surrounded by buildings. Most of the time, the robot was steered along a bike lane with cobble stone pavement. The area is around 500 m by 250 m in size.

Figure 11 depicts the three mapping results, one obtained with scan matching (left), one with the RBPF (middle), and one with the graph mapper (right). The quality of all maps is lower than the quality of the map depicted in Figure 8. The reason for that is that while building the map in Figure 8, we also used the satellite image data which was not available for the algorithms under evaluation.

Based on the error plots in Figure 11 as well as the overall score depicted in the tables, we can see that graph mapping outperforms the RBPF and scan matching. The RBPF was not able to close the large loop and therefore performed similar to scan matching. However, note that in most parts of the map, the results of the scan matcher and RBPF are comparable to the one of graph mapping. Significant differences can be observed in the areas labeled as 1 and 3. Here, the two approaches fail to build a consistent map which is the reason for the significantly higher overall error.

In the area labeled as 4, the results of all algorithms yield matching errors. In that area, the robot makes a 180 degree turn and looks towards the open park area where almost no structure that would allow for pose correction is visible. Therefore, none of the tested algorithms was able to build a perfect map here.

Note that for this dataset, we present two alternative sets of relations. One using only local relations based on the sensor range. In addition, we provide a set where the relations are generated for pairs of randomly sampled poses. This set should be used if global consistency is desired. A comparison between the two data sets can be seen in Figure 12. The histograms count relations based by the difference in the time indices of the connected poses. As can be seen from the left image, using local relations based on the sensor range leads to a peaked histogram, since the relations only cover a small time frame. Additional minor peaks occur if the robot re-visits a region. In contrast, the set of relations used to evaluate the global

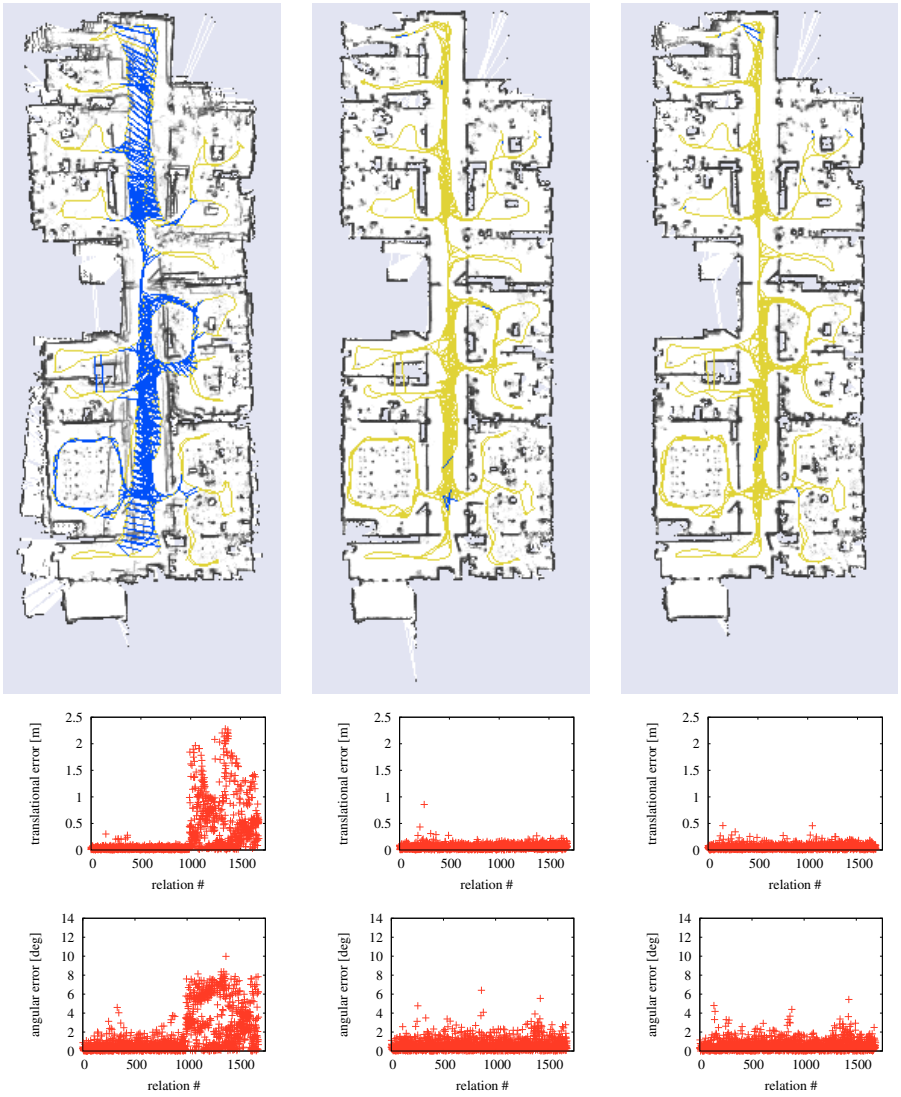


Fig. 10 This figure shows the Freiburg Indoor Building 079 dataset. Each column reports the results of one approach. Left: scan-matching, middle: RBPF and right a graph based algorithm. Within each column, the top image shows the map, the middle plot is the translational error and the bottom one is the rotational error.

consistency of the map is less peaked. Here, the relations uniformly sub-sample all available pairwise combinations of robot poses.

8.4.1 Utilizing Additional Relations

To illustrate that it is possible to incorporate additional relations as claimed in Section 4.3, we added in a further experiment the satellite image data which was used to obtain the close-to-true pose information for the Freiburg hospital. These additional relations favor

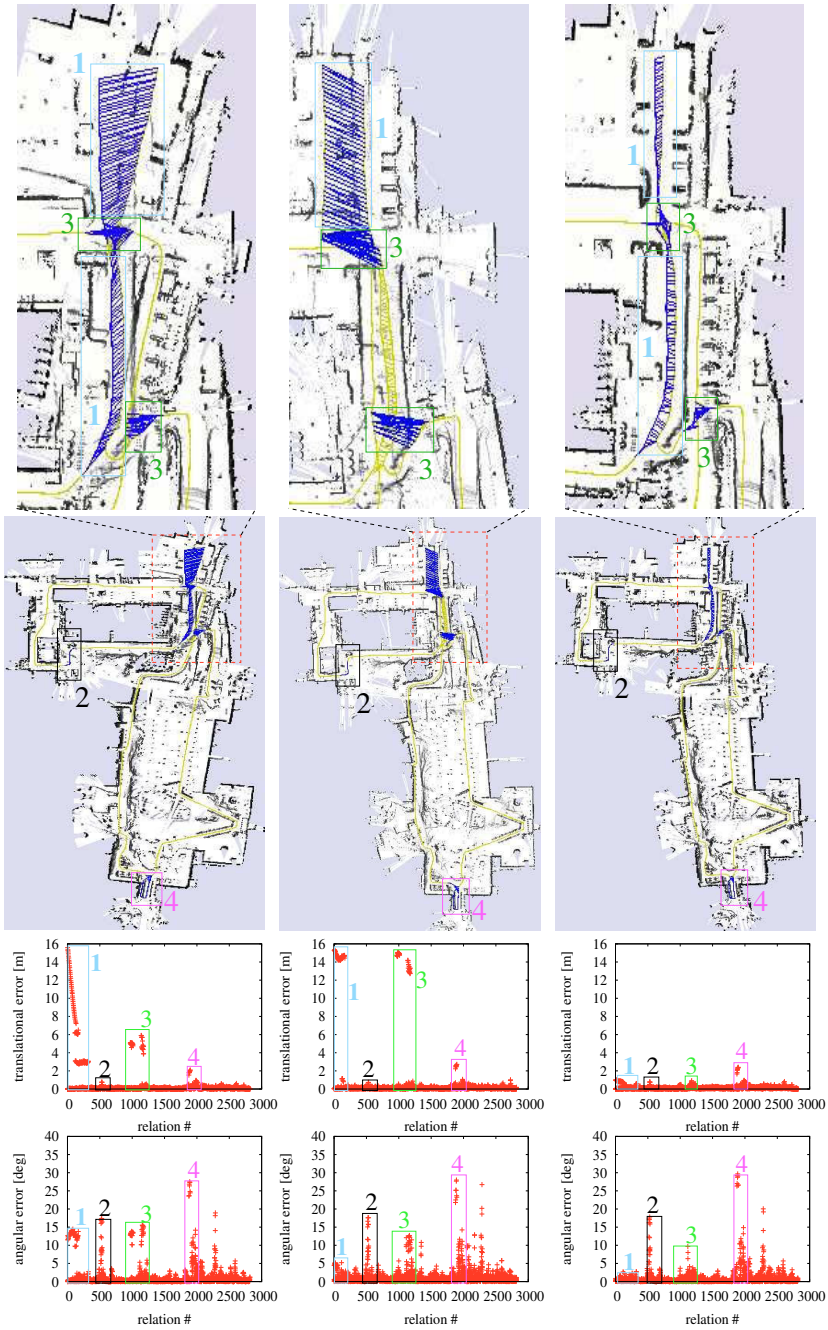


Fig. 11 Maps and error plots of the Freiburg University Hospital. Each column reports the results of one approach. Left: scan-matching, middle: RBPF and right a graph based algorithm. The second row depicts the created maps. The first row shows close-ups of areas in these maps. The error plots in the middle are regarding translation and on the bottom regarding rotation. There are corresponding areas marked in the plots and the maps.

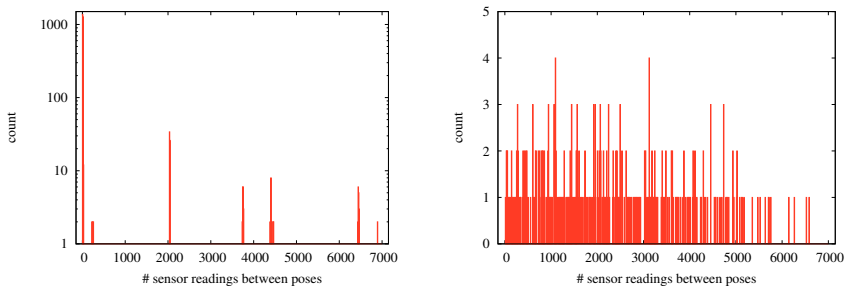


Fig. 12 Comparison of local relations with global relations based on their histograms. The abscissa shows the number of sensor readings between the two positions in a relation (a bin size of 10 was chosen). On the left side the histogram for the local relation set is shown, while the right side displays the global relations.

approaches that are able to generate global consistency as it is desired for robots that, for example, build blueprints.

The resulting scores for such a setting are given in the last rows of Table 1 and Table 2, respectively. As expected, the error in case of the evaluation including global relations is higher.

8.5 Summary of the Experiments

Our evaluation illustrates that the proposed metric provides a ranking of the results of mapping algorithms that is likely to be compatible with a ranking obtained from visual inspection by humans. Inconsistencies yield increased error scores since in the wrongly mapped areas the relations obtained from manual matching are not met. By visualizing the error of each relation as done in the plots in this section, one can identify regions in which algorithms fail and we believe that this helps to understand where and why different approaches have problems to build accurate maps.

We furthermore encourage authors to evaluate their algorithms based on multiple datasets and not just using a single in order to illustrate the generality of the method and not being optimized for a single dataset.

9 Conclusion

In this paper, we have presented a framework for analyzing the results of SLAM approaches that allows for creating objective benchmarks. We proposed a metric for measuring the error of a SLAM system based on the corrected trajectory. Our metric uses only relative relations between poses and does not rely on a global reference frame. This overcomes serious shortcomings of approaches using a global reference frame to compute the error. The metric even allows for comparing SLAM approaches that use different estimation techniques or different sensor modalities.

In addition to the proposed metric, we provide robotic datasets together with relative relations between poses for benchmarking. These relations have been obtained by manually matching observations and yield a high matching accuracy. We present relations for self-recorded datasets with laser range finder data as well as for a set of log-files that are frequently used in the SLAM community to evaluate approaches. In addition, we provide

an error analysis for three mapping systems including two modern laser-based SLAM approaches, namely a graph-based approach as well as system based on a Rao-Blackwellized particle filter. We believe that our results are a valuable benchmark for SLAM researchers since we provide a framework that allows for objectively and comparably easy analyzing the results of SLAM systems.

Acknowledgments

This work has partly been supported by the DFG under contract number SFB/TR-8 and the European Commission under contract numbers FP6-2005-IST-6-RAWSEEDS, FP7-231888-EUROPA, and FP6-IST-045388-INDIGO. The authors gratefully thank Mike Bosse, Patrick Beeson, and Dirk Haehnel for providing the MIT Killian Court, the ACES, and the Intel Research Lab datasets.

References

[Amigoni *et al.*, 2007] F. Amigoni, S. Gasparini, and M. Gini. Good experimental methodologies for robotic mapping: A proposal. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[Balaguer *et al.*, 2007] B. Balaguer, S. Carpin, and S. Balakirsky. Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems. In *IROS 2007 Workshop*, 2007.

[Bar-Shalom *et al.*, 2001] Y Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with Application to Tracking and Navigation*. Jonh Wiley and Sons, 2001.

[Bonarini *et al.*, 2006] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos. Rawseeds a project on SLAM benchmarking. In *Proceedings of the IROS'06 Workshop on Benchmarks in Robotics Research*, 2006. available online at http://www.robot.uji.es/EURON/pdfs/Lecture Notes IROS06.pdf.

[Bosse *et al.*, 2003] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1899–1906, Taipei, Taiwan, 2003.

[Burgard *et al.*, 2009] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós. A comparison of slam algorithms based on a graph of relations. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009. To appear.

[Canny, 1986] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[Censi, 2006] A. Censi. Scan matching in a probabilistic framework. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2291–2296, 2006.

[Darpa, 2007] Darpa. Darpa Urban Challenge, 2007. http://www.darpa.mil/grandchallenge/.

[Dellaert *et al.*, 1998] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[Dellaert, 2005] F. Dellaert. Square Root SAM. In *Proc. of Robotics: Science and Systems (RSS)*, pages 177–184, Cambridge, MA, USA, 2005.

[Dissanayake *et al.*, 2000] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1009–1014, 2000.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordan, editors. *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[Duckett *et al.*, 2002] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287 – 300, 2002.

[EPFL and IROS, 2002] EPFL and IROS. Cleaning Robot Contest, 2002. http://robotika.cz/competitions/cleaning2002/en.

[ESA, 2008] ESA. Lunar robotics challenge, 2008. http://www.esa.int/esaCP/SEM4GKRTKMF_index_0.html.

[Estrada *et al.*, 2005] C. Estrada, J. Neira, and J.D. Tardós. Hierachical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[Eustice *et al.*, 2005a] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, 2005.

[Eustice *et al.*, 2005b] R. Eustice, M. Walter, and J.J. Leonard. Sparse extended information filters: Insights into sparsification. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 641–648, Edmonton, Cananda, 2005.

[Frese *et al.*, 2005] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[Frese, 2006] U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[Frese, 2008] U. Frese. Dlr spatial cognition data set. http://www.informatik.uni-bremen.de/agebv/en/DlrSpatialCognitionDataSet, 2008.

[Früh and Zakhor, 2004] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60:5–24, 2004.

[Grisetti *et al.*, 2007a] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[Grisetti *et al.*, 2007b] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[Grisetti *et al.*, 2007c] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[Gutmann and Konolige, 1999] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1999.

[Hähnel *et al.*, 2003] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[Hermosillo *et al.*, 2003] J. Hermosillo, C. Pradalier, S. Sekhavat, C. Laugier, and G. Baille. Towards motion autonomy of a bi-steerable car: Experimental issues from map-building to trajectory execution. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[Hoover *et al.*, 1996] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. K. Bowyer, D. W. Eggert, A. W. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):673–689, 1996.

[Howard and Roy, 2003] A. Howard and N. Roy. Radish: The robotics data set repository, standard data sets for the robotics community, 2003. http://radish.sourceforge.net/.

[Julier *et al.*, 1995] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, 1995.

[Kaess *et al.*, 2007] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[Kümmerle *et al.*, 2009] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[Leonard and Durrant-Whyte, 1991] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[Lu and Milios, 1994] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)*, pages 935–938, 1994.

[Lu and Milios, 1997] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[Montemerlo *et al.*, 2003] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[Nüchter *et al.*, 2005] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[Olson *et al.*, 2006] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[Olson, 2008] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.

[Pfaff *et al.*, 2006] P. Pfaff, W. Burgard, and D. Fox. Robust monte-carlo localization using adaptive likelihood models. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 181–194. Springer-Verlag Berlin Heidelberg, Germany, 2006.

[Ranganathan *et al.*, 2007] A. Ranganathan, M. Kaess, and F. Dellaert. Loopy sam. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[RoboCup Federation, 2009] RoboCup Federation. RoboCup Competitions, 2009. http://www.robocup.org.

[Scharstein and Szeliski, 2002] D. Scharstein and R. Szeliski. Middlebury stereo vision page, 2002. http://www. middlebury. edu/stereo.

[Smith and Cheeseman, 1986] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[Smith *et al.*, 1990] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[Stachniss *et al.*, 2007a] C. Stachniss, U. Frese, and G. Grisetti. OpenSLAM.org – give your algorithm to the community. http://www.openslam.org, 2007.

[Stachniss *et al.*, 2007b] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard. Evaluation of gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[Symeo GmbH, 2008] Symeo GmbH. http://www.symeo.de, 2008.

[Thrun and colleagues, 2006] S. Thrun and colleagues. Winning the darpa grand challenge. *Journal on Field Robotics*, 2006.

[Thrun *et al.*, 2004] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7/8):693–716, 2004.

[Thrun *et al.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[Thrun, 2001] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[Torralba *et al.*, 2007] A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: the open annotation tool, 2007. http://labelme.csail.mit.edu/.

[Uhlmann, 1995] J. Uhlmann. *Dynamic Map Building and Localization: New Theoretical Foundations*. PhD thesis, University of Oxford, 1995.

[Wulf *et al.*, 2008] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Benchmarking urban six-degree-of-freedom simultaneous localization and mapping. *Journal of Field Robotics*, 25(3):148–163, 2008.

[Yguel *et al.*, 2007] M. Yguel, C.T.M. Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[J4] C. Stachniss, C. Plagemann, and A.J. Lilienthal. Gas distribution modeling using sparse gaussian process mixtures. *Autonomous Robots*, 26:187ff, 2009.

Autonomous Robots manuscript No.
(will be inserted by the editor)

Learning Gas Distribution Models using Sparse Gaussian Process Mixtures

Cyrill Stachniss · Christian Plagemann · Achim J. Lilienthal

Received: date / Accepted: date

Abstract In this paper, we consider the problem of learning two-dimensional spatial models of gas distributions. To build models of gas distributions that can be used to accurately predict the gas concentration at query locations is a challenging task due to the chaotic nature of gas dispersal. We formulate this task as a regression problem. To deal with the specific properties of gas distributions, we propose a sparse Gaussian process mixture model, which allows us to accurately represent the smooth background signal and the areas with patches of high concentrations. We furthermore integrate the sparsification of the training data into an EM procedure that we apply for learning the mixture components and the gating function. Our approach has been implemented and tested using datasets recorded with a real mobile robot equipped with an electronic nose. The experiments demonstrate that our technique is well-suited for predicting gas concentrations at new query locations and that it outperforms alternative and previously proposed methods in robotics.

Keywords Gas distribution modeling · gas sensing · Gaussian processes · mixture models

C. Stachniss
University of Freiburg, Dept. of Computer Science, Georges Koehler Allee 79, 79110 Freiburg, Germany
Tel.: +49-761-203-8024, Fax: +49-761-203-8007, E-mail: stachnis@informatik.uni-freiburg.de

C. Plagemann
Stanford University, Computer Science Dept., 353 Serra Mall, Stanford, CA 94305-9010, USA
Phone: +1-650-723-9558, Fax: +1-412-725-1449, E-mail: plagemann@stanford.edu

A. J. Lilienthal
University of Örebro, AASS Research Institute, Fakultetsgatan 1, 70182 Örebro, Sweden
Tel.: +46-19-30-3602, Fax: +46-19-30-3463, E-mail: achim@lilienthals.de

1 Introduction

The problem of modeling gas distributions has important applications in industry, science, and every-day life. Mobile robots equipped with gas sensors can be deployed for pollution monitoring in public areas [DustBot, 2008], surveillance of industrial facilities producing harmful gases, or inspection of contaminated areas within rescue missions.

Although humans have a comparably good odor sensor allowing to distinguish between around 10 000 odors, it is hard for us to build spatial representations of sensed gas distributions. Building gas distribution maps is a challenging task in principle due to the chaotic nature of gas dispersal and because only point measurements of gas concentration are available. The complex interaction of gas with its surroundings is dominated by two physical effects. First, on a comparably large timescale, *diffusion* mixes the gas with the surrounding atmosphere achieving a homogeneous mixture of both in the long run. Second, turbulent air flow fragments the gas emanating from a source into intermittent *patches* of high concentration with steep gradients at their edges [Roberts and Webster, 2002]. This chaotic system of localized patches of gas makes the modeling problem a hard one. In addition, gas sensors provide information about a small spatial region only since gas sensor measurements require direct interaction between the sensor surface and the molecules to be analyzed. This makes gas sensing different to perceiving the environment with other popular robotic sensors like laser range finders, with which a larger area can be measured directly.

Fig. 1 illustrates actual gas concentration measurements recorded with a mobile robot along a corridor containing a single gas source. The distribution consists of a rather smooth "background" signal and several peaks, which indicate high gas concentrations. The challenge in gas distribution mapping is to model this background signal while being able to cover also the areas of high concentration and their sharp boundaries.

From a probabilistic point of view, the task of modeling a gas distribution can be described as finding a model that best explains the observations and that is able to accurately predict new ones. A suitable measure for evaluating models and for comparing alternative ones is to consider the *predictive data likelihood* of an *independent test set*. This measure compares each test data point (which is not contained in the training set) with a predictive distribution estimated by the model. For this, it does not require insight into the model internals and it does not depend on any explicit notion of model complexity or the number of model parameters as, for example, the Bayesian Information Criterion (BIC). The predictive data likelihood is therefore the measure of choice for evaluating especially nonparametric models. As a drawback, one needs a sufficiently large amount of data to be able to separate out a test set without risking that the training set becomes too small to capture the sought-after distribution. The gas mapping application comes with an abundance of available data, such that the predictive test set likelihood constitutes a robust measure for model accuracy.

Simple spatial averaging, which represents a straight-forward approach to the modeling problem, disregards the different nature of the background concentration and the peaks resulting from areas of high gas concentrations and, thus, achieves only limited prediction accuracy. On the other hand, precise physical simulation of the gas dynamics in the environment would require immense computational resources as well as precise knowledge about the physical conditions, which is not known in most practical scenarios.

To achieve a balance between model accuracy and efficiency, we treat gas distribution mapping as a supervised regression problem. We derive a solution by means of a sparse mixture model of Gaussian processes [Tresp, 2000] that is able to handle both physical phenomena highlighted above. Formally, we interpret gas sensor measurements obtained from static sensors or from a mobile robot as noisy samples from a time-constant distribution.

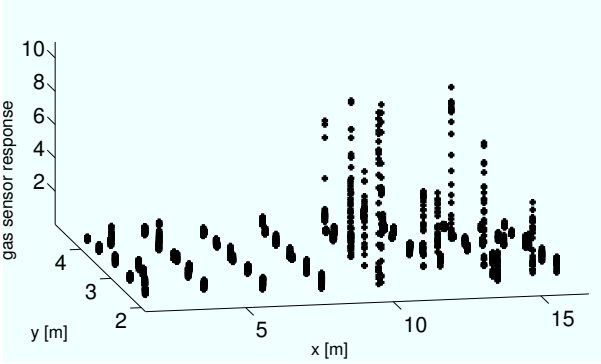


Fig. 1 Gas concentration measurements acquired by a mobile robot in a corridor. The distribution consists of a rather smooth "background" signal and several peaks, which indicate high gas concentrations.

This implies that the gas distribution in fact exhibits a time-constant structure, an assumption that is often made in unventilated and un-populated indoor environments [Wandel *et al.*, 2003].

While existing approaches to gas distribution mapping, such as averaging [Ishida *et al.*, 1998, Purnamadjaja and Russell, 2005, Pyk *et al.*, 2006] or kernel extrapolation [Lilienthal and Duckett, 2004] represent the average concentration per location only, our mixture model actually allows us to do both, computing the mean gas concentration as well as the multi-modal, predictive densities. We further obtain a more accurate estimate of the gas concentration by distinguishing explicitly different components of the distribution, particularly a "background" component where the concentration varies smoothly and a second component that corresponds to the area in which localized patches of gas occur. In a scenario with a constant, uniform airflow, the latter mixture component represents the gas plume [Murlis *et al.*, 1992].

As a by-product, we present a generic algorithm that learns a GP mixture model and at the same time reduces the number of used training data points in order to achieve an efficient representation even for large data sets. We demonstrate in experiments carried out with real mobile robots that our model has a lower mean squared error and a higher data likelihood on test data sets than other existing methods for gas distribution modeling. Thus, it allows to predict gas concentration at query locations more accurately.

This article is organized as follows. After introducing our mixture model in Sec. 2, we propose our method for learning the model components from data and for achieving a sparse approximation in Sec. 3. We then present experimental results involving real mobile platforms in Sec. 4 and discuss related work in Sec. 5.

2 A Mixture Model for Gas Distributions

The general gas distribution mapping problem given a set of concentration measurements $y_{1:n}$ acquired at locations $\mathbf{x}_{1:n}$, is to learn a predictive model $p(y_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, y_{1:n})$ for gas concentrations y_* at a query location $\mathbf{x}_*$. We approach this problem in a nonparametric way, i.e., not assuming a parametric form of the underlying function $f(\cdot)$ in $y = f(\mathbf{x}) + \epsilon$, using the Gaussian process model [Rasmussen and Williams, 2006]. In this Bayesian approach to the non-linear regression problem, one places a prior on the space of functions $p(f)$ using

the following definition: A Gaussian process is a collection of random variables, any of which have a joint Gaussian distribution. More formally, if we assume that $\{(\mathbf{x}_i, f_i)\}_{i=1}^n$ with $f_i = f(\mathbf{x}_i)$ are samples from a Gaussian process and define $\mathbf{f} = (f_1, \ldots, f_n)^\top$, we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \,, \quad \boldsymbol{\mu} \in \mathbb{R}^n, \mathbf{K} \in \mathbb{R}^{n \times n} \,. \tag{1}$$

For simplicity of notation, we can assume $\boldsymbol{\mu} = \mathbf{0}$, since the expectation is a linear operator and, thus, for any deterministic mean function $m(\mathbf{x})$, the Gaussian process over $f'(\mathbf{x}) := f(\mathbf{x}) - m(\mathbf{x})$ has zero mean.

The interesting part of the model is indeed the covariance matrix $\mathbf{K}$. It is specified by $[\mathbf{K}]_{ij} := \mathrm{cov}(f_i, f_j) = k(\mathbf{x}_i, \mathbf{x}_j)$ using a *covariance function* k which defines the covariance of any two function values $\{f_i, f_j\}$ sampled from the process given their input vectors $\{\mathbf{x}_i, \mathbf{x}_j\}$ as parameters. Intuitively, the covariance function specifies how *similar* two function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ are depending only on the corresponding inputs. The standard choice for k is the squared exponential covariance function

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \, \exp\left(-\frac{1}{2}\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{\ell^2}\right) \,, \tag{2}$$

where the so-called *length-scale* parameter ℓ defines the global smoothness of the function f and σ_f^2 denotes the amplitude (or signal variance) parameter. These parameters, along with the global noise variance σ_n^2 that is assumed for the noise component, are known as the *hyperparameters* of the process. They are denoted as $\boldsymbol{\theta} = \langle \sigma_f, \ell, \sigma_n \rangle$.

Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of training data where $\mathbf{x}_i \in \mathbb{R}^d$ are the inputs and $y_i \in \mathbb{R}$ the targets, the goal in regression is to predict target values $y_* \in \mathbb{R}$ at a new input point $\mathbf{x}_*$. Let $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]^\top$ be the $n \times d$ matrix of the inputs and $\mathbf{X}_*$ be defined analogously for multiple test data points. In the GP model, any finite set of samples is jointly Gaussian distributed

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right), \tag{3}$$

where $k(\mathbf{X}, \mathbf{X})$ refers to the covariance matrix built by evaluating the covariance function $k(\cdot, \cdot)$ for all pairs of all row vectors $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ of $\mathbf{X}$. To make predictions at $\mathbf{X}_*$, we obtain the predictive mean

$$\bar{f}(\mathbf{X}_*) := \mathbb{E}[f(\mathbf{X}_*)] = k(\mathbf{X}_*, \mathbf{X}) \, \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} \mathbf{y} \tag{4}$$

and the (noise-free) predictive variance

$$\mathbb{V}[f(\mathbf{X}_*)] = k(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}_*, \mathbf{X}) \, \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, \mathbf{X}_*) \,, \tag{5}$$

where $\mathbf{I}$ is the identity matrix. The corresponding (noisy) predictive variance for an observation $\mathbf{y}_*$ can be obtained by adding the noise term σ_n^2 to the individual components of $\mathbb{V}[f(\mathbf{X}_*)]$.

The standard GP model recapitulated above has two major limitations in our problem domain. First, the computational complexity is high, since to compute the predictive variance given in Eq. (5), one needs to invert the matrix $k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I$. This introduces a complexity of $\mathcal{O}(n^3)$ where n is the number of training examples. As a result, an important

issue for GP-based solutions to practical problems is the reduction of this complexity. This can, as we will show in Sec. 3, be achieved by artificially limiting the training data set in a way that introduces small loss in the data likelihood while at the same time minimizing the runtime. As a second limitation, the standard GP model generates a *uni-modal* distribution per input location $\mathbf{x}$. This assumption hardly fits our application domain in which a relatively smooth "background" signal is typically *mixed* with medium- and high-concentration "packets" of gas. In the following, we address this issue by deriving a *mixture model* of Gaussian processes.

2.1 Mixtures of Gaussian Process Models

The GP mixture model [Tresp, 2000] constitutes a locally weighted sum of several Gaussian process models. For simplicity of notation, we consider without loss of generality the case of single predictions only ($\mathbf{x}_*$ instead of $\mathbf{X}_*$). Let $\{\mathcal{GP}_1, \ldots, \mathcal{GP}_m\}$ be a set of m Gaussian processes representing the individual mixture components. Let $P(z(\mathbf{x}_*) = i)$ be the probability that $\mathbf{x}_*$ is associated with the i-th component of the mixture. Let $\bar{f}_i(\mathbf{x}_*)$ be the mean prediction of $\mathcal{GP}_i$ at $\mathbf{x}_*$. The likelihood of observing y_* is thus given by

$$h(\mathbf{x}_*) := p(y_* \mid \mathbf{x}_*) = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i)\mathcal{N}_i(y_*; \mathbf{x}_*)\,, \tag{6}$$

where we define $\mathcal{N}_i(y; \mathbf{x})$ as the Gaussian density function with mean $\bar{f}_i(\mathbf{x})$ and variance $\mathbb{V}[f_i(\mathbf{x})] + \sigma_n^2$ evaluated at y. One can sample from such a mixture by first sampling the mixture component according to $P(z(\mathbf{x}_*) = i)$ and then sampling from the corresponding Gaussian. For some applications such as information-driven exploration missions, it is practical to estimate the mean and variance for this multi-modal model. The mean $\mathbb{E}[h(\mathbf{x}_*)]$ of the mixture model is given by

$$\bar{h}(\mathbf{x}_*) := \mathbb{E}[h(\mathbf{x}_*)] = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i)\bar{f}_i(\mathbf{x}_*) \tag{7}$$

and the corresponding variance is computed as

$$\mathbb{V}[h(\mathbf{x}_*)] = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i)\left(\mathbb{V}[f_i(\mathbf{x}_*)] + (\bar{f}_i(\mathbf{x}_*) - \bar{h}(\mathbf{x}_*))^2\right)\,. \tag{8}$$

2.2 The Choice of the Covariance Function

The covariance function in a Gaussian Process as well as in our mixture model is a crucial component as it encodes knowledge about the function to approximate. It specifies the dependency between two function values $f(\mathbf{x}_i)$, $f(\mathbf{x}_j)$ and this dependency is computed only based on the corresponding inputs.

The standard choice for the covariance function is the squared exponential (SE) shown in Eq. (2), however, there are several other possibilities to define a covariance function. In this paper, we also analyze how the choice of the covariance function affects the quality of the gas distribution model. In detail, we analyze the squared exponential and two instances of the Matérn covariance function.

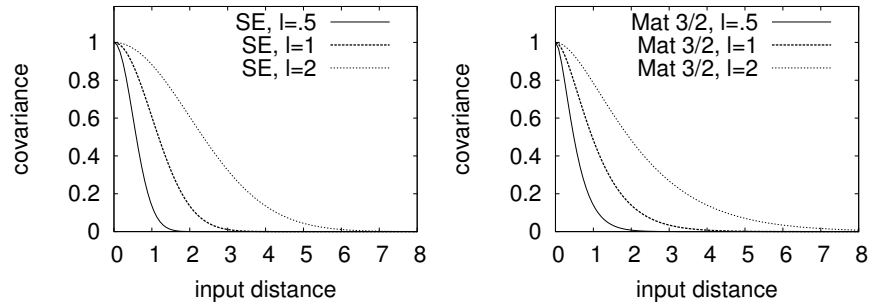


Fig. 2 Example plots of the squared exponential covariance function (left) and the Matérn 3/2 covariance function (right), each plotted for varying hyperparameters.

In case of the Matérn covariance function, we consider the so-called "Matérn 3/2" and "Matérn 5/2" functions among the class of Matérn kernels. They are given by

$$k_{Mat\ 3/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{3}||\mathbf{x}_i, \mathbf{x}_j||}{l}\right) \exp\left(-\frac{\sqrt{3}||\mathbf{x}_i, \mathbf{x}_j||}{l}\right) \tag{9}$$

and

$$k_{Mat\ 5/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{5}||\mathbf{x}_i, \mathbf{x}_j||}{l} + \frac{5||\mathbf{x}_i, \mathbf{x}_j||^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}||\mathbf{x}_i, \mathbf{x}_j||}{l}\right). \tag{10}$$

As for the case of the SE covariance function, the parameter ℓ is the length-scale that defines the global smoothness of the function f and σ_f^2 denotes the amplitude (or signal variance) parameter.

Fig. 2 shows 2d-plots of these covariance functions illustrating the assumed dependency between data points of the function to model depending only on the distance of the inputs.

When comparing the properties of the individual covariance functions, the SE function is a rather smooth one and, therefore, leads to a comparably strong smoothing of the function approximation. This, however, might contradict the nature of gas distribution as well as other physical phenomena. Therefore, we also consider the Matérn covariance function that typically produces rougher estimates and thus might be better suited for the problem studied in this paper. Among the two Matérn kernels used in this paper, the Matérn 5/2 is smoother than the Matérn 3/2.

3 Learning the Model from Data

Given a training set $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^n$ of gas concentration measurements y_j and the corresponding sensing locations $\mathbf{x}_j$, the task is to jointly learn the assignment $z(\mathbf{x}_j)$ of data points to mixture components and, given this assignment, the individual regression models $\mathcal{GP}_i$. Tresp [2000] describes an approach based on Expectation Maximization (EM) for solving this task. We take his approach, but also seek to minimize the number of training data points to achieve a computationally tractable model even for large training data sets $\mathcal{D}$. This is of major importance in our application, since typical gas concentration data sets easily exceed $n = 1\,000$ data points and the standard GP model (see Sec. 2) is of cubic complexity $\mathcal{O}(n^3)$.

Different solutions have been proposed for lowering this upper bound, such as dividing the input space into different regions and solving these problems individually or by deriving sparse approximations for the whole space. Sparse GPs [Smola and Bartlett, 2000, Snelson and Ghahramani, 2006a] use a reduced set of inputs to approximate the full GP model. This new set can be either a subset of the original inputs [Smola and Bartlett, 2000] or a set of r new pseudo-inputs [Snelson and Ghahramani, 2006a] which are obtained using an optimization procedure. This reduces the complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nr^2)$ with $r \ll n$, which in practice results in a nearly linear complexity.

We apply a method similar to sparse GPs and select a subset of the original inputs. In the remainder of this section, we describe a greedy forward-selection algorithm integrated into the EM-learning procedure which achieves a sparse mixture model by selecting a subset of the original inputs. while also maximizing the cross validation data likelihood.

3.1 Initializing the Mixture Components

In a first step, we subsample n_1 data points and learn a standard GP for this set (including the optimization of the hyperparameters). This model $\mathcal{GP}_1$ constitutes the first mixture component. To improve the estimate of gas concentration in areas that are poorly modeled by this initial model, we learn an "error model", termed $\mathcal{GP}_\Delta$, that captures the absolute differences between a set of target values and the predictions of $\mathcal{GP}_1$. We then sample points according to $\mathcal{GP}_\Delta$ and use them to initialize the next mixture component. In this way, the new mixture is initialized with the data points that are poorly approximated by the first one and a hyperparameter optimization is performed. This process is repeated until the desired number of model components is reached. For typical gas modeling scenarios, we found that two mixture components are sufficient to achieve good results. In our experiments, the converged mixture models nicely reflect the bimodal nature of gas distributions, having one smooth "background" component and a layer of locally concentrated structures.

It should be mentioned, that depending on the actual data, the error model ("error GP") might have to be evaluated at all $n - n_1$ inputs which would lead to large computational overhead. Instead, we actually average multiple spatially close measurements and evaluate only at uniformly sampled locations. This is clearly an approximation but only used for the error model of our approach. We, however, did not encounter problems using this strategy which is actually used only for initialization.

3.2 Iterative Learning via Expectation-Maximization

The Expectation Maximization (EM) algorithm can be used to obtain a maximum likelihood estimate when hidden and observable variables need to be estimated. It consists of two steps, the so-called estimation (E) step and the maximization (M) step which are executed alternately.

In the E-step, we estimate the probability $P(z(\mathbf{x}_j) = i)$ that data point j corresponds to model component i. This is done by computing the likelihood of each data point for the model components individually. Thus, the new $P(z(\mathbf{x}_j) = i)$ is computed given the previous estimate as

$$P(z(\mathbf{x}_j) = i) \leftarrow \frac{P(z(\mathbf{x}_j) = i)\mathcal{N}_i(y_j; \mathbf{x}_j)}{\sum_{k=1}^{m} P(z(\mathbf{x}_j) = k)\mathcal{N}_k(y_j; \mathbf{x}_j)} \; . \tag{11}$$

In the M-step, we update the components of our mixture model. This is achieved by integrating the probability that a data point belongs to a model component into the individual GP learning steps (see also [Tresp, 2000]). This is achieved by modifying Eq. (4) to

$$\bar{f}_i(\mathbf{X}_*) = k(\mathbf{X}_*, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \mathbf{\Psi}^i \right]^{-1} \mathbf{y} \,, \tag{12}$$

where $\mathbf{\Psi}^i$ is a matrix with

$$[\mathbf{\Psi}^i]_{jj} = \frac{\sigma_n^2}{P(z(\mathbf{x}_j) = i)} \tag{13}$$

and zeros in the off-diagonal elements. Eq. (5) is updated accordingly. The matrix $\mathbf{\Psi}^i$ allows us to consider the probabilities that the individual inputs belong to the corresponding components. Figuratively speaking, the contribution of an unlikely data point to a model is reduced by increasing the data point specific noise term. If the assignment probability, on the other hand, is one, only σ_n^2 remains and the point is fully included as in the standard GP model.

Learning a GP model also involves the estimation of its hyperparameters $\boldsymbol{\theta} = \{\sigma_f, \ell, \sigma_n\}$. To estimate them for $\mathcal{GP}_i$, we first apply a variant of the hyperparameter heuristic used by Snelson and Ghahramani [2006a] in their open-source implementation. We extended it to incorporate the correspondence probability $P(z(\mathbf{x}_k) = i)$ into this initial guess

$$\ell \leftarrow \max_{\mathbf{x}_j} \; P(z(\mathbf{x}_j) = i) \; ||\mathbf{x}_j - \bar{\mathbf{x}}|| \tag{14}$$

$$\sigma_f^2 \leftarrow \frac{\sum_{j=1}^n P(z(\mathbf{x}_j) = i) \; (y_j - \mathbb{E}[y])^2}{\sum_{j=1}^n P(z(\mathbf{x}_j) = i)} \tag{15}$$

$$\sigma_n^2 \leftarrow \frac{1}{4}\sigma_f^2 \,, \tag{16}$$

where $\bar{\mathbf{x}}$ refers to the weighted mean of the inputs, each $\mathbf{x}_j$ having a weight of $P(z(\mathbf{x}_j) = i)$. To optimize the hyperparameters based on this initial estimate, one could apply, for example, Rasmussen's conjugate-gradient–based approach [Rasmussen, 2006]. In our experiments, however, this approach lead to overfitting problems and we therefore resorted to cross validation-based optimization. Concretely, we repeatedly sample hyperparameters and evaluate the model accuracy according to Sec. 3.2 on a separate validation set. As a hyperparameter sampling strategy, we draw in each even iteration of this sampling new hyperparameters from an uninformed prior and in each odd iteration, we improve the current best parameters $\boldsymbol{\theta}'$ by sampling from a Gaussian with mean $\boldsymbol{\theta}'$. The standard deviation of that Gaussian is decreased with the iteration number.

In our experiments, this rather straight forward strategy converged quickly after a few iterations (approx. 50 iterations, see Fig. 11 for an example). Note that there are more sophisticated strategies, for example simulated annealing, that can be used instead. However, we selected a simpler approach since it provided satisfactory results and can be implemented with five lines of code.

3.3 Learning the Gating Function

In our mixture model, the gating function defines for each data point the probability of being assigned to the individual mixture components. The EM algorithm learns the assignment

probabilities for the used training inputs $\mathbf{x}_j$, maximizing the cross validation data likelihood. To generalize these assignments to the whole input space (to form a proper gating *function*), we place another GP prior on the gating variables. Concretely, we learn a gating GP for each component i that uses the $\mathbf{x}_j$ as inputs and the $z(\mathbf{x}_j)$ obtained from the EM algorithm as targets. Let $\bar{f}_i^z(\mathbf{x})$ be the prediction of z for $\mathcal{GP}_i$. Given this set of m GPs, we can compute the correspondence probability for a new test point $\mathbf{x}_*$ as

$$P(z(\mathbf{x}_*) = i) = \frac{\exp(\bar{f}_i^z(\mathbf{x}_*))}{\sum_{j=1}^{m} \exp(\bar{f}_j^z(\mathbf{x}_*))} \,. \tag{17}$$

3.4 Summary

This section briefly summarizes our approach for learning the GP mixture model. First, we initialize the mixture components which are individual GPs. This done by randomly sampling data point for the first component. Then, an error GP is learned to estimate the prediction error. The data points for the subsequent component are then sampled based on the error GP. Second, the we apply the expectation maximization algorithm to optimize the mixture components and to estimate the hidden mixture/class assignment variables. In each iteration of the EM, the hyperparameters for the mixture components are iteratively optimized. Finally, the gating function is learned using again the GP framework. The gating function models the class assignments for the whole input space. Learning is done based on separated training and test sets.

3.5 Illustrating Example

To visualize our approach, we now give a simple, one-dimensional example. The left diagram of Fig. 3 shows simulated data points, of which most were sampled uniformly from the interval $[2 : 2.5]$ and some are distributed with a larger spread at two distinct locations. The same diagram also shows a standard GP model learned on this set, which is not able to explain the data well. The right diagram of the figure shows $\mathcal{GP}_\Delta$, i.e. the resulting error model, which characterizes the local deviations of the model predictions from the data points. Based on this model, a second mixture component is initialized and used as input to the EM algorithm.

The individual diagrams in Fig. 4 illustrate the iterations of the EM algorithm (to be read from left to right and from top to bottom). They depict the two components of the mixture model. The learned gating function after convergence of the algorithm is depicted in the left diagram of Fig. 5. The right diagram in the same figure gives the final GP mixture model. It is clearly visible that the mixture model better represents this data set than the standard GP model, which assumes a smooth, uni-modal process (see the left diagram of Fig. 3).

4 Experimental Results

We carried out pollution monitoring experiments in a real-world setting, in which a mobile robot followed a predefined sweeping trajectory covering the area of interest. Along its path, the robot was stopped for several seconds, 10 s (outdoors) and 30 s (indoors), at predefined points to acquire measurements. The spacing between the grid points was set to values

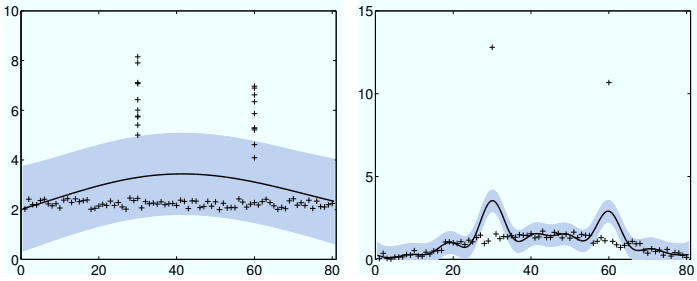


Fig. 3 Left: The standard GP used to initialize the first mixture component. Right: The error GP used to initialize the next mixture component.

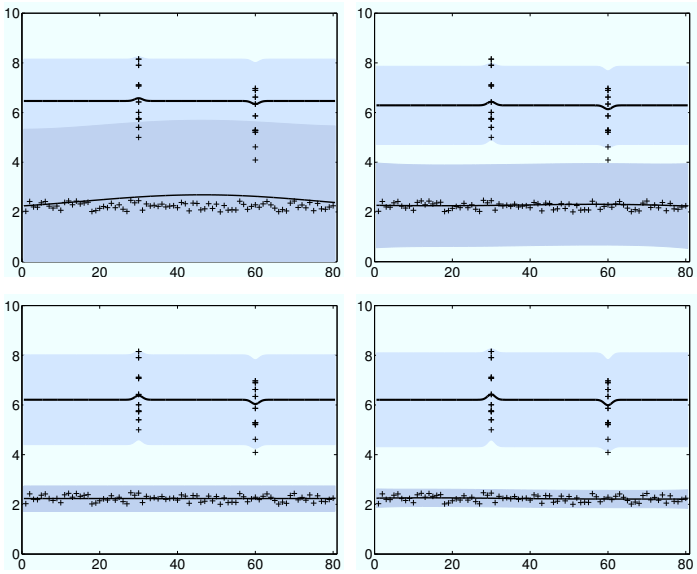


Fig. 4 Components during different iterations of the EM algorithm.

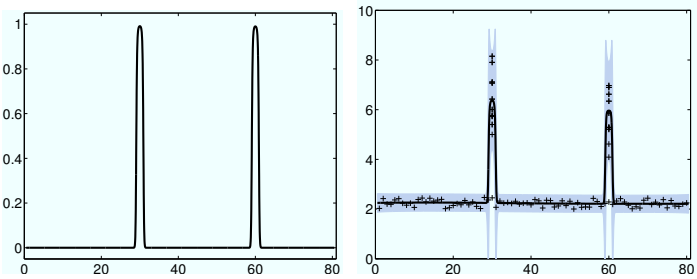


Fig. 5 Left: The learned gating function. Right: Resulting distribution of the GP mixture model.

Dataset	GP (SE)	GP (Mat3/2)	GP (Mat5/2)	GPM avg (SE)	GPM avg (Mat3/2)	GPM avg (Mat5/2)
3-rooms	-1.22	-1.25	-1.27	-1.50	-1.51	-1.52
corridor	-0.98	-1.06	-0.98	-1.58	-1.58	-1.60
outdoor	-1.11	-1.17	-1.22	-1.72	-1.88	-1.85

Table 1 Average negative log likelihoods of test data points for different approaches. The results of the comparison between the GP and the GP mixture model with corresponding covariance functions shown in this table differ significantly (10 repetitions, $\alpha = 5\%$).

Dataset	GP	GPM avg	GPM
3-rooms	-1.22	-1.50	-1.54
corridor	-0.98	-1.58	-1.60
outdoor	-1.11	-1.72	-1.80

Table 2 Comparison between standard GP (GP), the GP mixture model with averaging (GPM avg) according to Eq. (8) and Eq. (7), and the GP mixture model with multi-modal estimates (GPM) based on 10 repetitions (here using the SE covariance function).

between 0.5 m to 2.0 m depending on the topology of the available space, see Fig. 6. In the experiments, the sweeping motion was performed twice in opposite directions which allows us to use the second visit for evaluating our predictions. Due to the slow response of the gas sensors and in order to avoid disturbance to the gas distribution created by the robot itself, the robot was driven at a maximum speed of 5 cm/s in between the stops. The gas source was a small cup filled with ethanol and in the experiments, the robot approached the cup up to a distance of approximatively 0.1 m.

The robot was equipped with a SICK laser range scanner used for pose correction, with an electronic nose, and an anemometer. The electronic nose is a Figaro TGS 2620 gas sensor enclosed in an aluminum tube. This tube was mounted horizontally at the front side of the robot. The electronic nose is actively ventilated through a fan that creates a constant airflow towards the gas sensor. This lowers the effect of external airflow and the movement of the robot on the sensor response.

Note that in this work, we concentrate only on the gas concentration measurements and do not consider the pose uncertainty of the vehicle. One can apply one of the various SLAM systems available to account for the uncertainty in the robot's pose [Frese, 2006, Grisetti *et al.*, 2007, Lilienthal *et al.*, 2007].

4.1 Inspected Environments

Three environments with different properties were selected for the pollution monitoring experiments. The first experiment, termed *3-rooms*, was carried out in an enclosed indoor area that consists of three rooms which are separated by slightly protruding walls in between them. The area covered by the robot is approximately 14 m $\times$ 6 m. There is little exchange of air with the "outer world" in this environment. The gas source was placed in the central room and all three rooms were monitored by the robot. The second location was a part of a *corridor* with open ends and a high ceiling. The area covered by the trajectory of the robot is approximately 14 m $\times$ 2 m. The gas source was placed on the floor in the middle of the investigated corridor segment. Finally, an *outdoor* scenario was considered. Here, the experiments were carried out in an 8 m $\times$ 8 m region that is part of a much bigger open area.

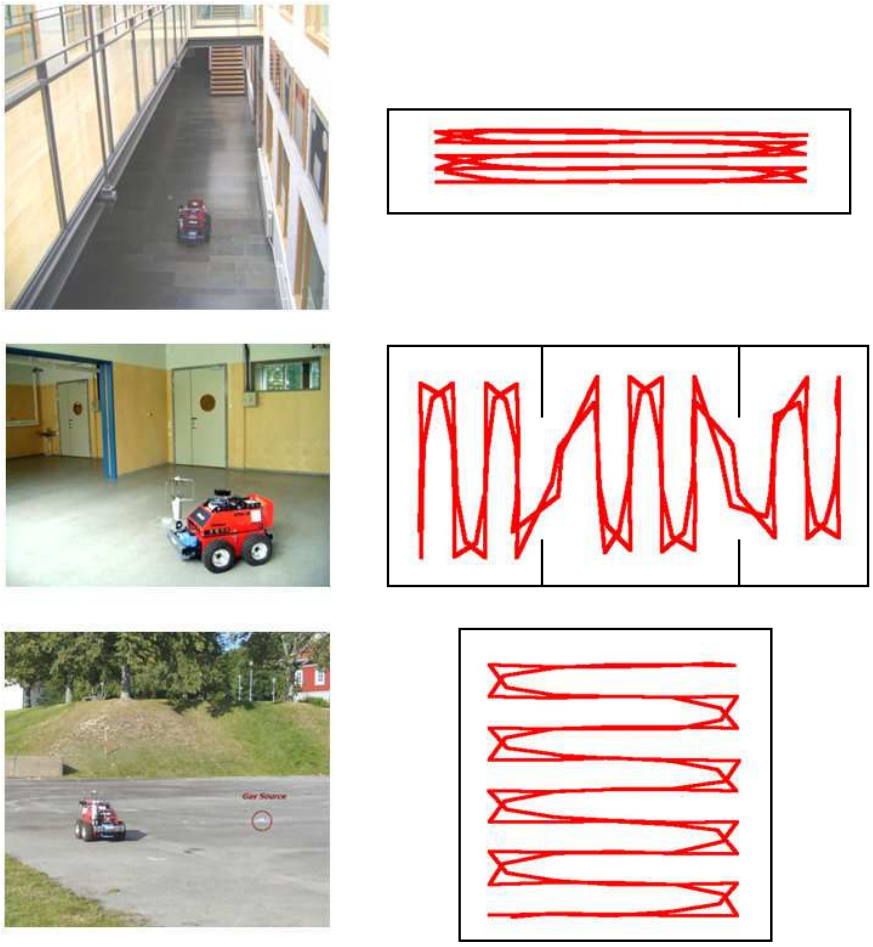


Fig. 6 Pictures of the robot inspecting three different environments as well as the corresponding sweeping trajectories.

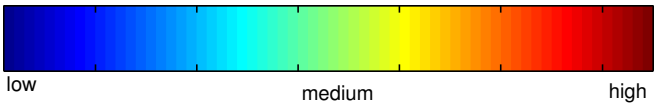
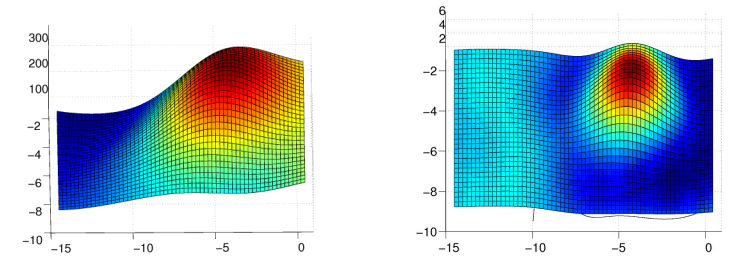


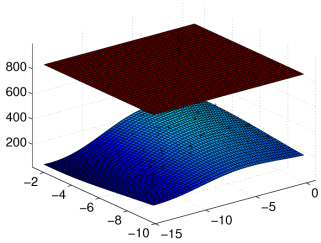
Fig. 7 Color schema for the gas concentrations visualizations (Matlab default). Gas distribution measurements are always normalized between 0 and 1 given the current set of observations used for learning the model.

We used the raw sensor readings in all three environments as training sets and applied our approach to learn the gas distribution models. The robot moved through the environment twice. We used the first run for learning the model and the second one for evaluating it. To benchmark our results, we compare against gas distribution models learned using (a) standard GP regression, (b) a grid-based interpolation approach, and (c) kernel extrapolation. For the Gaussian process regression, we furthermore analyze the influence of different co-

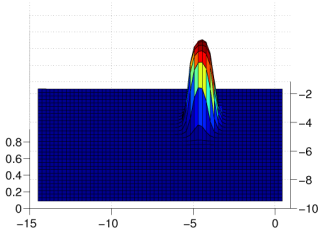


Initial, uni-modal model

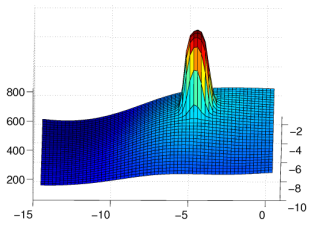
Error model



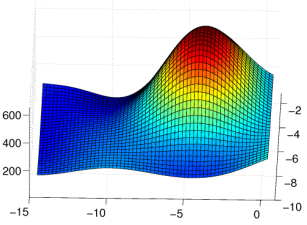
Means of the mixture components



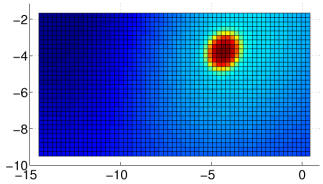
Gating function



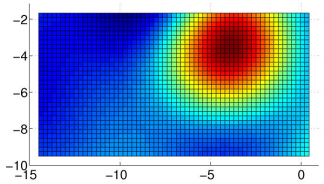
GPM mean (3D view)



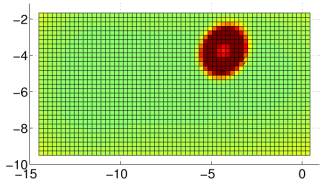
Standard GP mean (3D view)



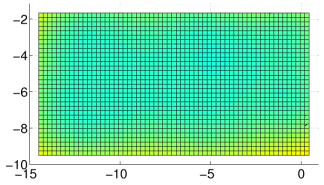
GPM mean (2D view)



Standard GP mean (2D view)



GPM Variance (2D view)



Standard GP variance (2D view)

Fig. 8 The *3-rooms* dataset with one ethanol gas source in the central room. The room structure itself is not visualized here. In all plots, blue represents low, yellow reflects medium, and red refers to high values. See Fig. 7 for the color encoding. The unit of the x- and y-axis is meter.

variance functions in the obtained results. For the visualizations, we always used the default Matlab color scheme depicted also in Fig. 7 and normalized the gas concentration measurements obtained to values between zero and one.

4.2 Evaluation

Fig. 8 shows the learned models for the *3-room* dataset. The left plot in the first row illustrates the mean prediction for the standard GP on the subsampled training set that defines the first mixture component. The right diagram depicts the error GP representing the differences between the initial prediction and a set of observations. Based on the error GP, a new mixture component is initialized and the EM algorithm is carried out. The means of the two mixture components after convergence are shown in the left diagram of the second row and the learned gating function is visualized in the adjacent diagram on the right. The left diagram in the third row shows the mean prediction of the final mixture model. As can be seen, the model consists of a smooth "background" distribution and a peak of gas concentration—close to the gas source—with a sharp boundary. In contrast to this, the standard GP (right diagram in the third row) learned using the same data is overly smooth for this dataset, especially in proximity to the gas source. For both models, the squared exponential covariance function has been used here.

Table 1 summarizes the negative log likelihoods of the test data (second part of the dataset, which was not used for training) given our mixture model (GPM) as well as the standard GP model (GP). As can be seen, our GPM method outperforms the standard GP model in all settings. A t-test on 10 repeated learning runs revealed that these results are significant ($\alpha = 5\%$). Two reasons for the increased model accuracy of GPM w.r.t. standard GPs can be seen in the 2D plots in the last two rows of Fig. 8. First, as already mentioned before, the standard GP overly smoothes the area close to the gas source and, second, its variance estimates around the source are too low (since standard GPs assume a constant noise rate for the whole domain). The table furthermore analyses the results obtained with different covariance functions. The Matérn kernels perform on average slightly better than the squared exponential function. This is probably the case because the Matérn kernels are less smooth which is in line with the nature of the problem addressed in this paper. In Table 2, we provide two likelihoods for our model, the one given in Eq. (6) (called "GPM" in the table) and the one computed based on the averaged prediction specified in Eq. (7) and Eq. (8) (called "GPM avg").

Fig. 9 visualizes the final results for the *corridor* experiment for the GPM model (means of the mixture components in the left diagram and the predictive uncertainty on the right). The raw dataset from this experiment is plotted in Fig. 1. In this experiment, the area of high gas concentration was also mapped comparably accurate by the standard GP, but again the variance close to the area of high gas concentration was too small. This can seen by comparing the images in the right column of Fig. 9, which show the standard GP results for different covariance functions in the top three rows and for the GPM below.

By carefully inspecting the results (best viewed in color), one can see slight differences resulting from the covariance functions. The squared exponential function yields smoother results than the Matérn kernels which can be seen on the border around the areas of high concentrations. The results measured by means of the NLPD computed based on separated test sets over multiple runs illustrate that the GPM models always outperformed the standard model (see tables). Furthermore, the Matérn kernels seem to be slightly better suited to

model gas distributions since they are less smooth compared to the squared exponential function.

Similar results are also obtained in the *outdoor* dataset. Mean and variance predictions of the different GP mixture models with different covariance functions are provided in Fig. 10. The corresponding result of the standard GP including a plot that illustrates the evolution of the negative log likelihood (NLPD) during sampling of the hyperparameters for the standard GP model and mixture GP model (SE covariance) is given in Fig. 11.

In all our experiments, we limited the number of data points in the reduced input set to n_1 = 100. The datasets itself contained between 2 500 and 3 500 measurements so our model was able to make accurate predictions with less than 5% of the data. Matrices of that size can be easily inverted. As a result the overall computation time for learning our model including cross validation is around 1 minute for all datasets shown above running Matlab on a standard laptop computer without explicitly optimized code.

Finally, we compared the mean estimates of our mixture model to the results obtained with the method of Lilienthal and Duckett [2004] as well as with an often used approach that uses a grid in combination with linear interpolation like in [Pyk *et al.*, 2006]. The results of this comparison in terms of the MSE measure are shown in Fig. 12. As can be seen from the diagram, our method outperforms both alternative methods.

4.3 Distribution Modeling in an Easy Setup

We also tested our gas distribution modeling algorithm with a "smoother" data set. The electronic nose on the mobile robot is also equipped with a temperature sensor and we used the temperature measurements as input to the gas distribution modeling algorithm proposed in this paper. Even so, the obtained measurements were temperature measurements instead of gas concentration measurements, our approach can be directly applied.

The measurements were recorded along a random sweeping trajectory in a corridor. The data set indicates a roughly linear gradient in the temperature distribution. In this situation, we expect that our mixture model should perform similar compared to the standard GP approach or the kernel extrapolation technique since the simpler techniques are also well suited to model such a function.

We therefore carried out the modeling task based on the temperature datasets with the different approaches. Our expectation was actually matched perfectly in this setting. Both mixture components of our method actually converged to approximatively the same solution and this model is more or less identical to the one generated by the standard GP approach as well as to the kernel extrapolation method. All three approaches yield nearly identical results differing by less than 1%. This holds for the MSE as well as for the NLPD (for GP and GPM), see Fig. 13. Obviously, the standard GP has a lower computational load than the mixture approach and thus is preferable if the designer of the system can ensure that no mixture components are needed to model the data.

5 Related Work

A common approach to creating representations for time-averaged concentration fields is to acquire measurements using a fixed grid of gas sensors over a long period of time. Equidistant gas sensor locations can be used to directly measure and map the average concentration values according to a given grid approximation of the environment. This approach was taken

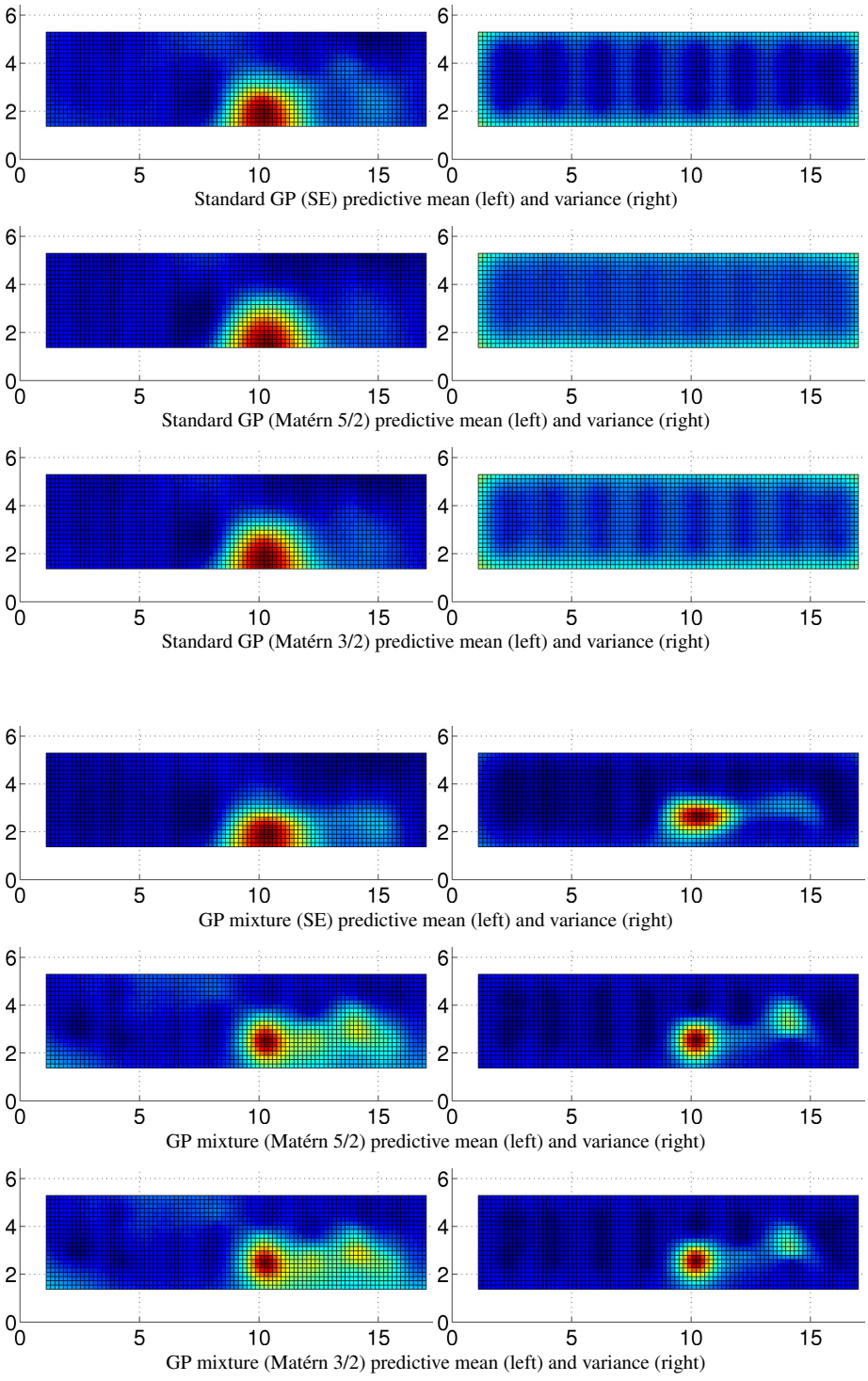


Fig. 9 Models learned from concentration data recorded in the *corridor* environment. The gas source was placed at the location (10, 3). We evaluated the standard GP and the our mixture model all using the different covariance functions. The unit of the x- and y-axis is meter.

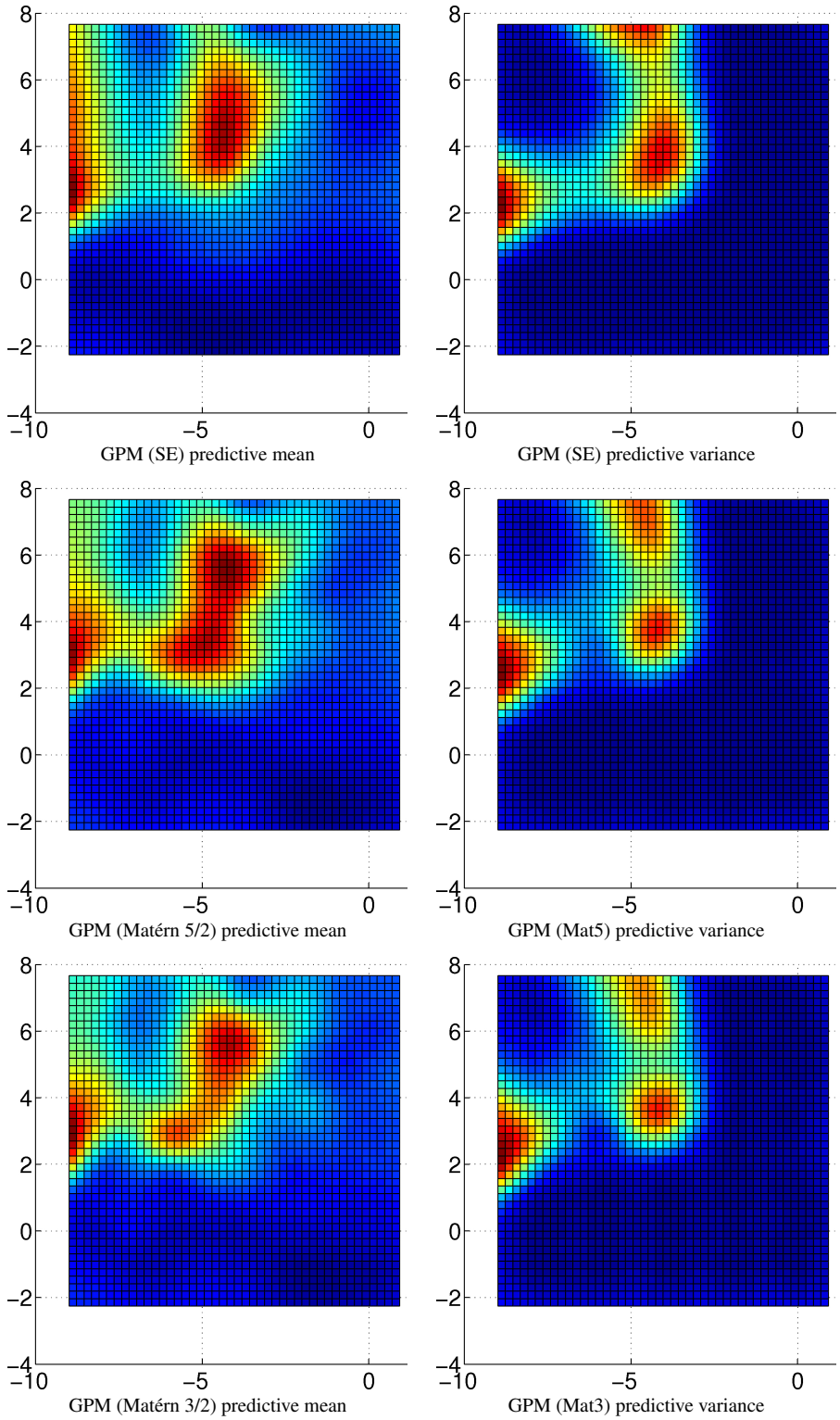


Fig. 10 Results for the *outdoor* dataset in an $8\,m$ by $8\,m$ area with an ethanol source in the center. The measured airflow indicates a major wind direction approximately from south-east to north-west. The unit of the x- and y-axis is meter.

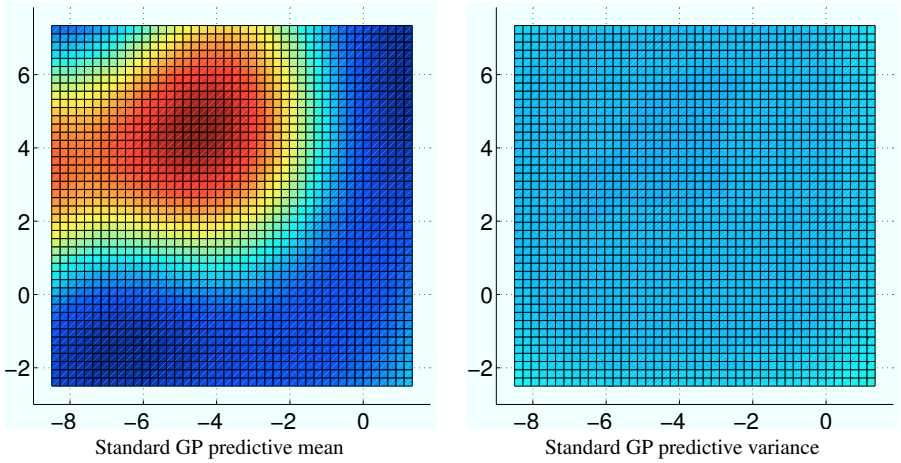


Fig. 11 Corresponding results for the *outdoor* dataset obtained with the standard GP model (top) and the evolution of the NLPD shown for the first 60 iterations (bottom). The unit of the x- and y-axis in the plots in the first row is meter.

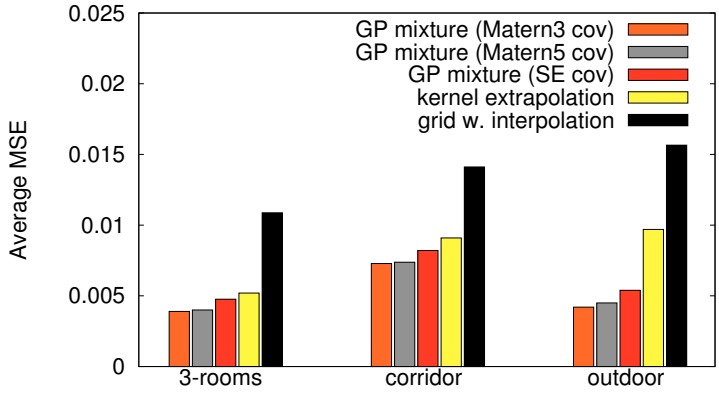


Fig. 12 Experimental comparison of our GP mixture model with different covariance functions to two alternative techniques in three real-world settings. The bars show the mean squared error of predicted compared to the measured concentration on a test set, averaged over 10 runs.

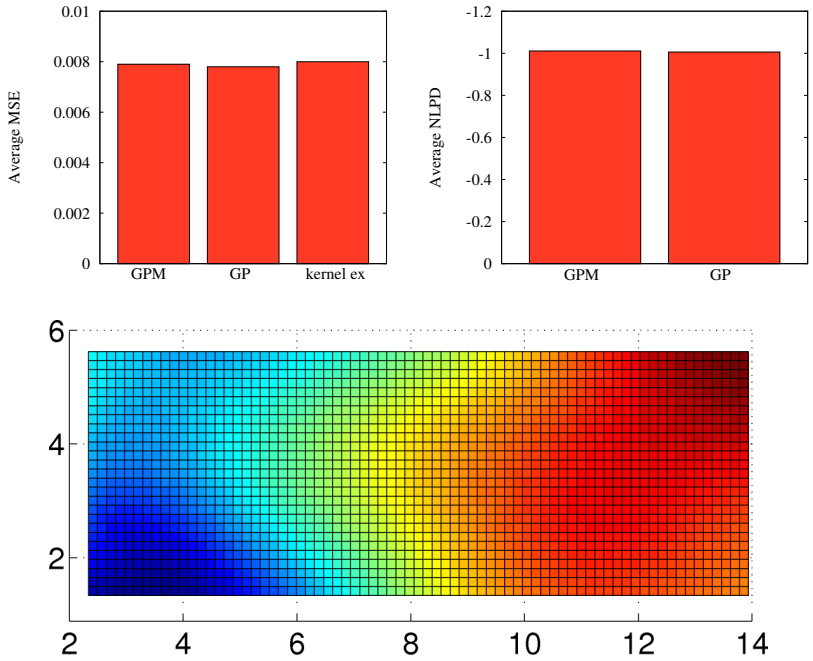


Fig. 13 Experimental comparison of the GP mixture model (GPM), the standard GP model (GP), and kernel extrapolation in a simple setting. Top: As expected, all three approaches perform more or less equal. Bottom: Model learned by the GPM approach (all approaches produced highly similar estimates). The unit of the x- and y-axis in the plots in the bottom row is meter.

by Ishida *et al.* [1998]—additionally considering partially simultaneous measurements. A similar method was used in [Purnamadjaja and Russell, 2005], but instead of the average concentration, the *peak* concentration observed during a sampling period of 20 s was considered to create the map.

Consecutive measurements with a single sensor and time-averaging over 2 minutes for each sensor location were used by Pyk *et al.* [2006] to create a map of the distribution of ethanol. Methods, which aim at determining a map of the instantaneous gas distribution from successive concentration measurements, rely on the assumption of a time-constant distribution profile, i.e. on uniform delivery and removal of the gas to be analyzed as well as on stable environmental conditions. Thus, the experiments of Pyk *et al.* were performed in a wind tunnel with a constant airflow and a homogeneous gas source. To make predictions about locations outside of the directly observed regions, the same authors apply bi-cubic interpolation in the case of equidistant measurements and triangle-based cubic filtering in the case, in which the measurement points are not equally distributed. A problem with such interpolation methods is that there is no means of "averaging out" instantaneous response fluctuations at measurement locations. Even if response values are measured close to each other, they will appear independently in the gas distribution map with interpolated values in between. Consequently, interpolation-based maps tend to become more and more jagged the more new measurements are added [Lilienthal *et al.*, 2006].

Histogram-based methods approximate the continuous distribution of gas concentration by means of binning according to regular grids. Hayes *et al.* [2002] for instance suggest

using two-dimensional histograms over the number of "odor hits" received in the corresponding area. "Odor hits" are counted whenever the response level of a gas sensor exceeds a defined threshold. In addition to the dependency of the gas distribution map on the selected threshold, a disadvantage of processing binary information only is that useful information contained in the (continuous) sensor readings is discarded. Further disadvantages of histogram-based methods for gas distribution modeling are their dependency on a properly chosen bin size and the lack of generalization across bins or beyond the inspection area.

Gas distribution mapping based on kernel extrapolation can be seen as an extension of the histogram-based approach. The idea was introduced by Lilienthal and Duckett [2004]. In this model, spatial integration is carried out by convolving sensor readings and modeling the information content of the point measurements with a Gaussian kernel. As discussed in [Lilienthal *et al.*, 2006], this method is related to nonparametric estimation using Parzen windows. The complexity of model-free approaches for converging to a stable representation—either in terms of time consumption or the number of sensors—scales quadratically with the size of the environment.

A model-based approach to estimate concentration maps has been described by Marques *et al.* [2005]. In this approach, the work space is discretized into a 2-d regular grid and the concentration in each cell is represented by a state variable. Using an advection-diffusion model of chemical transport, a reduced order Kalman filter is applied in order to estimate the state variables corresponding to the grid cells. According to the assumption of a non-turbulent transport model, the experimental run presented was carried out in an indoor environment with artificially introduced laminar airflow of approx. 1.5 m/s. Model-based approaches have also been applied to infer the parameters of an analytical gas distribution model from the measurements [Ishida *et al.*, 1998]. They naturally depend on the characteristics of the assumed model. Complex numerical models based on the simulation of fluid dynamics are computationally expensive and require accurate knowledge of the state of the environment (boundary conditions) which are typically not available in practice. Simpler analytical models, on the other hand, often make rather unrealistic model assumptions which hardly fit the real situation. Model-based approaches also rely on well-calibrated gas sensors and an established understanding of the sensor-environment interaction.

The Kalman filter approach by Marques *et al.* [2005] provides an estimate of the predictive uncertainty. A related approach is the work by Blanco *et al.* [2009] in which a Kalman filter is used for sequential Bayesian estimation on a 2-d grid. Instead of the advection-diffusion model, a stationary distribution is assumed in the latter work. It is important to note that the covariance obtained from these two approaches is the covariance of the mean, which can only decrease as new observations are processed. Since the predictive variance computed with the algorithm proposed in this paper can adapt to the real variability of the measurements at each location, its performance in terms of the average negative log likelihood is substantially better than with the approach by Blanco *et al.* [2009] (personal communication). We believe that this is also true for the mapping algorithm by Marques *et al.* [2005] although the two methods cannot be compared directly due to the strong assumptions on the environmental conditions by Marques *et al.*

In contrast to the above-mentioned approaches, we apply a Gaussian process-based mixture model to the problem of learning probabilistic gas distribution maps. The history of the idea behind the Gaussian process approach to regression dates back to Wiener [1964], Kolmogoroff [1941], O'Hagan [1978], and others (see [Rasmussen and Williams, 2006, Sec. 2.8]). For a detailed and quantitative comparison of GPs with alternative approaches such as neural networks, we refer to [Rasmussen, 1996]. GPs allow us to model the dependencies between measurements by means of a covariance function. They enable us to make

predictions at locations not observed so far and do not only provide the mean gas distribution but also the predictive uncertainty. Our mixture model is furthermore able to model sharp boundaries around areas of high gas concentration. Technically, we build on Tresp's mixture model of GP experts (see [Tresp, 2000]) better deal with the varying properties in the data. Extensions of this technique using infinite mixtures have been proposed by Rasmussen and Ghahramani [2002] and Meeds and Osindero [2006]. Other model extensions that aim at increasing the expressiveness of Gaussian processes include, e.g., heteroscedastic GPs for modeling input-dependent noise [Le *et al.*, 2005, Kersting *et al.*, 2007, Snelson and Ghahramani, 2006b], nonstationary GPs for modeling input-dependent smoothness [Paciorek and Schervish, 2003, Plagemann *et al.*, 2008, Schmidt and O'Hagan, 2003], or special covariance functions for non-vectorial inputs [Driessens *et al.*, 2006, Collins and Duffy, 2002]. Compared to the latter extensions to the standard GP model, the mixture model approach can be seen as the natural choice for the gas-mapping task, since the distribution of data points is multi-modal. Future work, however, could include a quantitative comparison of the alternative approaches or aim at integrating several of them.

The work presented here extends our previous RSS'2008 paper [Stachniss *et al.*, 2008]. First, we investigated the use of different covariance functions in the GP model for gas distribution mapping. This showed that there are better choices than the previously used squared exponential covariance function. Second, we extended the experimental section providing a larger set of experiments. We furthermore identified and evaluated a scenario which is well designed for the standard GP approach and evaluated the performance of our proposed mixture model. It turned out that in such a situations, designed for the standard GP, our approach performs equally well.

6 Conclusion

We considered the problem of modeling gas distributions from sensor measurements by means of sparse Gaussian process mixture models. Gaussian processes are an attractive modeling technique in this context since they do not only provide a gas concentration estimate for each point in the space but also the predictive uncertainty. Our approach learns a GP mixture model and simultaneously decreases the computational complexity by reducing the training set in order to achieve an efficient representation even for a large number of observations. The mixture model allows us to explicitly distinguish the different components of the spatial gas distribution, namely areas of high gas concentration from the smoothly varying background signal. This improves the accuracy of the gas concentration prediction.

Our method has been implemented and tested using gas sensors mounted on a real robot. With our method, we obtain gas distribution models that better explain the sensor data compared to techniques such as the standard GP regression for gas distribution mapping. Our approach and the one of Lilienthal and Duckett [2004] provide similar mean gas concentration estimates, their approach as well as the majority of techniques in the field, however, lack the ability of also estimating the corresponding predictive uncertainties.

Acknowledgments

This work has partly been supported by the DFG under contract number SFB/TR-8, and by the EC under contract number FP6-045299-Dustbot: Networked and Cooperating Robots

for Urban Hygiene, and FP7-224318-DIADEM: Distributed Information Acquisition and Decision-Making for Environmental Management.

References

[Blanco *et al.*, 2009] J.L. Blanco, J. Gonzalez, and A.J. Lilienthal. An efficient approach to probabilistic gas distribution mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009. Submitted to ICRA 2009.

[Collins and Duffy, 2002] M. Collins and N. Duffy. Convolution kernels for natural language. *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 1:625–632, 2002.

[Driessens *et al.*, 2006] K. Driessens, J. Ramon, and T. Gärtner. Graph kernels and Gaussian processes for relational reinforcement learning. *Machine Learning*, 2006.

[DustBot, 2008] DustBot. DustBot - Networked and Cooperating Robots for Urban Hygiene. http://www.dustbot.org, 2008.

[Frese, 2006] U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Journal of Autonomous Robots*, 21(2):103–122, 2006.

[Grisetti *et al.*, 2007] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[Hayes *et al.*, 2002] A.T. Hayes, A. Martinoli, and R.M. Goodman. Distributed Odor Source Localization. *IEEE Sensors Journal, Special Issue on Electronic Nose Technologies*, 2(3):260–273, 2002.

[Ishida *et al.*, 1998] H. Ishida, T. Nakamoto, and T. Moriizumi. Remote Sensing of Gas/Odor Source Location and Concentration Distribution Using Mobile System. *Sensors and Actuators B*, 49:52–57, 1998.

[Kersting *et al.*, 2007] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA, March 2007.

[Kolmogoroff, 1941] A. Kolmogoroff. Interpolation und extrapolation von stationren zuflligen folgen .(russian. german. *Bull. Acad. Sci. URSS, Ser. Math.*, 5:3–14, 1941.

[Le *et al.*, 2005] Q.V. Le, A.J. Smola, and S. Canu. Heteroscedastic gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, pages 489–496, New York, NY, USA, 2005. ACM Press.

[Lilienthal and Duckett, 2004] A. Lilienthal and T. Duckett. Building Gas Concentration Gridmaps with a Mobile Robot. *Robotics and Autonomous Systems*, 48(1):3–16, 2004.

[Lilienthal *et al.*, 2006] A. Lilienthal, A. Loutfi, and T. Duckett. Airborne Chemical Sensing with Mobile Robots. *Sensors*, 6:1616–1678, 2006.

[Lilienthal *et al.*, 2007] A. Lilienthal, A. Loutfi, J.L. Blanco, C. Galindo, and J. Gonzalez. A rao-blackwellisation approach to gdm-slam: Integrating slam and gas distribution mapping. In *Proc. of the European Conference on Mobile Robots (ECMR)*, pages 126–131, 2007.

[Marques *et al.*, 2005] Lino Marques, André Martins, and A. T. de Almeida. Environmental monitoring with mobile robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3624–3629, 2005.

[Meeds and Osindero, 2006] E. Meeds and S. Osindero. An alternative infinite mixture of gaussian process experts. In *Advances in Neural Information Processing Systems*, 2006.

[Murlis *et al.*, 1992] J. Murlis, J. S. Elkington, and R. T. Carde. Odor Plumes and How Insects Use Them. *Annual Review of Entomology*, 37:505–532, 1992.

[O'Hagan, 1978] A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40(1), 1978.

[Paciorek and Schervish, 2003] Christopher J. Paciorek and Mark J. Schervish. Nonstationary Covariance Functions for Gaussian Process Regression. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2003.

[Plagemann *et al.*, 2008] C. Plagemann, K. Kersting, and W. Burgard. Nonstationary gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*, Antwerp, Belgium, 2008.

[Purnamadjaja and Russell, 2005] A.H. Purnamadjaja and R.A. Russell. Congregation Behaviour in a Robot Swarm Using Pheromone Communication. In *Proc. of the Australian Conf. on Robotics and Automation*, 2005.

[Pyk *et al.*, 2006] P. Pyk, S. Bermúdez Badia, U. Bernardet, P. Knüsel, M. Carlsson, J. Gu, E. Chanie, B.S. Hansson, T.C. Pearce, and P.F. Verschure. An Artificial Moth: Chemical Source Localization Using a Robot Based Neuronal Model of Moth Optomotor Anemotactic Search. *Autonomous Robots*, 20:197–213, 2006.

[Rasmussen and Ghahramani, 2002] C.E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems 14*, 2002.

[Rasmussen and Williams, 2006] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[Rasmussen, 1996] C.E. Rasmussen. *Evaluation Of Gaussian Processes And Other Methods For Non-Linear Regression*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 1996.

[Rasmussen, 2006] C.E. Rasmussen. Minimize. http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize, 2006.

[Roberts and Webster, 2002] P.J.W. Roberts and D.R. Webster. Turbulent Diffusion. In H. Shen, A. Cheng, K.-H. Wang, M.H. Teng, and C. Liu, editors, *Environmental Fluid Mechanics - Theories and Application*. ASCE Press, Reston, Virginia, 2002.

[Schmidt and O'Hagan, 2003] A.M. Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *JRSS, series B*, 65:745–758, 2003.

[Smola and Bartlett, 2000] A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In *NIPS*, pages 619–625, 2000.

[Snelson and Ghahramani, 2006a] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1259–1266, 2006.

[Snelson and Ghahramani, 2006b] E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse gaussian process es. In *Uncertainty in Artifical Intelligence*, 2006.

[Stachniss *et al.*, 2008] C. Stachniss, C. Plagemann, A. Lilienthal, and W. Burgard. Gas distribution modeling using sparse gaussian process mixture models. In *Proc. of Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008. To appear.

[Tresp, 2000] V. Tresp. Mixtures of gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2000.

[Wandel *et al.*, 2003] M. Wandel, A. Lilienthal, T. Duckett, U. Weimar, and A. Zell. Gas distribution in unventilated indoor environments inspected by a mobile robot. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 507–512, 2003.

[Wiener, 1964] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[J5] K.M. Wurm, C. Stachniss, and G. Grisetti. Bridging the gap between feature- and grid-based slam. *Robots and Autonomous Systems*, 2009. In press.

Bridging the Gap Between Feature- and Grid-based SLAM

Kai M. Wurm Cyrill Stachniss Giorgio Grisetti
University of Freiburg, Dept. of Computer Science, Georges-Köhler-Allee 79, 79110 Freiburg, Germany

Abstract

One important design decision for the development of autonomously navigating mobile robots is the choice of the representation of the environment. This includes the question which type of features should be used or whether a dense representation such as occupancy grid maps is more appropriate. In this paper, we present an approach which performs SLAM using multiple representations of the environment simultaneously. It uses reinforcement to learn when to switch to an alternative representation method depending on the current observation. This allows the robot to update its pose and map estimate based on the representation that models the surrounding of the robot in the best way. The approach has been implemented on a real robot and evaluated in scenarios, in which a robot has to navigate in- and outdoors and therefore switches between a landmark-based representation and a dense grid map. In practical experiments, we demonstrate that our approach allows a robot to robustly map environments which cannot be adequately modeled by either of the individual representations.

Key words: SLAM, features, grid maps, learning, dual representation

1. Introduction

Building maps is one of the fundamental tasks of mobile robots. In the literature, the mobile robot mapping problem is often referred to as the *simultaneous localization and mapping (SLAM)* problem. It is considered to be a complex problem, because for localization a robot needs a consistent map of the environment and for acquiring a map a robot requires a good estimate of its location. This mutual dependency between the estimates about the pose of the robot and the map of the environment makes the SLAM problem hard and involves searching for a solution in a high-dimensional space.

A large variety of different estimation techniques has been proposed to address the SLAM problem. Extended Kalman filters, sparse extended information filters, maximum likelihood methods, particle filters, and several other techniques have been applied to estimate the trajectory of the robot as well as a map of the environment. Most approaches to mapping use a single scheme for representing the environment. Among the most popular ones are feature-based models such as sets of landmarks or dense representations such as occupancy grids. In a practical robotic application, the decision which model to use is largely influenced by the type of the environment the robot is deployed in. In large open spaces with predefined landmarks, for example, feature-based approaches often are preferred, whereas occupancy grid maps have widely been used in unstructured environments. In real world scenarios, however, one generally cannot assume that the environment is uniformly covered by specific features. Consider, for example, a surveillance system which can operate both inside of buildings and outside on parking spaces or large outdoor storage areas. Such a system has to be capable of dynamically choosing the best representation in each area to maximize its robustness.

The contribution of this paper is a novel approach which allows a mobile robot to utilize different rep-

resentations of the environment. At the example of a combination of feature-based models with occupancy grid maps we describe how a robot can perform the mapping process using both types of representation. It applies reinforcement learning to select the representation that is best suited to model the area surrounding the robot based on the current sensor observations and the state of the filter. We apply the approach in the context of a Rao-Blackwellized particle filter to maintain the joint posterior about the trajectory of the robot and the map of the environment.

As we will demonstrate in the experiments, our approach outperforms pure grid and pure feature-based approaches. Furthermore, our approach allows for modeling heterogeneous environments which cannot be adequately represented by either of the single representations. A motivating example is shown in Figure 1. Here, the environment consists of outdoor and indoor parts. A feature-based representation is well suited to model the outdoor part (Figure 1a) but cannot be used to correct odometry errors inside the buildings due to the lack of relevant features. A grid-based representation, in contrast, leads to false matches in the outdoor parts due to the sparsity of non max-range measurements there but accurately represents the inside of the buildings (see Figure 1b). Our system combines the advantages of both representations to generate a consistent map (Figure 1c).

This paper is organized as follows. After a discussion of related work, we briefly introduce the SLAM approach utilized in this paper, namely Rao-Blackwellized particle filters, in Section 3. Whereas Section 4 presents our approach for mapping with a dual representation of the environment, Section 5 explains our model selection technique based on reinforcement learning. Finally, we present experimental results obtained in simulation and on real robots in Section 6.

2. Related Work

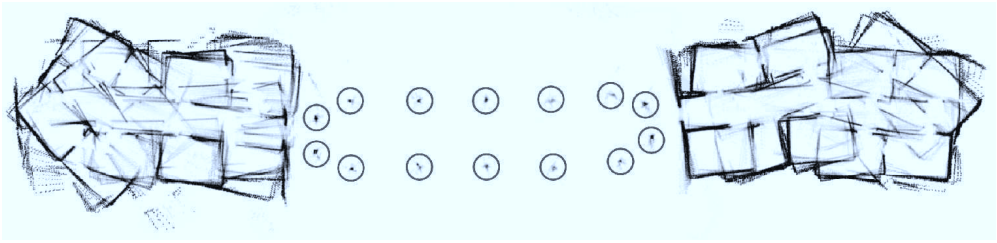
Mapping techniques for mobile robots can be roughly classified according to the map representation and the underlying estimation technique. One popular map representation is the occupancy grid [16]. Whereas such grid-based approaches are computationally expensive and typically require a huge amount of memory, they are able to represent arbitrary objects. It should be noted that to correct the robot pose estimate a certain amount of obstacles in the range of the robot's sensor is needed. This can be a problem if the range of the sensor is short as is the case with small scale laser scanners or if the environment is a large open area.

Feature-based representations are attractive because of their compactness. This is a clear advantage in terms of memory consumption and processing speed. However, such systems rely on predefined feature extractors, which assumes that some structures in the environments are known in advance. This clearly limits the field of action of a robot.

The model of the environment and the applied state estimation technique are often coupled. One of the most popular approaches are extended Kalman filters (EKFs) in combination with predefined landmarks. The effectiveness of the EKF approaches results from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [12,20]. Their weakness lies in the strong assumptions that have to be made on both the robot motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely identifiable. There exist techniques [18] to deal with unknown data association in the SLAM context, however, if these assumptions are violated, the filter is likely to diverge [5,11,25].

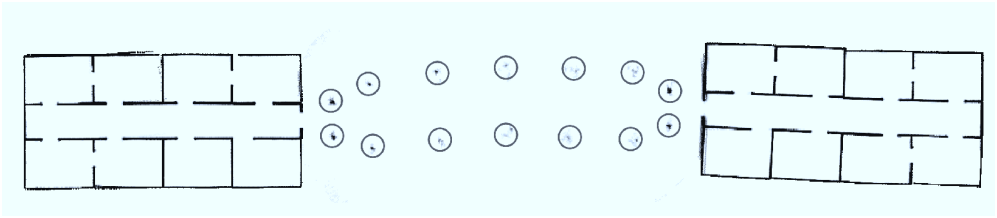
Thrun *et al.* [24] proposed a method that uses the inverse of the covariance matrix. The advantage of the sparse extended information filters (SEIFs) is that they make use of the approximative sparsity of the information matrix and in this way can perform predictions and updates in constant time. Eustice *et al.* [4] presented a technique to make use of exactly sparse information matrices in a delayed-state framework.

In a work by Murphy, Doucet, and colleagues [2,17], Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a possible trajectory of the robot and a map of the environment. The framework has been subsequently extended by Montemerlo *et al.* [14,15] for approaching the SLAM problem with landmark maps. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [3] and Hähnel *et al.* [8]. Whereas the first work describes an efficient map representation, the second presents an improved motion model that reduces the number of required particles. The work of Grisetti *et al.* [6] describes an improved variant of the algorithm proposed by Hähnel *et al.* [8] combined with the



(a) Feature-based mapping system (no features inside the buildings). Features are illustrated by circles.

(b) Grid-based mapping system (few structural information outside)



(c) Combining features and grid maps

Figure 1. When mapping environments that contain large open spaces with few landmarks as well as dense structures, a combination of feature maps and grids maps outperforms the individual techniques.

ideas of FastSLAM2 [14]. Instead of using a fixed proposal distribution, the algorithm computes an improved Gaussian proposal distribution on a per-particle basis on the fly. A further extension of this method which overcomes the limitation of the Gaussian assumption has recently been presented by Stachniss *et al.* [21]. Additional improvements concerning both runtime and memory requirements have been achieved by Grisetti *et al.* [7] by reusing already computed proposal distributions.

So far, there exist only very few methods that try to combine feature-based models with grid maps. One is the hybrid metric map (HYMM) approach [10]. It estimates the location of features and performs a triangulation between them. In this triangulation, a so called dense map is maintained which can be transformed according to the update of the corresponding landmarks. This allows the robot to obtain a dense map by using a feature-based mapping approach. However, it is still required that the robot is able to reliably extract landmarks.

A hybrid map is also used in [19]. Sim *et al.* propose a vision-based SLAM system which extracts 3D point landmarks from stereo camera images. In addition to the map of landmarks, an occupancy grid map is constructed which is used for safe navigation of the robot. In contrast to the approach described in this paper, the SLAM-system is only using the feature map for pose estimation, while the grid map is used for path planning in an exploration task. A similar approach is described by Makarenko *et al.* [13]. Here, an decision-theoretic exploration algorithm is described which uses a feature map for SLAM and maintains a grid map to determine known and unknown regions of the environment. However, the grid map is not used to correct the estimate of the robot's pose. Another combination of grid and feature maps has been proposed by Ho and Newman [9]. They use grid maps and visual features in a SLAM system. While the grid map generated from laser scans is used for pose estimation, visual features are used to improve the detection of loop closures.

3. Mapping with Rao-Blackwellized Particle Filters

According to Murphy [17], the key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the map m and the trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot. This estimation is performed given the observations $z_{1:t} = z_1, \ldots, z_t$ and the odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$ obtained by the mobile robot. The Rao-Blackwellized particle filter for SLAM makes use of the following factorization

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}). \quad (1)$$

This factorization allows us to first estimate only the trajectory of the robot and then to compute the map given that trajectory. This technique is often referred to as Rao-Blackwellization.

Typically, Eq. (1) can be calculated efficiently since the posterior about maps $p(m \mid x_{1:t}, z_{1:t})$ can be computed analytically using "mapping with known poses" [16] since $x_{1:t}$ and $z_{1:t}$ are known.

To estimate the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ about the potential trajectories, one can apply a particle filter. Each particle represents a potential trajectory of the robot. Furthermore, an individual map is associated with each sample. The maps are built from the observations and the trajectory hypothesis represented by the corresponding particle.

This framework allows a robot to learn models of the environment and estimate its trajectory but it leaves open how the environment is represented. So far, this approach has been applied using feature-based models [14,15] or grid maps [3,6,8,17]. Each representation has its advantages and one typically needs some prior information about the environment to select the appropriate model. In this paper, we combine both types of maps to represent the environment. This allows us to combine the advantages of both worlds. Depending on the most recent observation, the robot selects that model which is likely to be the best model in the current situation. In case the environment suggests the use of one single model, the result is the same as using the original approach.

4. Dual Model of the Environment

Our mapping system applies such a Rao-Blackwellized particle filter to maintain the joint posterior about the trajectory of the robot and the map of the environment. In contrast to previous algorithms, each particle carries a grid map as well as a map of features. The key idea is to maintain both representations simultaneously and to select in each step the model that is best suited to update the pose and map estimate of the robot. Our approach is independent of the actual features that are used. In our current system, we use a laser range finder and extract clusters of beam end points which are surrounded by free space. In this way, we obtain features from trees, street lamps, etc. Note that other feature detectors can be transparently integrated into our approach. The detector itself is completely transparent to the algorithm.

In each step, our algorithm considers the current estimate as well as the current sensor and odometry observation to select either the grid or the feature model to perform the next update step. This decision affects the proposal distribution in the particle filter used for mapping. The proposal distribution is used to obtain the next generation of particles as well as to compute the importance weights of the samples.

In the remainder of this section, we first introduce the characteristics of our particle filter. We then explain in the subsequent section how to actually select the model for the current step.

If the grid map is to be used, we draw the new particle poses from an improved proposal distribution as introduced by Grisetti *et al.* [6]. This proposal performs scan-matching on a per particle basis and then approximates the likelihood function by a Gaussian. This technique has been shown to yield accurate grid maps of the environment, given that there is enough structure to perform scan-matching for an initial estimate.

When using feature maps, we apply the proposal distribution as done by Montemerlo *et al.* [15] in the FastSLAM algorithm. For each particle $s_{t-1}^{(i)}$ in the current particle set a new hypothesis of the robot's pose is generated by sampling from the probabilistic motion model:

$$s_t^{(i)} \sim p(s_t \mid u_t, s_{t-1}^{(i)}) \quad (2)$$

After the proposal is used to obtain the next generation of samples, the importance weights are com-

puted according to Grisetti *et al.* [6] and Montemerlo *et al.* [15] respectively. Note that we compute for each sample i two weights $w_g^{(i)}$ (based on the grid map) and $w_f^{(i)}$ (based on the feature map). For resampling, one weight is required but we need both values in our decision process as explained in the following section.

To carry out the resampling step, we apply the adaptive resampling strategy originally proposed by Doucet [1]. It computes the so-called effective sample size or effective number of particles (N_{eff}) to decide whether to resample or not. This is done based on the weights resulting from the proposal used to obtain this generation of samples.

5. Model Selection

The probably most important aspect of our proposed algorithm is to decide which representation to choose given the current sensor readings and the filter. In the following, we describe different strategies we investigated and which are evaluated in the experimental section of this paper.

5.1. *Observation Likelihood Criterion*

A mapping approach that relies on scan-matching is most likely to fail if laser readings cannot be aligned to the map generated so far. For example, this will probably be the case in large open space with sparse observations. In such a situation it is often better to use a pre-defined feature extractor (in case there are feature) to estimate the pose of the robot.

A measure that can be used to detect such a situation is the observation likelihood that scan-matching seeks to maximize

$$l(z_t, x_t, m_{g,t}) = \max_{x_t} p(z_t \mid x_t, m_{g,t}). \tag{3}$$

To point-wise evaluate the observation likelihood of a laser observation, we use the so called "beam endpoint model" [23]. In this model, the individual beams within a scan are considered to be independent. The likelihood of a beam is computed based on the distance between the endpoint of the beam and the closest obstacle in the map from that point.

Calculating the average likelihood for all particles results in a value that can be used as a heuristic to decide which map representation to use in a given situation:

$$\bar{l} = \frac{1}{N} \sum_i l(z_t, x_t^{(i)}, m_{g,t}^{(i)}) \tag{4}$$

A *heuristic* for selecting the feature-based representation instead of the grid map can be obtained based on a threshold ($\bar{l} \leq c_1$).

However, care has to be taken when choosing c_1. If this threshold is not chosen optimally the feature map might be used even if it offers no advantage over the grid map. This will increase the likelihood of a poor state estimate and therefore of inconsistencies in the map.

5.2. *N_{eff} Criterion*

As described above, each particle i carries two weights $w_g^{(i)}$ and $w_f^{(i)}$, one for the grid-map and one for the feature-map. These weights can be seen as an indicator of how well a particle explains the data and therefore can be also used as a heuristic for model selection. Since the weights of a particle are based on different types of measurement, they cannot be compared directly. What can be compared, however, is the weight distribution over the filter.

One way to measure this difference in the individual weights is to compute the variance of the weights. Intuitively a set of weights with low variance does not strongly favor any of the hypothesis represented by the particles, while a high variance indicates that some hypotheses are more likely than others.

This suggests that a strategy based on the N_{eff} value, which is strongly related to the variance of the weights, can be a reasonable heuristic. N_{eff} is computed for both sets of weights as

$$N_{\mathit{eff}}^g = \frac{1}{\sum_{i=1}^N (w_g^{(i)})^2} \text{ and } N_{\mathit{eff}}^f = \frac{1}{\sum_{i=1}^N (w_f^{(i)})^2}. \tag{5}$$

It can be easily seen, that a higher variance in the weights yields a lower N_{eff} value. Assuming that a set of particles with a higher variance in the weights is usually more discriminative, it seems reasonable to switch to the feature-based model whenever $N_{\mathit{eff}}^f < N_{\mathit{eff}}^g$.

In our experiments, this heuristic generally led to good results. Nevertheless, there are two aspects which have to be considered.

Firstly the variance in particles weights usually does not change abruptly but gradually. For this reason, the N_{eff} criterion might fail to indicate the optimal point in time to switch the actively used representation. This will most notably happen at

junctions between areas where one is best modeled using grid maps and the other is best modeled using feature maps. Note that such a behavior can also be advantageous for example in case of false feature detections.

A second problem arises from the fact that frequent resampling in a particle filter can lead to particle depletion [1]. Since our implementation uses adaptive resampling based on the N_{eff} value, choosing the representation with the lower N_{eff} will in general also lead to more frequent resampling actions.

5.3. *Reinforcement Learning for Model Selection*

Both approaches described above are clearly heuristics. In this section, we describe how to use reinforcement learning to combine the strengths of both heuristics while avoiding their pitfalls. The basic idea of reinforcement learning is to find a mapping from states S to actions A which maximizes a numerical reward signal r (see [22] for an introduction). Such a mapping is called a policy and can be learned by interacting with the environment. Inspired by the human learning method of trial and error, this class of learning algorithms performs a series of actions and analyzes the obtained reward.

There exist a number of algorithms for reinforcement learning. Depending on the prior knowledge an agent has about its environment some approaches may be more appropriate than others. For example, if it can be modeled as an Markov decision process, techniques such as policy iteration can be utilized. In case no model of the environment is available, Monte Carlo methods or Temporal-Difference Learning (TD learning) can be applied.

For our approach, we use the SARSA algorithm [22] which is a popular algorithm among the TD methods and does not require a model of the environment. It learns an action-value function $Q(s, a)$ which assigns a value to state-action pairs. Those values can then be used to generate a policy (e.g., choose the action that has the highest value in a given state). The basic steps are given in Algorithm 1.

To apply this method to our model selection problem, we have to define the states S, the actions A, and the reward $r : S \to R$. Defining the actions is straight forward as $A = \{a_g, a_f\}$, where a_g defines the use of the grid map and a_f the use of the feature map.

Algorithm 1 The SARSA Algorithm

```
Initialize Q(s, a) arbitrarily
for all episodes do
  initialize s
  choose a from s using policy derived from Q
  repeat
    take action a, observe r, s'
    choose a' from s' using policy derived from Q
    Q(s, a) = Q(s, a) + α[r + γQ(s', a') − Q(s, a)]
    s = s'; a = a'
  until s is a terminal state
end for
```

The state set has to be defined in a way that it represents all necessary information about the sensor input and the filter to make a decision. To achieve this, our state consists of the average scan matching likelihood $\bar{l}$, a boolean variable given by $N^f_{\mathit{eff}} < N^g_{\mathit{eff}}$, and a boolean variable indicating if a known feature has currently been detected or not. This results in

$$S := \{\bar{l}\} \times \{1_{N^f_{\mathit{eff}} < N^g_{\mathit{eff}}}\} \times \{1_{\text{ feature detected}}\}. \quad (6)$$

The value of $\bar{l}$ is divided into (here seven) discrete intervals $(0.0-0.15, 0.16-0.3, 0.31-0.45, 0.46-0.6, 0.61-0.75, 0.76-0.9, 0.91-1.0)$, resulting in $7 \times 2 \times 2 = 28$ states. It is important to keep the number of states small since learning the policy otherwise may require too many computational resources, even as a preprocessing step which needs to be executed only once.

The policy is learned on simulated data where the true robot pose x^*_t is available in every time step t. We use the weighted average deviation from the true pose to define our reward-function. To avoid a punishment that result from wrong decisions in the past (e.g., a wrong rotation), we only use the deviation accumulated since the last evaluation step $t-1$:

$$r(s_t) = r(s_{t-1}) \; - \; \sum_{i=1}^{N} w^{(i)}_t \; ||x^{(i)}_t - x^*_t|| \quad (7)$$

Deviations from the simulated path result in negative rewards. As mentioned in the previous section, each particle stores two weights. For calculating the weighted average, we use $w^{(i)}_g$ if the last action taken was a_g and $w^{(i)}_f$ if a_f was taken.

The environment for learning consists of building-like structures with hallways and an outdoor part that models a set of trees. We recorded a simulated path and executed 1000 runs of the learning algorithm. During learning, we us an ε-greedy policy. In

state s, a greedy policy chooses the action a which has the highest value $Q(s, a)$. In contrast to this, an ε-greedy policy allows exploratory actions by choosing a random action with likelihood ε.

More exploration usually facilitates faster learning so a value of $\varepsilon = 0.6$ was used in our learning experiments. The learning rate α was set to a fixed value of 0.001, the discounting factor γ was set to 0.9, which are standard values and led to good results in our experiments.

This technique results in a policy that tells the robot when to select the feature-based representation and when to choose the grid map. Note that our approach to learn a strategy for making decisions is independent of the actual feature detector used. One could even use this approach to choose among multiple feature detectors.

The overall mapping algorithm is depicted in Algorithm 2.

Algorithm 2 Our combined approach

Require:
$\mathcal{S}_{t-1}$, the sample set of the previous time step
$z_{l,t}$, the most recent laser scan
$z_{f,t}$, the most recent feature measurement
u_{t-1}, the most recent odometry measurement
Ensure:
$\mathcal{S}_t$, the new sample set

maptype = decide($\mathcal{S}_{t-1}, z_{l,t}, z_{f,t}, u_{t-1}$)

$\mathcal{S}_t = \{\}$
for all $s^{(i)}_{t-1} \in \mathcal{S}_{t-1}$ **do**
$< x^{(i)}_{t-1}, w^{(i)}_{g,t-1}, w^{(i)}_{f,t-1} m^{(i)}_{g,t-1}, m^{(i)}_{f,t-1} >= s^{(i)}_{t-1}$

// compute proposal
if (**maptype** = **grid**) **then**
$x^{(i)}_t \sim P(x_t \mid x^{(i)}_{t-1}, u_{t-1}, z_{l,t})$
else
$x^{(i)}_t \sim P(x_t \mid x^{(i)}_{t-1}, u_{t-1})$
end if

// update importance weights
$w^{(i)}_{g,t} = \text{updateGridWeight}(w^{(i)}_{g,t-1}, m^{(i)}_{g,t-1}, z_{l,t})$
$w^{(i)}_{f,t} = \text{updateFeatureWeight}(w^{(i)}_{f,t-1}, m^{(i)}_{f,t-1}, z_{f,t})$

// update maps
$m^{(i)}_{g,t} = \text{integrateScan}(m^{(i)}_{g,t-1}, x^{(i)}_t, z_{l,t})$
$m^{(i)}_{f,t} = \text{integrateFeatures}(m^{(i)}_{f,t-1}, x^{(i)}_t, z_{f,t})$

// update sample set
$\mathcal{S}_t = \mathcal{S}_t \cup \{< x^{(i)}_t, w^{(i)}_{g,t}, w^{(i)}_{f,t}, m^{(i)}_{g,t}, m^{(i)}_{f,t} >\}$
end for

for $i = 1$ to N **do**
if (**maptype** = **grid**) **then**
$w^{(i)} = w^{(i)}_g$
else
$w^{(i)} = w^{(i)}_f$
end if
end for

$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2}$
if $N_{\text{eff}} < T$ **then**
$\mathcal{S}_t = \text{resample}(\mathcal{S}_t, \{w^{(i)}\})$
end if

6. Experiments

Our approach has been evaluated using simulated and real robot data. The experiments have been designed to verify that our mapping approach is able to reduce the error compared to the purely feature-based technique (FastSLAM [15]) and to the purely grid-based approach [6]. We specifically considered environments which cannot be mapped accurately using one single model. In those cases the result is the same as using the original approach.

6.1. *Simulation Experiments*

For generating the simulated data, we used the Carnegie Mellon Robot Navigation Toolkit. The simulated environment used to test our approach is shown in Figure 2. It shows two symmetric buildings connected by an alley. The environment is spanning 70 m in total. We simulated a laser range finder with a maximum range of 4m which is less than the distance between the trees in the alley (5m). This limited sensor range is a realistic setting since it corresponds to the maximum range of small scale laser scanners such as the Hokuyo URG.

The motivating example in the introduction of this paper shows example results obtained with the different approaches. Figure 1 (a) is the result of the purely feature-based FastSLAM approach. Since no features are found inside the building structures, the robot cannot correct its trajectory inside the buildings. In contrast, the trajectory through the alley is well approximated using this approach.

The purely grid-based approach [6] is able to correctly map the buildings but introduces large errors in the alley (see Figure 1 (b)). Due to the limited range of the sensor, too few obstacles are observed and therefore no accurate scan registration is possible and thus the grid-based approach fails to map the alley appropriately.

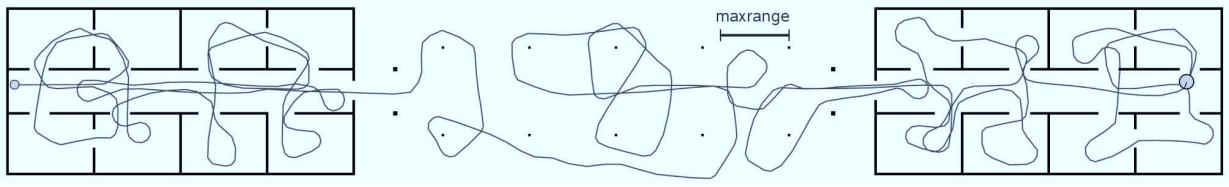


Figure 2. Simulated environment used to test our approach. Shown are the ground truth map and trajectory of the robot.

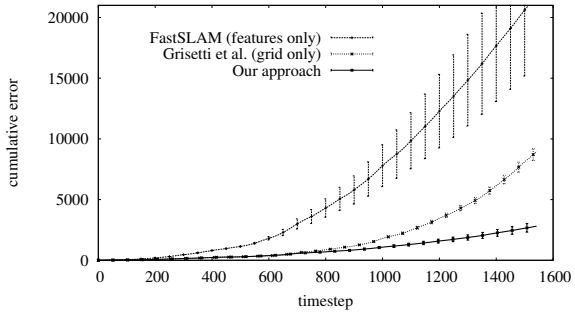


Figure 3. Deviation of the weighted mean of the samples from ground truth using grid- and feature-model on their own and using the combined approach. The error bars illustrate the 0.05 confidence level.

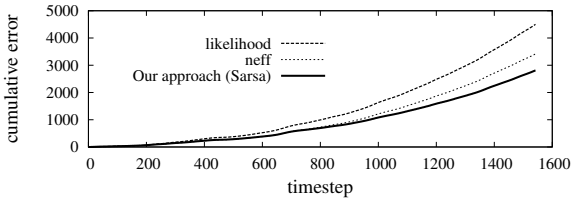


Figure 4. Deviation of the weighted mean of the samples from ground truth using the scan-match likelihood heuristic, the N_{eff} heuristic and our approach.

In contrast to this, our combined approach using the learned policy is able to correct the trajectory of the robot all the time by selecting the appropriate model. It uses the grid maps inside the buildings and the features outside. The resulting map is shown in Figure 1 (c).

To evaluate our approach more quantitatively, we repeated this experiment for 20 times with different random seeds. We compared our approach to the pure feature-based approach and the pure grid-based approach. The results in Figure 3 show, that the combined approach is significantly better than both pure approaches (0.05 significance).

In addition to this, we compared the solution obtained by SARSA with those of the scan-matching heuristic and the N_{eff} heuristic described above. We measured the absolute deviation from ground truth in every time step. Figure 4 illustrates that the average error of the learned model selection policy is lower than when using the heuristics. However, we could not show that this improvement is significant.

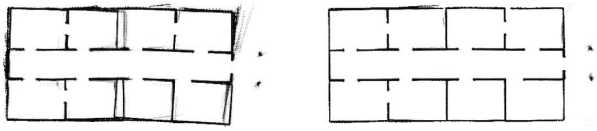


Figure 5. Typical mapping results when using the likelihood-heuristic (left) and our SARSA-based approach (right).

One interesting fact can be observed when comparing the results of these three technique by manual inspection. Even if the error measured as the deviation from the ground truth is not significantly smaller for the learned policy, the maps typically look nicer. The scan-match heuristic for example relies on a fixed threshold c_1. If the threshold is not optimally tuned, in can happen that the grid approach is not selected even though it would be the better model. This leads to walls which are blurred or slightly misaligned. Figure 5 depicts a magnified view of two maps illustrating the difference. Unfortunately, it is hard to design a measure that is able to take this blurriness into account. A similar effect can be observed when using the N_{eff} criterion.

Figure 6 shows the decisions our algorithm made while processing the simulated dataset. One can clearly see that the grid map is used for pose estimation inside the buildings while the feature map is used outside of the buildings. At a first glance it looks as if the system used the wrong model around time-step 1000. Using features here is correct though since the robot entered the building to the right only briefly and then moved in the outdoor part again until approximately time-step 1100.

6.2. *Real World Experiments*

Two real world data sets used in this experiment have been recorded at Freiburg University. The experiments have been conducted using an ActivMedia Pioneer 2-AT robot equipped with a SICK LMS laser range finder.

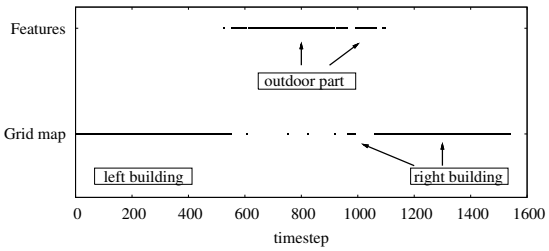


Figure 6. Active Representation chosen by our learned approach.

Figure 7. Test environment with poles.

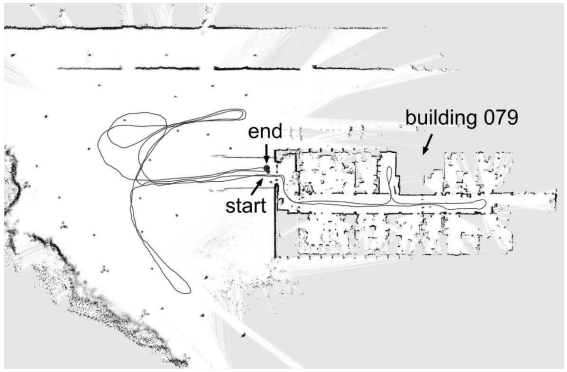


Figure 8. Grid map of environment and approximated robot trajectory.

6.2.1. *Poles*

This experiment has been conducted in an office-building and on the street in front of the building. Since the outdoor environment does not contain a sufficient amount of detectable features, 20 artificial landmarks (poles) have been placed there. We used poles with a diameter of 15 cm and a height of about 100 cm (see Figure 7). The robot was manually steered through the environment. It started outdoors in front of the building, went through the landmarks and then entered the building. After traversing the building the robots returns to the outdoor area and finishes its trajectory next to the starting location. To prevent the laser-scanner from detecting neighboring buildings, the sensor-range has been limited to 5 m. Again, this maximum range is not artificially bad but corresponds to small scale laser range scanners.

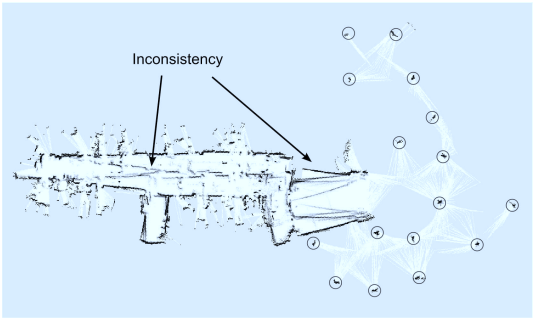
Since no ground truth was available, we measured the error against an approximated robot path which was generated using the grid-based approach of Grisetti *et al.* [6] with the full sensor range of the SICK laser scanner. Due to the 80 m sensing range, the robot always observed enough obstacles to build an accurate map. The resulting map and the obtained trajectory can be seen in Figure 8.

We compared the results from our approach to those generated by a pure grid- and feature-based approach. Looking at the exemplary results in Figure 9 notable differences in the quality of the maps can be seen. While the grid-based approach performs very well inside of the building it introduces numerous false matches in the outdoor area. In contrast, the feature based approach is able to map the outdoor part well but is obviously not suited to correct odometry errors inside the building. Combining the strengths of both approaches, our combined method leads to an overall consistent map.

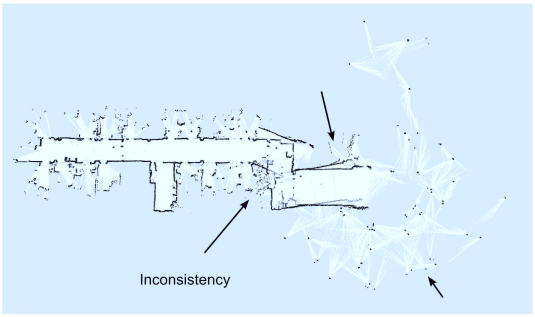
To evaluate our approach quantitatively we repeated the mapping process for 20 times. Figure 10 plots the cumulative deviation from the approximated ground truth trajectory for each of the three evaluated strategies. The results confirm the results of the simulated experiment. They show that the combined approach performs significantly better than both pure approaches (0.05 significance).

6.2.2. *Parking Lot*

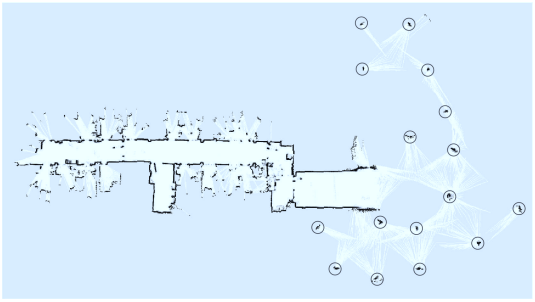
The Freiburg computer science campus includes a parking lot of about 50 m by 120 m (see Figure 11). Lamps are set in two rows at a distance of 16 m in one direction and 25 m in the other direction. The second dataset was recorded on this parking lot at a time when no cars were present and therefore only the lamps caused reflections of the laser beams. The robot was steered manually through a building, around the neighboring parking lot, and back into the building again. The trajectory is plotted in Figure 12. To evaluate our approach, again we limited the maximum laser range of the scanner to a range which is considerably smaller than the distance be-



(a) Feature-based mapping system



(b) Grid-based mapping system



(c) Combined SLAM system using features and grid maps

Figure 9. Examples of resulting maps in the poles experiment. Using only a feature map (a) or a grid map (b) leads to inconsistent maps in this environment. Combining both representation yields a consistent map (c).

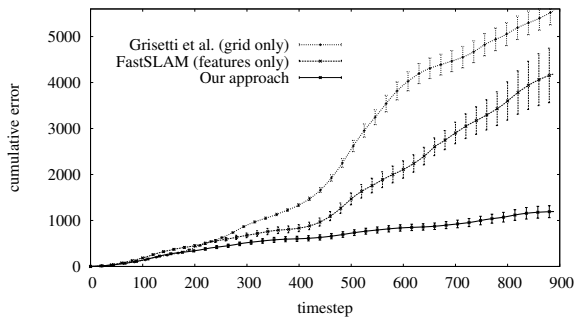


Figure 10. Results of the poles experiment. Cumulative error in the pose estimation measured against the approximated ground truth trajectory. The error bars correspond to the 0.05 confidence level.

tween two lamps.

The approximated ground truth trajectory has been generated in the same way as we did in the first experiment. Figure 13 shows the error of the weighted mean trajectory over time.

In summary, both real robot experiments lead to similar results as the experiment using simulated data. The combined approach performed significantly better compared to both traditional SLAM techniques using the limited sensor range.

The computational requirements of the presented approach are approximatively the sum of the individual techniques. On a notebook computer, our implementation runs online.

7. Conclusions

In this paper, we presented an improved approach to learning models of the environment with a Rao-Blackwellized particle filter. Our approach maintains feature maps as well as grid maps simultaneously to represent spatial structures. This allows the robot to select the model which provides the best expected estimates online. The model selection procedure is obtained by a reinforcement learning approach. The robot considers the previous estimate as well as the current observations to chose the model that will be used in the upcoming correction step. The process itself is independent of the actual feature detector. Our approach has been implemented and evaluated on real robot data as well as in simulation experiments. We showed that the presented technique allows a robot to more robustly learn maps of different types of environments. It outperforms traditional approaches that use only features or only grid maps. In real world experiments, we also showed that our approach is able to map environments which could not be modeled by either of the single approaches.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A3), and by the EC under contract number FP6-IST-034120-muFly.

References

[1] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing

Figure 11. Parking lot at Freiburg campus.

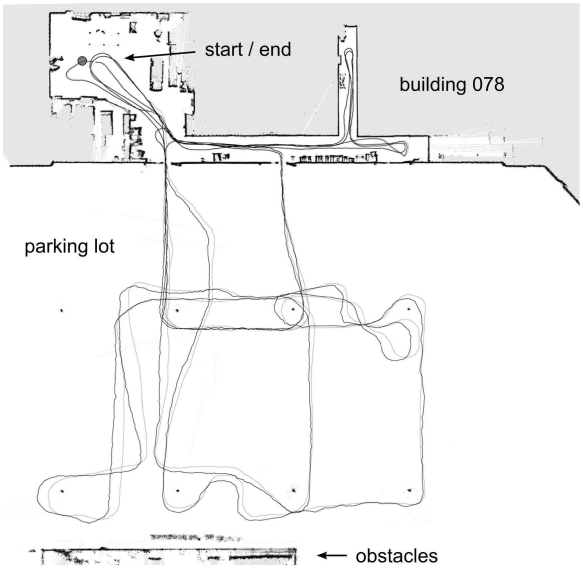


Figure 12. Grid map of the parking lot and neighboring building 078 at Freiburg campus. The approximated robot trajectory is shown in dark gray, the result of our combined mapping approach is shown in light gray.

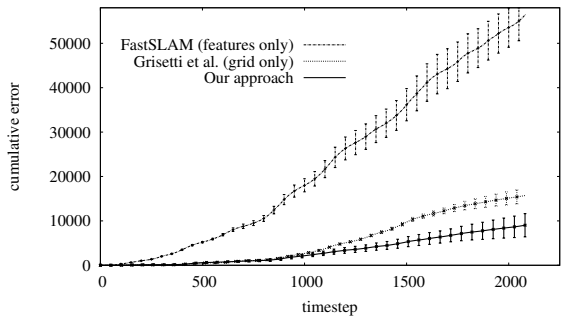


Figure 13. Results of the parking lot experiment. Deviation of the weighted mean of the samples from the estimated trajectory (using the 80m range scanner).

Group, Dept. of Engeneering, University of Cambridge, 1998.

[2] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 176–183, Stanford, CA, USA, 2000.

[3] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[4] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[5] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17–26, Seattle, WA, USA, 2001.

[6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[7] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and accurate slam with rao-blackwellized particle filters. *Journal of Robotics & Autonomous Systems*, 55(1):30–38, 2007.

[8] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[9] K.L. Ho and P.M. Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 2006.

[10] E.M. Nebot J.I. Nieto, J.E. Guivant. The hybrid metric maps (HYMMs): A novel map representation for denseslam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[11] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[12] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[13] A.A. Makarenko, S.B. Williams, F. Bourgoult, and H.F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[14] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[15] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous

localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[16] H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 116–121, St. Louis, MO, USA, 1985.

[17] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[18] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[19] R. Sim and J.J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2082–2089, Oct. 2006.

[20] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[21] C. Stachniss, G. Grisetti, W. Burgard, and N. Roy. Evaluation of gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[22] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[23] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*, chapter Robot Perception, pages 171–172. MIT Press, 2005.

[24] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H.F. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *J. of Robotics Research*, 23(7/8):693–716, 2004.

[25] J. Uhlmann. *Dynamic Map Building and Localization: New Theoretical Foundations*. PhD thesis, University of Oxford, 1995.

Kai M. Wurm is a research scientist at the University of Freiburg (Germany). He studied computer science at the University of Freiburg and received his diploma degree in 2007. His research interests lie in the fields of SLAM, multi-robot exploration, and terrain classification.

Cyrill Stachniss studied computer science at the University of Freiburg and received his Ph.D. degree in 2006. After his Ph.D., he was a senior researcher at ETH Zurich. Since 2007, he is an academic advisor at the University of Freiburg in the Laboratory for Autonomous Intelligent Systems. His research interests lie in the areas of robot navigation, exploration, SLAM, as well as learning approaches.

Giorgio Grisetti is working as a Post-doc at the Autonomous Intelligent Systems Lab of the University of Freiburg. Up to 2006, he was a PhD student at University of Rome "La Sapienzia" in the Intelligent Systems Lab. His advisor was Daniele Nardi and he received his PhD degree in April 2006. His research interests lie in the areas of mobile robotics. His previous and current work aims to provide effective solutions to mobile robot navigation in all its aspects: SLAM, localization, and path planning.

[J6] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009. In press.

Non-linear Constraint Network Optimization for Efficient Map Learning

Giorgio Grisetti* Cyrill Stachniss Wolfram Burgard

University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany
{grisetti | stachnis | burgard}@informatik.uni-freiburg.de, *corresponding author

***Abstract*— Learning models of the environment is one of the fundamental tasks of mobile robots since maps are needed for a wide range of robotic applications, such as navigation and transportation tasks, service robotic applications, and several others. In the past, numerous efficient approaches to map learning have been proposed. Most of them, however, assume that the robot lives on a plane. In this paper, we present a highly efficient maximum likelihood approach that is able to solve 3D as well as 2D problems. Our approach addresses the so-called graph-based formulation of the simultaneous localization and mapping (SLAM) and can be seen as an extension of Olson's algorithm [27] towards non-flat environments. It applies a novel parameterization of the nodes of the graph that significantly improves the performance of the algorithm and can cope with arbitrary network topologies. The latter allows us to bound the complexity of the algorithm to the size of the mapped area and not to the length of the trajectory. Furthermore, our approach is able to appropriately distribute the roll, pitch and yaw error over a sequence of poses in 3D mapping problems. We implemented our technique and compared it to multiple other graph-based SLAM solutions. As we demonstrate in simulated and in real world experiments, our method converges faster than the other approaches and yields accurate maps of the environment.**

I. Introduction

To efficiently solve the majority of robotic applications such as transportation tasks, search and rescue, or automated vacuum cleaning a map of the environment is required. Acquiring such models has therefore been a major research focus in the robotics community over the last decades. Learning maps under pose uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. The approaches mainly differ in the underlying estimation technique such as extended Kalman filters, information filters, particle filters, smoothing, or least-square error minimization techniques.

In this paper, we consider the popular and so-called "graph-based" or "network-based" formulation of the SLAM problem in which the poses of the robot are modeled by nodes in a graph [5, 8, 11, 14, 16, 22, 27, 13, 36, 26]. Spatial constraints between poses that result from observations and from odometry are encoded in the edges between the nodes.

In the context of graph-based SLAM, one typically considers two different problems. The first one is to identify the constraints based on sensor data. This so-called data association problem is typically hard due to potential ambiguities or symmetries in the environment. A solution to this problem is often referred to as the SLAM front-end and it directly deals with the sensor data. The second problem is to correct the poses of the robot to obtain a consistent map of the environment *given* the constraints. This part of the approach is often referred to as the optimizer or the SLAM back-end. To solve this problem, one seeks for a configuration of the nodes that maximizes the likelihood of the observations encoded in the constraints. Often, one refers to the negative observation likelihood as the error or the energy in the network. An alternative view to the problem is given by the spring-mass model in physics. In this view, the nodes are regarded as masses and the constraints as springs connected to the masses. The minimal energy configuration of the springs and masses describes a solution to the mapping problem. As a motivating example, Figure 1 depicts an uncorrected constraint network and the corresponding corrected one.

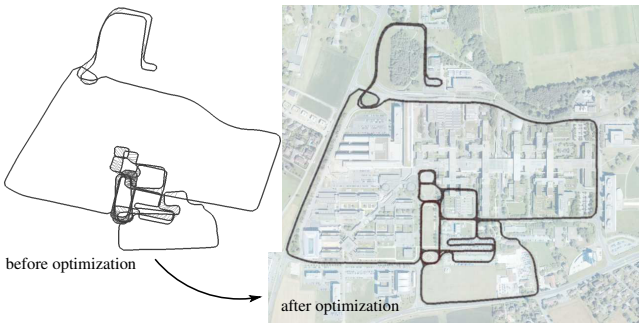


Fig. 1. Constraint network corresponding to a dataset recorded with an instrumented car at the EPFL campus in Lausanne before (left) and after (right) optimization. The corrected network is overlayed with an aerial image.

Popular solutions to compute a network configuration that minimizes the error introduced by the constraints are iterative approaches. They can be used to either correct all poses simultaneously [14, 20, 22, 36] or to locally update parts of the network [5, 11, 13, 16, 26, 27]. Depending on the used technique, different parts of the network are updated in each iteration. The strategy for defining and performing these local updates has a significant impact on the convergence speed.

In this paper, we restrict ourselves to the problem of finding the most likely configuration of the nodes *given* the constraints. To find the constraints from laser range data one can, for example, apply the front-end of the ATLAS framework introduced by Bosse *et al.* [2], hierarchical SLAM [6], or the work of Nüchter *et al.* [26]. In the context of visual SLAM, a potential approach to obtain such constraints has recently

been proposed by Steder *et al.* [33].

Our approach uses a tree structure to define and efficiently update local regions in each iteration by applying a variant of stochastic gradient descent. It extends Olson's algorithm [27] and converges significantly faster to highly accurate network configurations. Compared to other approaches to 3D mapping, our technique utilizes a more accurate way to distribute the rotational error over a sequence of poses. Furthermore, the complexity of our approach scales with the size of the environment and not with the length of the trajectory as it is the case for most alternative methods.

The remainder of this paper is organized as follows. In Section II, we formally introduce the graph-based formulation of the mapping problem and explain the usage of stochastic gradient descent to reduce the error of the network configuration. Whereas Section III introduces our tree parameterization, Section IV describes our approach to distribute the rotational errors over a sequence of nodes. In Section V we then provide an upper bound for this error distribution. Section VI, explains how to obtain a reduced graph representation to limit the complexity. After describing the experimental results with our approach in Section VII, we provide a detailed discussion of related work in Section VIII.

II. Maximum Likelihood Mapping using a Constraint Network

Most approaches to network-based or graph-based SLAM focus on estimating the most-likely configuration of the nodes and are therefore referred to as maximum-likelihood (ML) techniques [5, 11, 13, 14, 22, 27, 36]. Such techniques do not compute the full posterior about the map and the poses of the robot. The approach presented in this paper also belongs to this class of methods.

A. Problem Formulation

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. For a more precise formulation consider the following definitions:

- Let $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ be a vector of parameters which describes a configuration of the nodes. Note that the parameters x_i do not need to be the absolute poses of the nodes. They are arbitrary variables which can be mapped to the poses of the nodes in real world coordinates.
- Let us furthermore assume that δ_{ji} describes a constraint between the nodes j and i. It refers to an observation of node j seen from node i. These constraints are the edges in the graph structure.
- The uncertainty in δ_{ji} is represented by the information matrix Ω_{ji}.
- Finally, $f_{ji}(\mathbf{x})$ is a function that computes a zero noise observation according to the current configuration of the nodes j and i. It returns an observation of node j seen from node i.

Figure 2 illustrates an observation between two nodes.

Given a constraint between node j and node i, we can define the *error* e_{ji} introduced by the constraint as

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

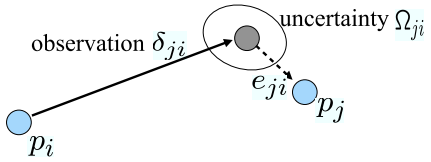


Fig. 2. Example of an observation of the node j seen from i.

as well as the *residual* r_{ji}

$$r_{ji}(\mathbf{x}) \quad = \quad -e_{ji}(\mathbf{x}). \tag{2}$$

Note that at the equilibrium point, e_{ji} is equal to 0 since $f_{ji}(\mathbf{x}) = \delta_{ji}$. In this case, an observation perfectly matches the current configuration of the nodes. Assuming a Gaussian observation error, the corresponding negative log likelihood results in

$$\begin{aligned} F_{ji}(\mathbf{x}) \quad &\propto \quad (f_{ji}(\mathbf{x}) - \delta_{ji})^T \, \Omega_{ji} \, (f_{ji}(\mathbf{x}) - \delta_{ji}) && (3) \\ &= \quad e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}) && (4) \\ &= \quad r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (5) \end{aligned}$$

Under the assumption that the observations are independent, the overall negative log likelihood of a configuration $\mathbf{x}$ is

$$\begin{aligned} F(\mathbf{x}) \quad &= \quad \sum_{\langle j,i \rangle \in \mathcal{C}} F_{ji}(\mathbf{x}) && (6) \\ &\propto \quad \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (7) \end{aligned}$$

Here $\mathcal{C} = \{\langle j_1, i_1 \rangle, \ldots, \langle j_M, i_M \rangle\}$ is a set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists.

The goal of an ML approach is to find the configuration $\mathbf{x}^*$ of the nodes that maximizes the likelihood of the observations. This can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{8}$$

There are multiple ways of solving Eq. (8). They range from approaches applying gradient descent, conjugate gradients, Gauss Seidel relaxation, multi-level relaxation, or LU-decomposition. In the following section, we briefly introduce stochastic gradient descent, which is the technique our approach is based on.

B. Stochastic Gradient Descent for Maximum Likelihood Mapping

Olson *et al.* [27] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to address the SLAM problem. The approach minimizes Eq. (8) by sequentially selecting a constraint $\langle j, i \rangle$ (without replacement) and by moving the nodes of the network in order to decrease the error introduced by the selected constraint. Compared to the standard formulation of gradient descent, the constraints are not optimized as a whole but individually. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \underbrace{\lambda \cdot K J_{ji}^T \Omega_{ji} r_{ji}}_{\Delta \mathbf{x}_{ji}} \tag{9}$$

Here $\mathbf{x}$ is the set of variables describing the locations of the poses in the network and K is a pre-conditioning matrix. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, and r_{ji} is the residual.

Reading the term $\Delta\mathbf{x}_{ji}$ of Eq. (9) from right to left gives an intuition about the iterative procedure:

- The term r_{ji} is the residual which corresponds to the negative error vector. Changing the network configuration in the direction of the residual r_{ji} will decrease the error e_{ji}.
- The term Ω_{ji} represents the information matrix of a constraint. Multiplying it with r_{ji} scales the residual components according to the information encoded in the constraint.
- The Jacobian J_{ji}^T maps the residual term into a set of variations in the parameter space.
- The term K is a pre-conditioning matrix. It is used to scale the variations resulting from the Jacobian depending on the curvature of the error surface. Approaches such as Olson's algorithm [27] or our previous work [13] apply a diagonal pre-conditioning matrix computed from the Hessian H as

$$K \quad = \quad [\mathrm{diag}(H)]^{-1}. \tag{10}$$

- Finally, the quantity λ is a learning rate that decreases with each iteration of SGD and that ensures the convergence of the system.

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing each constraint individually. Thus, a portion of the network, namely the nodes involved in a constraint, is updated in each step. Obviously, updating the different constraints one after each other can have antagonistic effects on a subset of variables. To merge the contribution of the individual constraints, one uses the learning rate to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different sub-problems to asymptotically converge towards an equilibrium point that is the solution reported by the algorithm.

Whereas this framework allows us to iteratively reduce the error given the network of constraints, it leaves open how the nodes are represented or parameterized. However, the choice of the parameterization has a strong influence on the performance of the algorithm. The next section addresses the problem of how to parameterize a graph so that the optimization can be carried out efficiently.

III. Tree Parameterization for SGD

The poses $\mathbf{p} = \{p_1, \ldots, p_n\}$ of the nodes define the configuration of the network. They can be described by a vector of parameters $\mathbf{x}$ such that a bijective mapping g between $\mathbf{p}$ and $\mathbf{x}$ exists.

$$\mathbf{x} = g(\mathbf{p}) \qquad\qquad \mathbf{p} = g^{-1}(\mathbf{x}) \tag{11}$$

As explained above, in each iteration SGD decomposes the problem into a set of subproblems and solves them sequentially, where a subproblem is the optimization of a single constraint.

The parameterization g defines not only how the variables of the nodes are described but also the subset of variables that are modified by a single constraint update. A good parameterization defines the subproblems in a way that the combination step leads only to small changes of the individual solutions.

Olson *et al.* [27] proposed to use the so-called incremental pose parameterization for 2D problems. For each node i in the graph, they store a the parameter x_i which is the vector difference between the poses of the node i and the node $i-1$

$$x_i = p_i - p_{i-1}. \tag{12}$$

This parameterization has the advantage of allowing fast constraint updates. As discussed in [13], updating a constraint between two nodes i and j requires to update all nodes $k = i+1, \ldots, j$. This leads to a low convergence speed if $i \ll j$. Furthermore this parameterization requires that the nodes are arranged in a sequence given by the trajectory.

As mentioned above, a major contribution of this paper is an algorithm that preserves the advantages of the incremental approach but overcomes its drawbacks. The first goal is to be able to deal with arbitrary network topologies since this enables us to compress the graph whenever robot revisits a place. As a result, the size of the network is proportional to the visited area and not to the length of the trajectory. The second goal is to make the number of nodes in the graph which are updated by each constraint mainly dependent on the topology of the environment and not the trajectory taken by the vehicle. For example, in the case of a loop-closure a large number of nodes need to be updated but in all other situations the update is limited to a small number of nodes to keep the interactions between constraints small.

Our idea is to define a parameterization based on a tree structure. To obtain a tree from a given graph, we compute a spanning tree. Given such a tree, we define the parameterization for a node as

$$x_i \quad = \quad p_i \ominus p_{\mathrm{parent}(i)}, \tag{13}$$

where $p_{\mathrm{parent}(i)}$ refers to the parent of node i in the spanning tree. The operators $\oplus$ and $\ominus$ are the standard pose compounding operators [22]. As defined in Eq. (13), the tree stores the relative transformations between poses.

Given a root node that represents the origin, such a spanning tree can be obtained by using Dijkstra's algorithm. In this work, we use the uncertainty encoded in the information matrices of the constraints as costs. In this way, Dijkstra's algorithm provides the "lowest uncertainty tree" (shortest path tree) of the graph.

Note that this tree does not replace the graph as an internal representation. The tree only defines the parameterization of the nodes. For illustration, Figure 3 depicts a graph together with one potential parameterization tree.

According to Eq. (13), one needs to process the tree up to the root to compute the actual pose of a node in the global reference frame. However, to obtain only the relative transformation between two arbitrary nodes, one needs to traverse the tree from the first node upwards to the first common ancestor of both nodes and then downwards to the

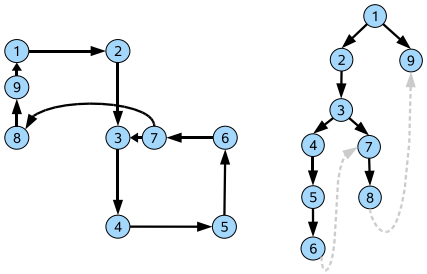


Fig. 3. Left: Example for a constraint network. Right: A possible tree parameterization for this graph. For illustration reasons, the off-tree constraints are also plotted (dashed gray).

second node. The same holds for computing the error of a constraint. Let the *path* $\mathcal{P}_{ji}$ of a constraint between the nodes i and j be the sequence of nodes in the tree that need to be traversed in order to reach the node j staring from node i. Such a path can be divided into an ascending part $\mathcal{P}_{ji}^{[-]}$ of the path starting from node i and a descending part $\mathcal{P}_{ji}^{[+]}$ to node j. We refer to the length of path of a constraint on the tree as $|\mathcal{P}_{ji}|$. We can then compute the residual of the constraint by

$$r_{ji} \quad = \quad (p_i \oplus \delta_{ji}) \ominus p_j \tag{14}$$

For simplicity of notation, we will refer to the pose vector of a node as the 6D vector $p_i = (x \;\; y \;\; z \;\; \phi \;\; \theta \;\; \psi)^T$ and to its associated homogeneous transformation matrix as P_i. The same holds for the parameters used for describing the graph. We denote the parameter vector of the pose i as x_i and its transformation matrix X_i. The transformation matrix corresponding to a constraint δ_{ji} is referred to as Δ_{ji}.

A transformation matrix X_k consists of a rotational matrix R_k and a translational component t and it has the following form

$$X_i \quad = \quad \begin{pmatrix} R_k & t_k \\ 0 & 1 \end{pmatrix} \tag{15}$$

with

$$X_i^{-1} \quad = \quad \begin{pmatrix} R_k^T & -R_k^T t_k \\ 0 & 1 \end{pmatrix}. \tag{16}$$

Accordingly, we can compute the residual in the reference frame of the node j as

$$r_{ji} \quad = \quad P_j^{-1}(P_i \Delta_{ji}) \tag{17}$$

$$= \quad \left(\prod_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} X_{k^{[+]}} \right)^{-1} \prod_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} X_{k^{[-]}} \Delta_{ji}. \tag{18}$$

At this point one can directly compute the Jacobian from the residual and apply Eq. (9) to update the constraint. Note that with this parameterization the Jacobian has exactly $|\mathcal{P}_{ji}|$ non zero blocks, since only the parameters in the path of the constraint appear in the residual.

IV. Updating the Tree Parameterization

So far, we described the prerequisites for applying the preconditioned stochastic gradient descent to correct the poses of a network. The goal of the update rule in SGD is to iteratively update the configuration of a set of nodes in order to reduce the error introduced by a constraint. In Eq. (9), the term $J_{ji}^T \Omega_{ji}$ maps the variation of the error to a variation in the parameter space. This mapping, however, is a linear function. As illustrated by Frese and Hirzinger [10], the error might increase when applying such a linear function in case of non-linear error surfaces. In the three-dimensional space, the three rotational components often lead to highly non-linear error surfaces. Therefore, it is problematic to apply SGD as well as similar minimization techniques directly to *large* mapping problems in combination especially when there is *high noise* in the observations.

In our approach, we therefore choose a modified update rule. To overcome the problem explained above, we apply a *non-linear function* to describe the variation. As in the linear case, the goal of this function is to compute a transformation of the nodes along the path $\mathcal{P}_{ji}$ of the tree so that the error introduced by the corresponding constraint is reduced. The design of this function is presented in the remainder of this section. In our experiments, we observed that such an update typically leads to a smooth deformation of the nodes along the path when reducing the error. This deformation is done in two steps. We first update the rotational components R_k of the variables x_k before we update the translational components t_k.

A. Update of the Rotational Component

Without loss of generality, we consider the origin p_i of the path $\mathcal{P}_{ji}$ to be in the origin of our reference system. The orientation of p_j (in the reference frame of p_i) can be computed by multiplying the rotational matrices along the path $\mathcal{P}_{ji}$. To increase the readability of the document, we refer to the individual rotational matrices along this path as R_k neglecting the indices (compare Eq. (18)). The orientation of p_j is described by

$$R_{1:n} \quad := \quad R_1 R_2 \ldots R_n, \tag{19}$$

where n is the length of the path $\mathcal{P}_{ji}$.

Distributing a given error over a sequence of 3D rotations, can be described in the following way: we need to determine a set of increments in the intermediate rotations of the chain so that the orientation of the last node (here node j) is $R_{1:n}B$ where B the matrix that rotates x_j to the desired orientation based on the error/residual. Formulated in a mathematical way, we need to compute a set of new rotational matrices R'_k to update the nodes so that

$$R_{1:n}B \quad = \quad \prod_{k=1}^{n} R'_k. \tag{20}$$

To obtain these R'_k we compute a rotation Q in the global reference frame such that

$$A_n B \quad = \quad Q A_n, \tag{21}$$

where A_n denotes the rotation of the n^{th} node *in the global reference frame*. By multiplying both sides of Eq. (21) with A_n^T from the right hand side we obtain

$$Q \quad = \quad A_n B A_n^T. \tag{22}$$

We now decompose the rotation Q into a set of incremental rotations

$$Q_{1:n} \quad := \quad Q \quad = \quad Q_1 Q_2 \cdots Q_n \tag{23}$$

and compute the individual matrices Q_k by using the spherical linear interpolation (slerp) [1].

For this decomposition of Q we use the parameter $u \in [0, 1]$ with $\mathbf{slerp}(Q, 0) = I$ and $\mathbf{slerp}(Q, 1) = Q$. Accordingly, the rotation Q_k is

$$Q_k \quad = \quad [\mathbf{slerp}(Q, u_{k-1})]^T \, \mathbf{slerp}(Q, u_k). \tag{24}$$

Furthermore, the rotation matrix A'_k of the poses P'_k along the path is

$$A'_k = Q_{1:k} A_k. \tag{25}$$

We furthermore compute the new rotational components R'_k of each node k as

$$R'_k = [A'_{\text{parent}(k)}]^T A'_k. \tag{26}$$

In Eq. (27), the learning rate λ is directly incorporated in the computation of the values u_k. In this way, the slerp function takes care of the appropriate scaling of the rotations.

In addition to that, we consider the pre-conditioning matrix and the length of the path when computing u_k. Similar to Olson *et al.* [27], we clamp the product $\lambda|\mathcal{P}_{ji}|$ to lie between $[0, 1]$ for not overshooting. In our implementation, we compute these values as:

$$u_k = \min\left(1, \lambda|\mathcal{P}_{ji}|\right) \left[\sum_{m \in \mathcal{P}_{ji} \wedge m \leq k} d_m^{-1}\right] \left[\sum_{m \in \mathcal{P}_{ji}} d_m^{-1}\right]^{-1} \tag{27}$$

Here, d_m is defined as the sum of the smallest eigenvalues of the information matrices of all constraints connecting the node m:

$$d_m = \sum_{\langle i,m \rangle} \min\left[\text{eigen}(\Omega_{im})\right] \tag{28}$$

This is an approximation which works well in case of roughly spherical covariances. Note that the eigenvalues need to be computed only once in the beginning and are then stored in the tree.

For simplicity of presentation, we demonstrated how to distribute the rotational error while keeping the node i fixed. In our implementation, however, we fix the position of the so-called "top node" in the path which is the node that is closest to the root of the tree (smallest level in the tree). As a result, the update of a constraint has less side-effects on other constraints in the network. Fixing the top node instead of node i can be obtained by simply saving the pose of the top node before updating the path. After the update, one transforms all nodes along path in way that the top node maintains its previous pose. Furthermore, we used the matrix notation in this paper to formulate the error distribution since it provides an easier notation. However, in our implementation we use quaternions for representing the rotations because they are numerically more stable. Both formulations are theoretically equivalent. Note that an open source implementation of our optimizer is available online [32].

B. Update of the Translational Component

Compared to the update of the rotational component described above, the update of the translational component can be done in a straightforward manner. In our current system, we distribute the translational error over the nodes along the path without changing the previously computed rotational component.

All nodes along the path are translated by a fraction of the residuals in the x, y, and z components. This fraction depends on the uncertainty of the individual constraints encoded in the corresponding covariance matrices and is scaled with the learning rate, similarly to the case of updating the rotational component.

V. Analysis of the Rotational Residual

When distributing an rotational error over a sequence of nodes $i, \ldots, j$, one may increase the absolute value of the residual $r_{k,k-1}$ between consecutive constraints along the path (and thus the error $e_{k,k-1}$). For the convergence of SGD, however, it is important that this error is bounded. Therefore, in this section we analyze evolution of the rotational residual after distributing an error according to Section IV-A.

A generic 3D rotation can be described in terms of an axis and an angle. Given an rotational matrix $\mathcal{R}$ we will refer respectively to its axis of rotation as $\mathbf{axisOf}(\mathcal{R})$ and as $\mathbf{angleOf}(\mathcal{R})$. According to [1], the slerp interpolation returns a set of rotation along the same axis as follows

$$\mathcal{R}' \quad = \quad \mathbf{slerp}(\mathcal{R}, u) \tag{29}$$

$$\mathbf{axisOf}(\mathcal{R}') \quad = \quad \mathbf{axisOf}(\mathcal{R}) \tag{30}$$

$$\mathbf{angleOf}(\mathcal{R}') \quad = \quad u \cdot \mathbf{angleOf}(\mathcal{R}). \tag{31}$$

When distributing the rotation Q over a sequence of poses according to Eq. (23), we decompose it into a sequence of incremental rotations $Q = Q_1 Q_2 \cdots Q_n$. From Eq. (24) we know that

$$\alpha_k = \mathbf{angleOf}(Q_k) = (u_k - u_{k-1}) \cdot \mathbf{angleOf}(Q). \tag{32}$$

In the following, we show that when distributing the rotational error along a loop the *angle* of the residual $\mathbf{angleOf}(r_{k,k-1})$ between the consecutive poses $k-1$ and k along the path does not increase more than α_k.

According to Eq. (18), the residual of a constraint between the nodes $k-1$ and k is

$$r_{k,k-1} \quad = \quad X_k^{-1} \Delta_{k,k-1}. \tag{33}$$

Since we are focusing only on the rotational component of the residual, we ignore the translational part:

$$r_{k,k-1} \quad = \quad R_k^T \Delta_{k,k-1} \tag{34}$$

$$R_k^T \quad = \quad r_{k,k-1} \Delta_{k,k-1}^T. \tag{35}$$

After updating the rotations $A_1, \ldots, A_n$ along the chain using Eq. (25), we obtain a new set of rotations $A'_1, \ldots, A'_n$ in the global reference frame. From these rotations, we recover the updated rotational parameters R'_k, by using Eq. (26):

$$\begin{aligned} R'_k &\overset{(26)}{=} A'^T_{k-1} A'_k && (36)\\ &\overset{(25)}{=} [Q_{1:k-1} R_{1:k-1}]^T \cdot [Q_{1:k} R_{1:k}] && (37)\\ &= [R_{1:k-1}]^T Q_k R_{1:k} && (38)\\ &= [R_{1:k-1}]^T Q_k R_{1:k-1} R_k. && (39) \end{aligned}$$

We then compute the residual $r'_{k,k-1}$ after the update as

$$\begin{aligned} r'_{k,k-1} &\overset{(34)}{=} R'^T_k \Delta_{k,k-1} && (40)\\ &\overset{(39)}{=} R_k^T [R_{1:k-1}]^T Q_k^T R_{1:k-1} \Delta_{k,k-1} && (41)\\ &\overset{(35)}{=} r_{k,k-1} \underbrace{\Delta^T_{k,k-1} [R_{1:k-1}]^T}_{=:Y^T} Q_k^T \underbrace{R_{1:k-1} \Delta_{k,k-1}}_{=:Y} \\ &= r_{k,k-1} Y^T Q_k^T Y && (42) \end{aligned}$$

In Eq. (42), the term $Y^T Q_k^T Y$ quantifies the increase in the residual of a constraint between two consecutive nodes after the update. Since Y and Q are rotation matrices, we obtain

$$|\mathbf{angleOf}(Y^T Q_k^T Y)| = |\mathbf{angleOf}(Q_k)| = |\alpha_k|. \quad (43)$$

Thus, the change of the new residual is at most α_k and therefore bounded. This is a relevant advantage compared to the error distribution presented by Grisetti *et al.* [12] which was not bounded in such a way.

VI. Complexity and Graph Reduction

Due to the nature of stochastic gradient descent, the complexity of our approach per iteration depends linearly on the number of constraints since each constraint is selected once per iteration (in a random order). For each constraint $\langle j, i \rangle$, our approach modifies exactly those nodes which belong to the path $\mathcal{P}_{ji}$ in the tree.

The path of constraint is defined by the tree parameterization. As a result, different paths have different lengths. Thus, we consider the average path length l to specify the complexity. It corresponds to average the number of operations needed to update a single constraint during one iteration. This results in a complexity of $\mathcal{O}(M \cdot l)$, where M is the number of constraints. In our experiments we found that l typically is in the order of $\log N$, where N is the number of nodes.

Note that there is further space for optimizations. The complexity of the approach presented so far depends on the length of the trajectory and not on the size of the environment. These two quantities are different if the robot revisits already known areas. This becomes important whenever the robot is deployed in a bounded environment for a long time and has to update its map over time. This is also known as lifelong map learning. Since our parameterization is not restricted to a trajectory of sequential poses, we have the possibility of a further optimization. Whenever the robot revisits a known place, we do not need to add new nodes to the graph. We can assign the current pose of the robot to an already existing node in the graph and update the constraints with respect to that node.

To avoid adding new constraints to the network, we can refine an existing constraint between two nodes in case of a new observation. Let $\delta^{(1)}_{ji}$ be a constraint already stored in the graph and let $\delta^{(2)}_{ji}$ be the new constraint that would result from the current observation. Both constraints can be combined to a single constraint which has the following information matrix and mean:

$$\begin{aligned} \Omega_{ji} &= \Omega^{(1)}_{ji} + \Omega^{(2)}_{ji} && (44)\\ \delta_{ji} &= \Omega^{-1}_{ji} (\Omega^{(1)}_{ji} \cdot \delta^{(1)}_{ji} + \Omega^{(2)}_{ji} \cdot \delta^{(2)}_{ji}) && (45) \end{aligned}$$

This can be seen as an approximation similar to adding a rigid constraint between nodes. However, if local maps (e.g., grid maps) are used as nodes in the network, it makes sense to use such an approximation since one can localize a robot in an existing map quite accurately.

As a result, the size of the problem does not increase when revisiting known locations. The complexity specified above stays the same but M as well as N refer to as the reduced quantities. As our experiments illustrate, this node reduction technique leads to an increased convergence speed since less nodes and constraints need to be considered.

VII. Experiments

This section is designed to evaluate the properties of our approach described above. We first demonstrate that our method is well suited to cope with the motion and sensor noise from an instrumented car equipped with laser range scanners. Second, we present the results of simulated experiments based on large 2D and 3D datasets. Finally, we compare our approach to different other methods including Olson's algorithm [27], multi-level relaxation [11], and SAM [4, 19].

A. Mapping with a Car-like Robot

In the first experiment, we applied our method to a real world three-dimensional dataset recorded with an instrumented car. Using such cars as robots became popular in the robotics community [3, 29, 34, 38]. We used a Smart car equipped with 5 SICK laser range finders and various pose estimation sensors for data acquisition. Our robot constructs local three-dimensional maps, so-called multi-level surface maps [36], and builds a network of constrains where each node represents such a local map. The localization system of the car is based on D-GPS (here using only standard GPS) and IMU data. This information is used to compute the incremental constraints between subsequent poses. Constraints resulting from revisiting an already known area are obtained by matching the individual local maps using ICP. More details on this matching can be found in our previous work [29].

We recorded a large-scale dataset at the EPFL campus where the robot moved on a 10 km long trajectory. The dataset includes multiple levels such as an underground parking garage and a bridge with an underpass. The motivating example of this paper (see Figure 1) depicts the input trajectory and an overlay of the corrected trajectory on an aerial image. As can

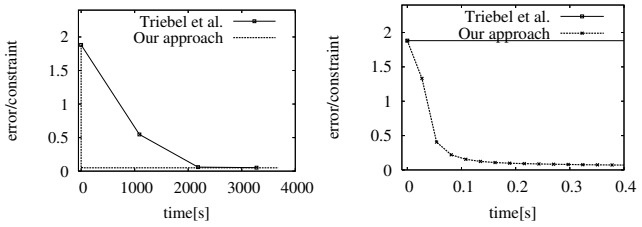


Fig. 4. The evolution of the average error per constraint (computed according to Eq. (7) divided by the number of constraints) of the approach of Triebel *et al.* [36] and our approach for the dataset recorded with the autonomous car. The right image shows a magnified view to the first 400 ms.

be seen, the trajectory actually matches to the streets in the aerial image (image resolution: 0.5 m per pixel).

We used this dataset to compare our new algorithm to the approach of Triebel *et al.* [36] that iteratively applies LU decomposition. In this experiment, both approaches converge to more or less the same solution. The time needed to achieve this correction, however, is by orders of magnitudes smaller when applying our new technique. Figure 4 plots the average error per constraint versus the execution time.

B. Quantitative Results and Comparison with SAM in 3D

The second set of experiments is designed to measure the performance of our approach for correcting 3D constraint networks and in comparison with the smoothing and mapping (SAM) approach of Dellaert [4]. In these simulation experiments we moved a virtual robot on the surface of a sphere. An observation was generated each time the current position of the robot was close to a previously visited location. We corrupted the observations with a variable amount of Gaussian noise to investigate the robustness of the algorithms.

Figure 5 depicts a series of graphs obtained by our algorithm using three datasets generated with different noise levels. The observation and motion noise was set to $\sigma = 0.05/0.1/0.2$ in each translational component (in m) and rotational component (in radians).

As can be seen, our approach converges to a configuration with a low error. Especially for the last dataset, the rotational noise with a standard deviation of 0.2 (radians) for each movement and observation is high. After around 250 iterations, the system converged. Each iteration took 200 ms for this dataset with around 85,000 constraints.

We furthermore compared our approach to the smoothing and mapping approach of Dellaert [4]. The SAM algorithm can operate in two modes: as a batch process which optimizes the entire network at once or in an incremental mode. The latter one only performs an optimization after a fixed number of nodes has been added. This way of incrementally optimizing the network is more robust since the initial guess for the network configurations is computed based on the result of the previous optimization. As a result, the risk of getting stuck in a local minima is typically reduced. However, this procedure leads to a significant computational overhead. Table I summarizes the results obtained with the SAM algorithm. As can be seen, the batch variant of the SAM algorithm got stuck in local minima for the sphere datasets with medium and large noise. The incremental version, in contrast,always converged but still required substantially more computation time than our current implementation of our approach.

TABLE I
COMPARISON TO SAM

noise level	**SAM (batch)**	**SAM (incremental)**	**Our method (batch)**
$\sigma = 0.05$	119 s	not tested (see batch)	20 s (100 iterations)
$\sigma = 0.1$	diverged	270 s (optimized each 100 nodes)	40 s (200 iterations)
$\sigma = 0.2$	diverged	510 s (optimized each 50 nodes)	50 s (250 iterations)

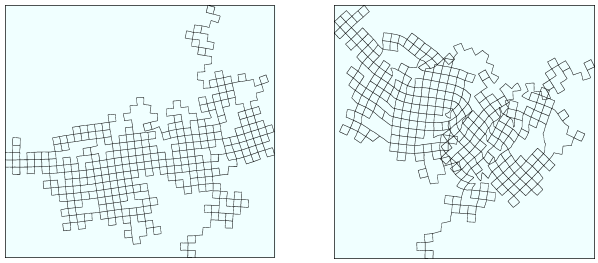


Fig. 7. The result of MLR strongly depends on the initial configuration of the network. Left: small initial pose error, right: large initial pose error.

C. Comparison to MLR and Olson's Algorithm in 2D

In this third experiment, we compare our technique to two current state-of-the-art SLAM approaches that aim to correct constraint networks, namely multi-level relaxation proposed by Frese *et al.* [11] and Olson's algorithm [27]. Since both techniques are designed for 2D scenarios, we also used the 2D version of our system, which is identical to the 3D version except that the three additional dimensions (z, roll, pitch) are not considered.

We furthermore tested two variants of our method: one that uses the node reduction technique described in Section VI and one that maintains all the nodes in the graph.

In these simulation experiments we moved a virtual robot on a grid world. Again, we corrupted the observations with a variable amount of noise for testing the robustness of the algorithms. We simulated different datasets resulting in graphs which consisted of 4,000 and 2,000,000 constraints.

Figure 6 depicts the actual graphs obtained by Olson's algorithm and our approach for different time steps. As can be seen, our approach converges faster. Asymptotically, both approaches converge to a similar solution. In all our experiments, the results of MLR strongly depend on the initial positions of the nodes. We found that in case of a good starting configuration of the network, MLR converges to a highly accurate solution similar to our approach (see left image of Figure 7). Otherwise, it is likely to diverge (right). Olson's approach as well as our technique are more or less independent of the initial poses of the nodes.

To quantitatively evaluate our technique we measured the error in the network after each iteration. The left image of

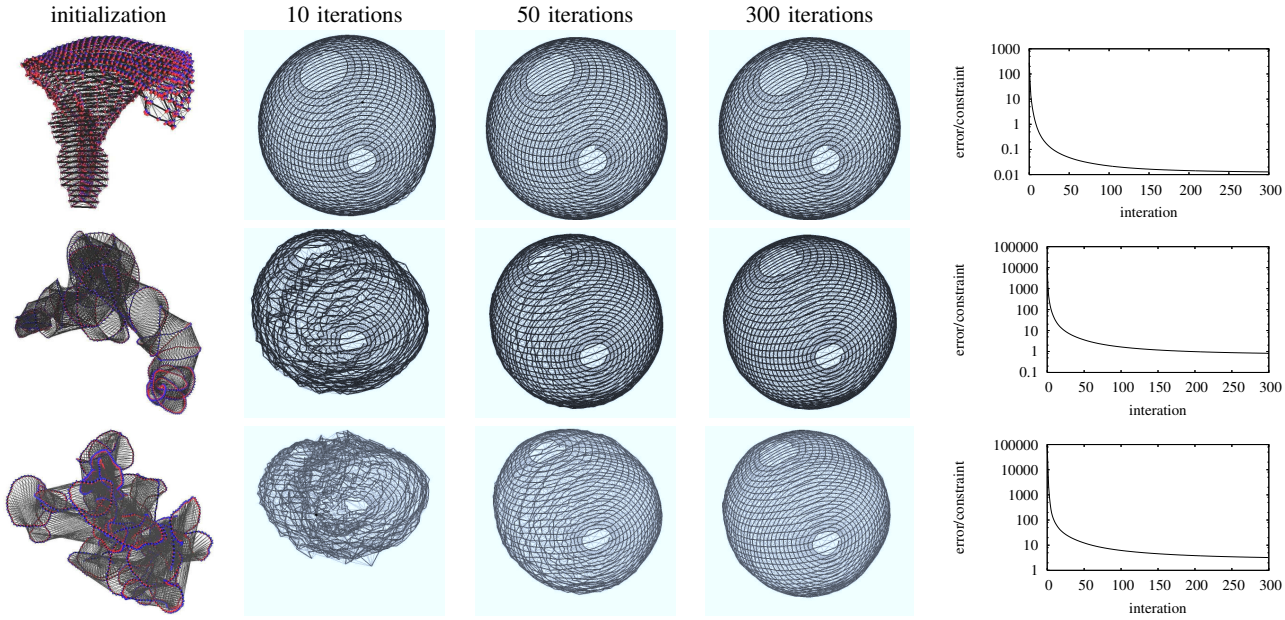


Fig. 5. Results obtained by our approach using a virtual robot moving on a sphere with three different noise realizations in motion and observations (row 1: $\sigma = 0.05$, row 2: $\sigma = 0.1$, row 3: $\sigma = 0.2$). Each network consists of around 85k constraints. The error is computed according to Eq. (7) divided by the number of constraints.

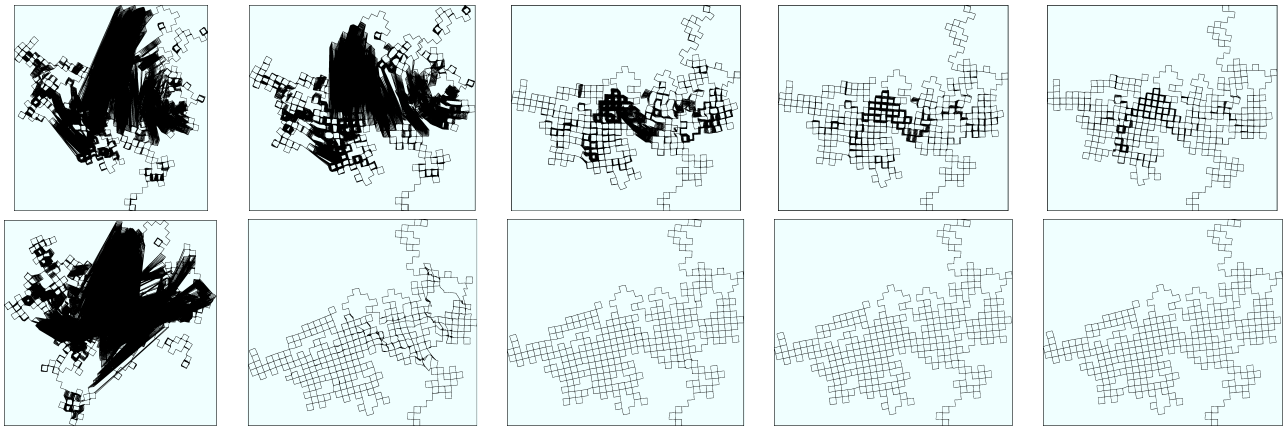


Fig. 6. Results obtained with Olson's algorithm (first row) and our approach (second row) after 1, 10, 50, 100, and 300 iterations for a network with 64,000 constraints. The black areas in the images result from constraints between nodes which are not perfectly corrected after the corresponding iteration (for timings see Figure 8).

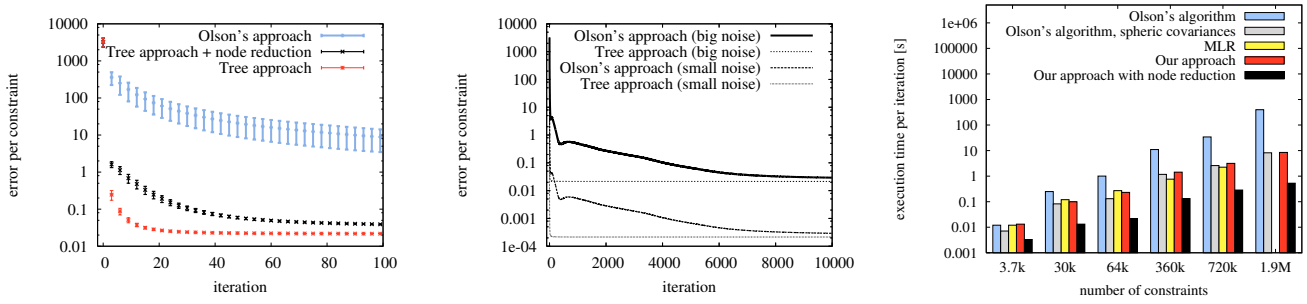


Fig. 8. The left image shows the error of our and Olson's approach in a statistical experiment ($\sigma = 0.05$ confidence). The image in the middle shows that both techniques converge asymptotically to the same error. The right image shows the average execution time *per iteration* for different networks. For the network consisting of 1,900,000 constraints, the executing of MLR required too much resources. The result is therefore omitted. The error is computed according to Eq. (7) divided by the number of constraints.

Figure 8 depicts a statistical experiments over 10 networks with the same topology but different noise realizations. As can be seen, our approach converges significantly faster than the approach of Olson *et al.* For medium size networks, both approaches converge asymptotically to approximatively the same error value (see middle image). For large networks, the high number of iterations needed for Olson's approach prevented us from experimentally analyzing the convergence. For the sake of brevity, we omitted comparisons to EKF and Gauss Seidel relaxation because Olson *et al.* already showed that their approach outperforms those techniques.

Additionally, we evaluated the average computation time per iteration of the different approaches (see right image of Figure 8) and analyzed a variant of Olson's approach which is restricted to spherical covariances. The latter approach yields execution times *per iteration* similar to our algorithm. However, this variant has still the same convergence speed with respect to the number of iterations as Olson's original technique. As can be seen from the right image of Figure 8, our node reduction technique speeds up the computations up to a factor of 20.

We also applied our 3D optimizer to such 2D problems and compared its performance to our 2D version. Both techniques lead to more or less the same results. The 2D version, however, is around three times faster that the 3D version. This results from removing the irrelevant components from the state space and thus avoids the corresponding trigonometric operations.

D. Error Distribution in 3D

We furthermore compared our technique to distribute a rotational error in 3D with our previously proposed method [12]. Compared to this method, our new distribution limits the fraction of the error that is added to the intermediate nodes – a bound that is not available in [12]. Without this bound, it can happen that the error of the overall network drastically increases because a high error is introduced in the intermediate nodes. Note that even if this effect occurs rarely in real datasets, it can lead to divergence. Figure 9 illustrates such an example recorded with a car in a parking lot with three floors.

While the previous method diverges after a few iterations, our new algorithm leads to a limited and balanced distribution of the error. This results in a more stable algorithm, which successfully solved all tested datasets.

E. Constraint Sampling

Stochastic gradient descent selects in each iteration a random order in which the constraints are updated. In our previous work [13], we neglected this randomization and selected a fixed order based on the level of a constraint in the tree. This was needed to perform efficient updates given our previously presented parameterization of the nodes.

With the parameterization presented in this paper, we are free to choose an arbitrary order. We therefore compared two different sampling techniques: random sampling and a variant in which we sample a constraint without replacement with a probability inversely proportional to the path-length.

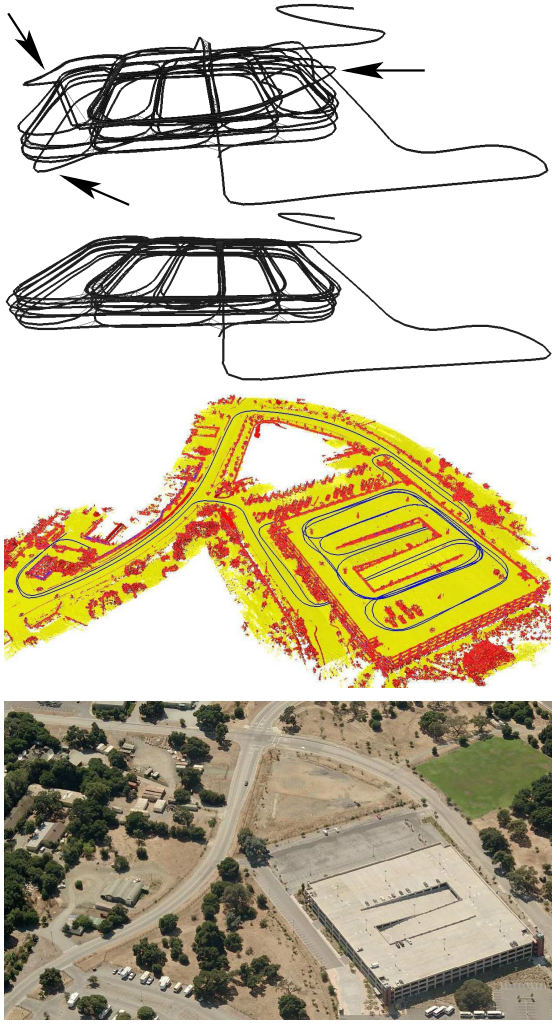


Fig. 9. Network obtained from a car driving multiple times through a parking lot with three floors. Different error distribution techniques result in different networks. The inconsistencies are marked by the arrows. First row: previous method [12], Second row: our approach, both after 3 iterations of the optimizer. Third row: a multi-level surface map created from the corrected constraint network. Fourth row: Aerial image of the parking lot.

We figured out that in situations with nested loops, it is advantageous to process first the constraints which have a shorter path length (and thus correspond to the smaller loops). This is due to angular "wraparounds" that are more likely to occur when first correcting a large loop starting with a poor initial guess. A wraparound is an error in the initial guess of a relative configuration between two nodes that is bigger than 180 degrees. Such wraparounds cause the algorithm to converge to a local minimum.

This effect can be observed in Figure 10. It illustrates a statistical experiment carried out using the sphere dataset (ten runs per strategy). As can be seen, sampling the constraints in each iteration inversely proportional to the length of their path in the tree gives the best results. In contrast to this, getting stuck in local minima is more likely when performing random sampling. Note that this effect occurs only for large networks or high noise in the rotational components. Otherwise, both

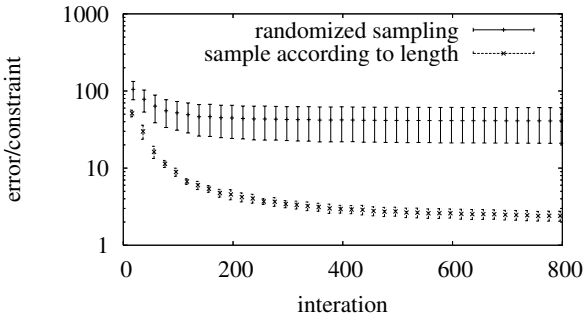


Fig. 10. The evolution of the error per constraint in a statistical experiment using different strategies to sample the constraint that is be updated next. The error is computed according to Eq. (7) divided by the number of constraints.

strategies provide comparable results. As a result, we sample without replacement the constraints in each iteration inverse-proportional to the length of the their path in the parameterization tree.

VIII. Related Work

Mapping techniques for mobile robots can be classified according to the underlying estimation technique. The most popular approaches are extended Kalman filters (EKFs) [21, 31], sparse extended information filters [7, 35], particle filters [23], and least square error minimization approaches [22, 11, 14]. For some applications, it might be even be sufficient to learn local maps only [15, 34, 39].

The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [21, 31]. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. If these assumptions are violated the filter is likely to diverge [18, 37].

Thrun *et al.* [35] proposed a method to correct the poses of a robot based on the inverse of the covariance matrix. The advantage of sparse extended information filters (SEIFs) is that they make use of the approximative sparsity of the information matrix. Eustice *et al.* [7] presented a technique that more accurately computes the error-bounds within the SEIF framework and therefore reduces the risk of becoming overly confident.

Recently, Dellaert and colleagues proposed a smoothing method called square root smoothing and mapping (SAM) [4, 19, 30]. It has several advantages compared to EKF-based solutions since it better covers the non-linearities and is faster to compute. In contrast to SEIFs, it furthermore provides an exactly sparse factorization of the information matrix. In addition to that, SAM can be applied in an incremental way [19] and is able to learn maps in 2D and 3D. Paskin [28] presented a solution to the SLAM problem using thin junction trees. In this way, he is able to reduce the complexity compared to the EKF approaches since thin junction trees provide a linear time filtering operation.

Frese's TreeMap algorithm [9] can be applied to compute nonlinear map estimates. It relies on a strong topological assumption on the map to perform sparsification of the information matrix. This approximation ignores small entries in the information matrix. In this way, Frese is able to perform an update in $\mathcal{O}(\log n)$ where n is the number of features.

An alternative approach to find maximum likelihood maps is the application of least square error minimization. The idea is to compute a network of relations given the sequence of sensor readings. These relations represent the spatial constraints between the poses of the robot. In this paper, we also follow this way of formulating the SLAM problem. Lu and Milios [22] first applied this approach in robotics to address the SLAM problem using a kind of brute force method. Their approach seeks to optimize the whole network at once. Gutmann and Konolige [14] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Howard *et al.* [16] apply relaxation to localize the robot and build a map. Duckett *et al.* [5] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of constraints. To make the problem linear, they assume knowledge about the orientation of the robot. Frese *et al.* [11] propose a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions. MLR is reported to provide very good results in flat environments especially if the error in the initial guess is limited.

Note that techniques such as Olson's algorithm, MLR, or our method focus on computing the best map and assume that the constraints are given. The ATLAS framework [2], hierarchical SLAM [6], or the work of Nüchter *et al.* [26], for example, can be used to obtain the data associations (constraints). They also apply a global optimization procedure to compute a consistent map. One can replace their optimization procedures by our algorithm and in this way make them more efficient.

A technique that combines 2D pose estimates with 3D data has been proposed by Howard *et al.* [17] to build maps of urban environments. They avoid the problem of distributing the error in all three dimensions by correcting only the orientation in the x, y-plane of the vehicle. The roll and pitch is assumed to be measured accurately enough by an IMU.

In the context of three-dimensional maximum likelihood mapping, only a few approaches have been presented so far [24, 25, 26, 36]. The approach of Nüchter *et al.* [26] describes a mobile robot that builds accurate three-dimensional models. In their approach, loop closing is achieved by uniformly distributing the error resulting from odometry over the poses in a loop. This technique provides good estimates but can not deal with multiple/nested loops.

Montemerlo and Thrun [24] proposed to utilize the conjugate gradients to efficiently invert the sparse information matrix of the system. Their approach was used to learn large campus maps using a Segway robot. Recently, Triebel *et al.* [36] described an approach that aims to globally correct the poses given the network of constraints in all six dimensions. At each iteration the problem is linearized and solved using LU decomposition. This yields accurate results for small and medium size networks especially when the error in the rotational component is small. As illustrated in our experimental section, this approach is orders of magnitudes slower than our method and is thus not suited to learn maps of large scenes.

The approach closest to the work presented here is the work of Olson *et al.* [27]. They apply stochastic gradient descent to reduce the error in the network. In contrast to their technique, our approach uses a different parameterization of the nodes in the network that better takes into account the topology of the environment. This results in a faster convergence. Furthermore, our approach allows us to avoid adding new nodes and constraints to the graph when revisiting already mapped areas. As a result, the complexity of our algorithm depends only on the size of the environment and not on the length of the trajectory traveled by the robot. This is an advantage compared to approaches such as MLR or Olson's algorithm since it allows for life-long map learning.

The work presented in this paper furthermore extends two previous conference publications [13, 12]. The first one [13] is only applicable to 2D scenarios and uses a different parameterization of the nodes. The second one is an extension to 3D [12]. It allows a robot to distribute a rotational error over a sequence of poses. This distribution, however, was not bounded as the one presented in this work. As demonstrated in the experimental section, the previous error distribution approach more often leads to divergence.

IX. Conclusion

In this paper, we presented a highly efficient solution to the problem of learning 2D and 3D maximum likelihood maps for mobile robots. Our technique is based on the graph-formulation of the simultaneous localization and mapping problem and applies a variant of stochastic gradient descent. Our approach extends an existing algorithm by introducing a tree-based parameterization for the nodes in the graph. This has a significant influence on the convergence speed and execution time of the method. Furthermore, it enables us to correct arbitrary graphs and not only a list of sequential poses. In this way, the complexity of our method depends on the size of the environment and not directly on the length of the input trajectory. This is an important precondition for lifelong map learning. Additionally, we presented a way to accurately distribute a 3D rotational error over a sequence of poses which increases the robustness over previous approaches.

Our method has been implemented and exhaustively tested in simulation experiments as well as on real robot data. We furthermore compared our method to three existing, state-of-the-art algorithms. The experiments demonstrates that our method converges faster and yields more accurate maps than the other approaches.

Acknowledgment

The authors would like to gratefully thank Udo Frese for his insightful comments and for providing us with his MLR implementation for comparisons. Further thanks go to Edwin Olson for fruitful discussions and to Michael Kaess and Frank Dellaert for carrying out the experiments with their SAM/iSAM implementation. Further thanks go to Slawomir Grzonka for his valuable input on the slerp interpolation used in this work as well as for his support while carrying out experiments. Additionally, we would like to thank Dirk Hähnel and Rainer Kümmerle for providing us with the parking lot dataset recorded with Stanford's autonomous car Junior. Further thanks go to Roland Siegwart and his lab at EPFL and ETH Zürich for the financial and technical support while working with the Smart car. This work has partly been supported by the DFG under contract number SFB/TR-8 (A3) and by the EC under contract number FP6-2005-IST-5-muFly, FP6-2005-IST-6-RAWSEEDS, and FP7-ICT-231888-EUROPA.

References

[1] T. Barrera, A. Hast, and E. Bengtsson. Incremental spherical linear interpolation. In *SIGRAD*, volume 13, pages 7–13, 2004.

[2] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1899–1906, Taipei, Taiwan, 2003.

[3] D. Braid, A. Broggi, and G. Schmiedel. The terramax autonomous vehicle. *Journal on Field Robotics*, 23(9):693–708, 2006.

[4] F. Dellaert. Square Root SAM. In *Proc. of Robotics: Science and Systems (RSS)*, pages 177–184, Cambridge, MA, USA, 2005.

[5] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Journal of Autonomous Robots*, 12(3):287 – 300, 2002.

[6] C. Estrada, J. Neira, and J.D. Tardós. Hierachical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[7] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[8] J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2004.

[9] U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Journal of Autonomous Robots*, 21(2):103–122, 2006.

[10] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17–26, Seattle, WA, USA, 2001.

[11] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[12] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[13] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[14] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, CA, USA, 1999.

[15] J. Hermosillo, C. Pradalier, S. Sekhavat, C. Laugier, and G. Baille. Towards motion autonomy of a bi-steerable car: Experimental issues from map-building to trajectory execution. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[16] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1055–1060, 2001.

[17] A. Howard, D.F. Wolf, and G.S. Sukhatme. Towards 3d mapping in large urban environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 419–424, 2004.

[18] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[19] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[20] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.

[21] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.
[22] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.
[23] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.
[24] M. Montemerlo and S. Thrun. Large-scale robotic 3-d mapping of urban structures. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*, 2004.
[25] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2006.
[26] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.
[27] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.
[28] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1157–1164, Acapulco, Mexico, 2003.
[29] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, W. Burgard, and R. Siegwart. Towards mapping of cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007. Under Review.
[30] A. Ranganathan, M. Kaess, and F. Dellaert. Loopy sam. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.
[31] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.
[32] C. Stachniss and G. Grisetti. TORO project at OpenSLAM.org. http://openslam.org/toro.html, 2007.
[33] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3d using attitude and noisy vision sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.
[34] S. Thrun and colleagues. Winning the darpa grand challenge. *Journal on Field Robotics*, 2006.
[35] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7/8):693–716, 2004.
[36] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.
[37] J. Uhlmann. *Dynamic Map Building and Localization: New Theoretical Foundations*. PhD thesis, University of Oxford, 1995.
[38] C. Urmson. *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2005.
[39] M. Yguel, C.T.M. Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

Giorgio Grisetti is working as a postdoctoral researcher in the Autonomous Intelligent Systems Lab at Freiburg University. He was a Ph.D. student at University of Rome "La Sapienza" in the Intelligent Systems Lab headed by Daniele Nardi where he received his Ph.D. degree in April 2006. His research interests lie in the areas of mobile robotics. His previous and current contributions in robotics aims to provide effective solutions to various mobile robot navigation problems including SLAM, localization, and path planning.

Cyrill Stachniss studied computer science at the University of Freiburg and received his Ph.D. degree in 2006. After his Ph.D., he was a senior researcher at ETH Zurich. Since 2007, he is now an academic advisor at the University of Freiburg in the Laboratory for Autonomous Intelligent Systems. His research interests lie in the areas of robot navigation, exploration, SLAM, as well as learning approaches.

Wolfram Burgard is a professor for computer science at the University of Freiburg where he heads of the Laboratory for Autonomous Intelligent Systems. He received his Ph.D. degree in Computer Science from the University of Bonn in 1991. His areas of interest lie in artificial intelligence and mobile robots. In the past, Wolfram Burgard and his group developed several innovative probabilistic techniques for robot navigation and control. They cover different aspects such as localization, map-building, path-planning, and exploration. For his work, Wolfram Burgard received several best paper awards from outstanding national and international conferences. In 2008, Wolfram Burgard became Fellow of the European Coordinating Committee for Artificial Intelligence.

[J7] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual slam for flying vehicles. *IEEE Transactions on Robotics*, 24(8):1088–1093, 2008.

Visual SLAM for Flying Vehicles

Bastian Steder Giorgio Grisetti Cyrill Stachniss Wolfram Burgard

***Abstract*— The ability to learn a map of the environment is important for numerous types of robotic vehicles. In this paper, we address the problem of learning a visual map of the ground using flying vehicles. We assume that the vehicles are equipped with one or two cheap down-looking cameras in combination with an attitude sensor. Our approach is able to construct a visual map that can later on be used for navigation. Key advantages of our approach are that it is comparably easy to implement, that it can robustly deal with noisy camera images, and that it can operate either with a monocular camera or a stereo camera system. Our technique uses visual features and estimates the correspondences between features using a variant of the PROSAC algorithm. This allows our approach to extract spatial constraints between camera poses which can then be used to address the SLAM problem by applying graph methods. Furthermore, we address the problem of efficiently identifying loop closures. We performed several experiments with flying vehicles which demonstrate that our method is able to construct maps of large outdoor and indoor environments.**

***Index Terms*— SLAM, vision, flying vehicles, attitude sensor**

I. Introduction

The problem of learning maps with mobile robots is a large and active research field in the robotic community. Traditional solutions to the simultaneous localization and mapping (SLAM) problem focus on learning 2D maps of large-scale environments [20]. Also different systems for building 3D maps have been proposed [7, 15, 22]. However, most of these approaches rely on bulky sensors having a high range and accuracy (e.g., SICK laser range finders) which cannot be used on robots such as small flying vehicles. As a result, several researchers focused on utilizing vision sensors instead of laser range finders. Cameras are an attractive alternative due to their limited weight and low power consumption. Existing approaches that address the vision-based SLAM problem mainly focus on scenarios in which a robot repeatedly observes a set of features [6, 14] and they have been shown to learn accurate feature maps.

This paper presents a system that allows aerial vehicles to acquire visual maps of large environments using an attitude sensor and low quality cameras pointing downwards. Such a setup can be found on different air vehicles such as blimps or helicopters. Our system deals with cameras that provide comparably low quality images which are also affected by significant motion blur. Furthermore, it can operate in two different configurations: with a stereo as well as with a monocular camera. If a stereo setup is available, our approach is able to learn visual elevation maps of the ground. If, however, only one camera is carried by the vehicle, our system can be applied by making a flat ground assumption providing a visual map without elevation information. To simplify the problem, we used an attitude (roll and pitch) sensor. In our system, we used an XSens MTi IMU, which has an error below 0.5 degrees. The advantages of our approach is that it is easy to implement, provides robust pose and map estimates, and that is suitable for small flying vehicles. Figure 1 depicts our blimp and helicopter used to evaluate this work as well as an example camera image obtained with our light-weight camera.

II. Related Work

Building maps with robots equipped with perspective cameras has received increasing attention in the last decade. Davison *et al.* [6] proposed a single camera SLAM algorithm based on a Kalman filter. The features are initialized by using a particle filter which estimates their depth. Montiel *et al.* [14] extended this framework by proposing an inverse depth parameterization of the landmarks. Since this parameterization can be better approximated by a Gaussian, the particle filter can be avoided in the initialization of the features. Subsequently, Clemente *et. al* [5] integrated this technique in a hierarchical SLAM framework which has been reported to successfully build large scale maps with comparably poor sensors.

All authors are members of the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

Fig. 1. Two aerial vehicles used to evaluate our mapping approach as well as an example image recorded from an on-board camera.

Chekhlov *et al.* [3] proposed an online visual SLAM framework which uses a SIFT-like feature descriptors and track the 3D motion of a single camera by using an unscented Kalman filter. The computation of the features is speeded up by utilizing the estimated camera position to guess the scale. Jensfelt *et al.* [10] proposed an effective way for online mapping applications by combining a SIFT feature extractor and an interest points tracker. While the feature extraction can be performed at low frequency, the movement of the robot is constantly estimated by tracking the interest points at high frequency.

Other approaches utilize a combination of inertial sensors and cameras. For example, Eustice *et. al* [7] rely on a combination of highly accurate gyroscopes, magnetometers, and pressure sensors to obtain a good estimate for the orientation and altitude of an underwater vehicle. Based on these estimates, they construct an accurate global map using an information filter based on high resolution stereo images. Piniés *et al.* [17] implemented a SLAM system for hand-held monocular cameras and employ an IMU to improve the estimated trajectories. Andreasson *et al.* [1] presented a technique that is based on a local similarity measure for images. They store reference images at different locations and use these references as a map. In this way, their approach is reported to scale well with the size of the environment.

Recently, Konolige and Agrawal [12] presented a technique inspired by scan-matching with laser range finders. These poses of the camera are connected by synthetic measurements obtained from incremental bundle adjustment performed on the images acquired at these poses, and an optimization procedure is used to find the configuration of camera poses which is maximally consistent with the measurement. Our approach uses a similar SLAM formulation but it computes the synthetic measurements between poses based on an efficient pairwise frame alignment technique.

Jung *et al.* [11] proposed a technique which is close to our approach. They use a high resolution stereo camera for building elevation maps with a blimp. The map consists of 3D landmarks extracted from interest points in the stereo image obtained by a Harris corner detector and the map is estimated using an Extended Kalman filter. Due to the wide field of view and the high quality of the images,

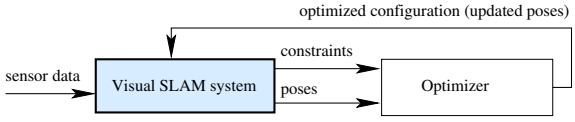


Fig. 2. System overview. This paper describes the visual SLAM system represented by the grayish box.

the non-linearities in the process were adequately solved by the EKF. In contrast to this, our approach is able to deal with low-resolution and low-quality images. It is suitable for mapping indoor and outdoor environments and for operating on small-size flying vehicles. We furthermore apply a more efficient error minimization approach [8] than the Kalman filter which is an extension of the work of Olson *et al.* [16].

Our previous work [19] focused more on the optimization approach and neglected the uncertainty of the vehicle when seeking for loop-closures. In the approach presented in this paper, we consider this uncertainty and provide a significantly improved experimental evaluation using real air vehicles.

III. Graph-based SLAM

In this paper, we address the SLAM problem by using its graph-based formulation. In this framework, the poses of the robot are described by the nodes of a graph. Edges between these nodes represent spatial constrains between them. They are typically constructed from observations or from odometry. Under this formulation, a solution to the SLAM problem is a configuration of the nodes which minimizes the error introduced by the constraints.

We apply an online variant of the 3D optimization technique recently presented by Grisetti *et al.* [8] to compute the maximum likely configuration of the nodes. Online performance is achieved by optimizing only the portions of the graph that require updates after introducing new constraints. Additional speedups result from reusing previously computed solutions to obtain the current one, as explained in [9]. Our system can be used as a black box to which one provides an initial guess of the position of the nodes as well as the edges and it computes the new configuration of the network (see Figure 2). The computed solution minimizes the error introduced by contradicting constraints.

In our approach, each node x_i models a 6DoF camera pose. The spatial constraints between two poses are computed from the camera images and the attitude measurements. An edge between two nodes i and j is represented by the tuple $\langle \delta_{ji}, \Omega_{ji} \rangle$, where δ_{ji} and Ω_{ji} are the mean and the information matrix of the measurement. Let $e_{ji}(\mathbf{x})$ be the error introduced by the constraint $\langle j, i \rangle$. Assuming the independence of the constraints, a solution to the SLAM problem is given by

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i \rangle} e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}). \quad (1)$$

Our approach relies on visual features extracted from the images obtained from two down-looking cameras. We use SURF features [2] which are invariant with respect to rotation and scale. Each feature is represented by a descriptor vector and the position, orientation, and scale in the image. By matching features between different images, one can estimate the relative motion of the camera and thus construct the graph which serves as input to the optimizer. In addition to that, the attitude sensor provides the roll and pitch angle of the camera. In our experiments, we found that the roll and the pitch measurements are comparably accurate even for low-cost sensors and can be directly integrated into the estimate. This reduces the dimensionality of each pose that needs to be estimated from $\mathbb{R}^6$ to $\mathbb{R}^4$.

In this context, the main challenge is to compute the constraints between the nodes (here camera poses) based on the data from the camera and the attitude sensor. Given these constraints, the optimizer processes the incrementally constructed graph to obtain estimates of the most likely configuration on-the-fly.

IV. Spatial Relation between Camera Poses

The input to the optimization approach mentioned in the previous section is a set of poses and constraints between them. In this section, we describe how to determine such constraints.

As a map, we directly use the graph structure of the optimizer. Thus, each camera pose corresponds to one node. Additionally, we store for each node the observed features as well as their 3D positions relative to the node. The constraints between nodes are computed from the features associated with the nodes. In general, at least three pairs of correspondences between image points and their 3D positions in the map are necessary to compute the camera position and orientation [18]. However, in our setting we need only two such pairs since the attitude of the camera is known from the IMU.

In practice, we can distinguish two different situations when extracting constraints: *visual odometry* and *place revisiting*. Odometry describes the relative motion between subsequent poses. To obtain an odometry estimate, we match the features in the current image to the ones stored in the previous n nodes. This situation is easier than place revisiting because the set of potential features correspondences is relatively small. In case of place revisiting, we compare the current features with all the features acquired from robot poses which lie within the 3σ confidence interval given by the pose uncertainty. This interval is computed with the approach of Tipaldi *et. al* [21] and applies covariance intersection on a spanning tree to obtain conservative estimates of the covariances. Since the number of features found during place revisiting can be quite high, we introduce a further approximation in the search procedure. First, we use only a small number of features from the current image when looking for potential correspondences. These features are the one which were better matched when computing visual odometry (which have the lowest descriptor distance). Second, we apply a *k*d-tree to efficiently query for similar features and we use the best-bins-first technique proposed by Lowe [13].

Every time a new image is acquired, we compute the current pose of the camera based on both visual odometry and place revisiting and augment the graph accordingly. The optimization of the graph is performed only if the computed poses are contradictory.

In the remainder of this section, we first describe how to compute a camera pose given a pair of known correspondences, and subsequently we describe our PROSAC-like procedure for determining the best transformation given a set of correspondences between the features in the current image and another set of features computed either via visual odometry or place revisiting.

A. Computing a Transformation from Feature Correspondences

In this section, we explain how to compute the transformation of the camera if we know the 3D position of two features f_1 and f_2 in the map and their projections i_1 and i_2 on the current image. Assuming known camera calibration parameters, we can compute the projections of the points on the normalized image plane. By using the attitude measurements from the IMU, we can compute the positions of these points as they would have been captured from a perfectly downwards facing camera. Let these transformed positions be i'_1, i'_2.

Subsequently, we compute the altitude of the camera according to the procedure illustrated in Figure 3, by exploiting the similarity of triangles. Once the altitude is known, we can compute the yaw of the camera by projecting the map features f_1 and f_2 into the same plane

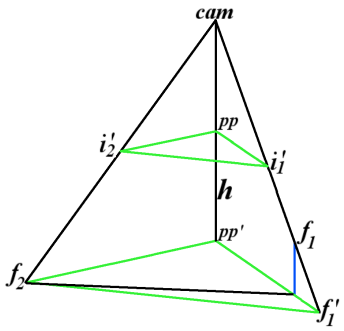


Fig. 3. This figure illustrates how to compute the height of the camera, given two corresponding features, under known attitude. *cam* is the camera position, i_1', i_2' are the projections of the features f_1 and f_2 on the normalized image plane, already rotated according to the attitude measured by the IMU. *pp* is the principle point of the camera (vertically downwards from the camera) on the projection plane. pp' and f_1' are the projections of *pp* and f_1 at the altitude of f_2. h is the altitude difference between the camera and f_2, and it can be determined by exploiting the similarity of the triangles $\{i_1', i_2', pp\}$ and $\{f_1', f_2, pp'\}$.

as i_1', i_2' and then the yaw is the angle between the two resulting lines on this plane.

Finally, we determine x and y as the difference between the positions of the map features and the projections of the corresponding image points, by reprojecting the image features into the map according to the known altitude and yaw angle.

B. Computing the Best Camera Transformation Based on a set of Feature Correspondences

In the previous section, we described how to compute the camera pose given only two correspondences. However, both visual odometry and place revisiting return a set of correspondences. In the following, we describe our procedure to efficiently select from the input set the pair of correspondences for computing the most likely camera transformation.

We first order these correspondences according to the Euclidean distance of their descriptor vectors. Let this ordered set be $C = \{c_1, ..., c_n\}$. Then we select pairs of correspondences in the order defined by the following predicate:

$$\langle c_{a_1}, c_{b_1}\rangle < \langle c_{a_2}, c_{b_2}\rangle \quad \Leftrightarrow \quad (b_1 < b_2 \vee (b_1 = b_2 \wedge a_1 < a_2)) \wedge a_1 < b_1 \wedge a_2 < b_2. \tag{2}$$

In this way, the best correspondences (according to the descriptor distance) are used first but the search procedure will not get stuck for a long time in case of false matches with low descriptor distances. This is illustrated in the following example: assume that the first correspondence c_1 is a false match. Our selection strategy generates the sequence $\langle c_1, c_2\rangle, \langle c_1, c_3\rangle, \langle c_2, c_3\rangle, \langle c_1, c_4\rangle, \langle c_2, c_4\rangle, \langle c_3, c_4\rangle$ ..... A pair without the false match $\langle c_2, c_3\rangle$ will be selected in the third step. A more naive selection strategy will try first all pairs of correspondences $\langle c_1, c_x\rangle$ with c_1 in the first position, and results in a less efficient search.

Only pairs that involve different features are used. The corresponding transformation T_{c_a,c_b} is then determined for the current pair (see section IV-A). This transformation is then evaluated based on the other features in both sets using a score function, which is presented in the next subsection. The process can be stopped, when a transformation with a satisfying score is found or when a timeout is reached. The solution with the highest score is returned as the current assumption for the transformation.

C. Evaluating a Camera Transformation

In the previous sections, we explained how to compute a camera transformation based on two pairs of corresponding features, and how to select those pairs from two input sets of features. By using different pairs, we can compute a set of candidate transformations. In this section, we explain how to evaluate them and how to choose the best one among them.

To select the best transformation, we rank them according to a score function. The score is computed by projecting the features in the map into the current camera image and by then comparing the distance between the feature positions in the image. The score is given by

$$\text{score}(T_{c_a,c_b}) \quad = \sum_{\{i \,|\, i \notin \{a,b\}\}} v(c_i). \tag{3}$$

In this equation, the function $v(c_i)$ is defined as the weighted sum of the relative displacement of the corresponding features in the current image and the Euclidean distance of their feature descriptors:

$$v(c_i) = 1 - \left[\alpha \frac{d^{\text{desc}}(c_i)}{d^{\text{desc}}_{\max}} + (1-\alpha)\frac{d^{\text{img}}(c_i)}{d^{\text{img}}_{\max}}\right] \tag{4}$$

In the sum of Eq. (3), we consider only those feature correspondences c_i whose distances $d^{\text{img}}(c_i)$ in the image and distances $d^{\text{desc}}(c_i)$ in the descriptor space are smaller than the thresholds $d^{\text{img}}_{\max}$ and $d^{\text{desc}}_{\max}$ introduced in Eq. (4). This prevents single outliers from leading to overly bad scores.

More in detail, $d^{\text{img}}_{\max}$ is the maximum distance in pixels between the original and the re-projected feature. In our experiments this value was set to 2 pixels for images of 320×240 pixels. The higher the motion blur in the image the larger this value should be set. The minimum value depends on the accuracy of the feature extractor. Increasing this threshold also allows the matching procedure to return less accurate solutions for the position estimation. The blending factor α mixes the contribution of the descriptor distance and the re-projection error. The more distinct the features, are the higher alpha can be chosen. In all our experiments, we set $\alpha = 0.5$. The value $d^{\text{desc}}_{\max}$ has been manually tuned. When using 64-dimensional SURF descriptors we had good results by setting this threshold to values around 0.3. The lower the quality of the image, the higher $d^{\text{desc}}_{\max}$ should be chosen.

Note that the technique to identify the correspondences between images is similar to the PROSAC [4] algorithm which is a variant of RANSAC. PROSAC takes into account a quality measure of the correspondences while sampling, conversely RANSAC draws the samples uniformly. We use the distance between feature descriptors as a quality measure. In our variant of PROSAC, the correspondences are selected deterministically. Since we only need two correspondences to compute the camera transformation, the chances that the algorithm gets stuck due to wrong correspondences are very small.

After identifying the transformation between the current pose of the camera and a node in the map, we can directly add a constraint to the graph. In the subsequent iteration of the optimizer, the constraint is thus taken into account when computing the updated positions of the nodes.

V. Experiments

In this section, we present the experiments carried out to evaluate our approach. We used only real world data which we partially recorded with a sensor platform carried in the hand of a person as well as with a real blimp and a helicopter (see Figure 1). In all experiments our system was running at 5 to 15 hertz on a 2.4 GHz Dual-core. Videos of the experiments can be downloaded at `http://www.informatik.uni-freiburg.de/~steder/homepage/videos`.

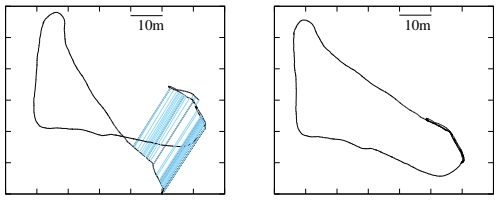


Fig. 4. The left image shows the path of the camera in black and the matching constraints in gray. The right image shows the corrected trajectory after applying the optimization technique.

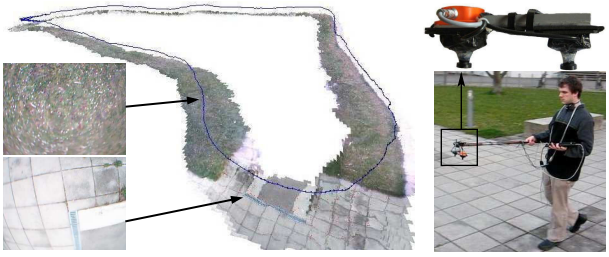


Fig. 5. The left image shows a perspective view of the map of the outdoor experiment together with two camera images recorded at the corresponding locations. The right one shows a person with the sensor platform mounted on a rod to simulate a freely floating vehicle.

A. Outdoor Environments

In the first experiment, we measured the performance of our algorithm using data recorded in outdoor environments. Since even calm winds outside buildings prevent us from making outdoor experiments with our blimp or our small size helicopter, we mounted a sensor platform on the tip of a rod and carried this by hand to simulate a freely floating vehicle. This sensor platform is equipped with two standard Web cams (Logitech Communicate STX). The person carried the platform along a long path around a building over different types of ground like grass and pavement. The trajectory has a length of about 190 m. The final graph contains approximately 1400 nodes and 1600 constraints. The trajectory resulting from the visual odometry is illustrated in the left image of Figure 4. Our system autonomously extracted data association hypotheses and constructed the graph. These matching constraints are colored light blue/gray in the same image. After applying our optimization technique, we obtained a map in which the loop has been closed successfully. The corrected trajectory is shown in the right image of Figure 4. A perspective view, which also shows the elevations, is depicted in Figure 5.

This experiment illustrates that our approach is able to build maps of comparably large environments and that it is able to find the correct correspondences between observations. Note that this result has been achieved without any odometry information and despite the fact that the cameras are of low quality and that the images are blurry due to the motion and mostly show grass and concrete.

B. Statistical Experiments

The second experiment evaluates the performance of our approach quantitatively in an indoor environment. The data was acquired with the same sensor setup as in the previous experiment. We moved in the corridor of our building which has a wooden floor. For a statistical evaluation of the accuracy of our approach, we placed artifical objects on the ground at known locations. We measured their locations manually with a measuring tape (up to an accuracy of approximately 3 cm). The distance in the x coordinate between neighboring landmarks is 5 m and 1.5 m in the y direction. The six

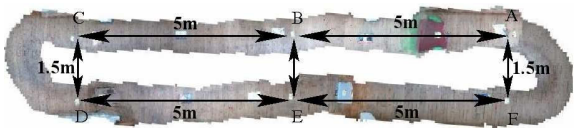


Fig. 6. Top view of the map of the indoor experiment. The image shows the map after least square error minimization. The labels A to F present six landmarks for which we determined the ground truth location manually to evaluate the accuracy of our approach.

TABLE I
ACCURACY OF THE RELATIVE POSE ESTIMATE BETWEEN LANDMARKS

landmarks	A-B	B-C	C-D	D-E	E-F	F-A	loop
mean error [m]	0.18	0.26	0.11	0.20	0.21	0.12	1.10
sigma [m]	0.21	0.32	0.12	0.39	0.3	0.15	1.25
error [%]	3.6	5.2	7.3	4.0	4.2	8.0	4.8

landmarks are labeled A to F in Figure 6. We used these six known locations as ground truth, which allowed us to measure the accuracy of our mapping technique. Figure 6 depicts a resulting map after applying the least square error minimization approach. We repeated the experiment 10 times and measured the relative distance between them.

Table I summarizes this experiment. As can be seen, the error of the relative pose estimates is always below 8% and typically around 5% compared to the true difference. This results mainly from the error in our self-made and comparably low quality stereo setup. To our opinion, this is an accurate estimate for a system consisting of two cheap cameras and an IMU, lacking sonar, laser range data, and real odometry information.

C. Experiments with a Blimp

The third experiment is also a statistical analysis carried out with our blimp. The blimp has only one camera looking downwards. Instead of the stereo setup, we mounted a sonar sensor to measure its altitude. Furthermore, no attitude sensor was available and we therefore assumed the roll and pitch angle to be zero (which is an acceptable approximation given the smooth motion of a blimp). We placed two landmarks on the ground with a distance of 5 m and flew 10 times over the scene. The mean estimated distance between the two landmarks was 4.91 m with a standard deviation of 0.11 m. Thus, the real position was within the 1σ interval.

The next experiment in this paper is designed to illustrate that such a visual map can be used for navigation. We constructed the map shown in Figure 7 with our blimp. During this task, the blimp was instructed to return always to the same location and was repeatedly pushed away several meters. The blimp was always able to register its current camera image against the map constructed so far and in this way kept track of its location relative to the map. This enabled the controller of the blimp to steer the air vehicle to the desired location. The experiment lasted 18 min and the blimp recorded during that time around 10,800 images. The robot processed around 500,000 features and the map was constructed online.

D. Experiments with a Light-weight Helicopter

We finally mounted an analog RF-camera on our light-weight helicopter depicted in Figure 1. This helicopter is not equipped with an attitude sensor nor with a sonar sensor to measure its altitude. Since neither stereo information nor the elevation of the helicopter is known, the scale of the visual map was determined by a known size of one landmark (a book lying on the ground). Furthermore, the attitude was assumed to be zero which is a quite rough approximation

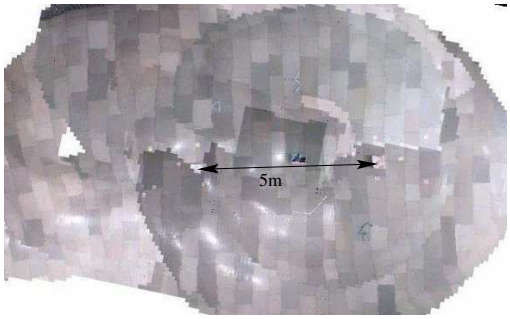


Fig. 7. Map constructed by the blimp. The ground truth distance between both landmarks is 5 m and the estimated distance was 4.91 m with 0.11 m standard deviation (10 runs). The map was used to autonomously steer the blimp to user specified locations.

Fig. 8. A person pushes the blimp away. The blimp is able to localize itself and navigate back using the map shown in Figure 7 (see video material).

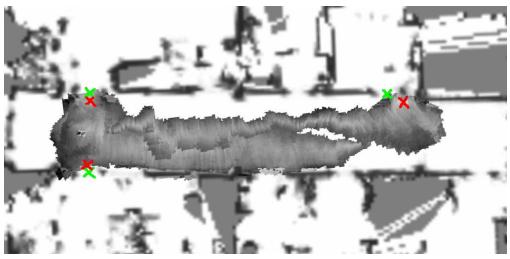


Fig. 9. Visual map build with a helicopter overlayed on a 2D grid map constructed from laser range finder data recorded with a wheeled robot.

for a helicopter. We recorded a dataset by flying the helicopter and overlayed the resulting map with an occupancy grid map recorded from laser range data with a wheeled robot. Figure 9 depicts the result. The red and green crosses indicate the same locations in the occupancy grid map and the visual map. Even under the hard sensory limitations, our approach was able to estimate its position in a quite accurate manner. The helicopter flew a distance of around 35 m and the map has an error in the landmark locations that varies between 20 cm and 60 cm.

VI. Conclusions

In this paper, we presented a robust and practical approach to learn visual maps based on down looking cameras and an attitude sensor. Our approach applies a robust feature matching technique based on a variant of the PROSAC algorithm in combination with SURF features. The main advantages of the proposed methods are that it can operate with monocular or with a stereo camera system, that it is easy to implement, and that it is robust to noise in the camera images.

We presented a series of real world experiments carried out with a small-size helicopter, a blimp, and by manually carrying a sensor platform. Different statistical evaluations of our approach show its ability to learn consistent maps of comparably large indoor and outdoor environments. We furthermore illustrated that such maps can be used for navigation tasks of air vehicles.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8 and by the EC under contract numbers FP6-2005-IST-6-RAWSEEDS and FP6-IST-34120-muFly.

References

[1] H. Andreasson, T. Duckett, and A. Lilienthal. Mini-SLAM: Minimalistic visual SLAM in large-scale environments based on a new interpretation of image similarity. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006.

[3] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and Calway A. Robust real-time visual slam using scale prediction and exemplar based feature description. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, usa, 2007.

[4] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, USA, 2005.

[5] L.A. Clemente, A. Davison, I. Reid, J. Neira, and J.D. Tardós. Mapping large loops with a single hand-held camera. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[6] A. Davison, I. Reid, , N. Molton, and O. Stasse. MonoSLAM:real time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[7] R.M. Eustice, H. Singh, J.J. Leonard, and M.R. Walter. Visually mapping the RMS Titanic: conservative covariance estimates for SLAM information filters. *Int. Journal of Robotics Research*, 25(12), 2006.

[8] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[9] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[10] P. Jensfelt, D. Kragic, J. Folkesson, and M. Björkman. A framework for vision based bearing only 3D SLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, CA, 2006.

[11] I. Jung and S. Lacroix. High resolution terrain mapping using low altitude stereo imagery. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Nice, France, 2003.

[12] K. Konolige and M. Agrawal. Frame-frame matching for realtime consistent visual mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[13] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Krekyra, Greece, 1999.

[14] J.M. Montiel, J. Civera, and A.J. Davison. Unified inverse depth parameterization for monocular SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2006.

[15] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[16] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[17] P. Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós. Inertial aiding of inverse depth slam using a monocular camera. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2797–2802, Rome, Italy, 2007.

[18] L. Quan and Z.-D. Lan. Linear N-Point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999.

[19] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3D using attitude and noisy vision sensors. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[20] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[21] G.D. Tipaldi, G. Grisetti, and W. Burgard. Approximate covariance estimation in graphical approaches to SLAM. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[22] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[J8] C. Stachniss, G. Grisetti, O. Martínez-Mozos, and W. Burgard. Efficiently learning metric and topological maps with autonomous service robots. *it – Information Technology*, 49(4):232–238, 2007. Extended version.

Efficiently Learning Metric and Topological Maps with Autonomous Service Robots

Cyrill Stachniss Giorgio Grisetti Óscar Martínez Mozos Wolfram Burgard

University of Freiburg, Dept. of Computer Science, Georges-Köhler-Allee 79, 79110 Freiburg, Germany

***Abstract*— Models of the environment are needed for a wide range of robotic applications, from search and rescue to automated vacuum cleaning. Learning maps has therefore been a major research focus in the robotics community over the last decades. In general, one distinguishes between metric and topological maps. Metric maps model the environment based on grids or geometric representations whereas topological maps model the structure of the environment using a graph.**

The contribution of this paper is an approach that learns a metric as well as a topological map based on laser range data obtained with a mobile robot. Our approach consists of two steps. First, the robots solves the simultaneous localization and mapping problem using an efficient probabilistic filtering technique. In a second step, it acquires semantic information about the environment using machine learning techniques. This semantic information allows the robot to distinguish between different types of places like, e.g., corridors or rooms. This enables the robot to construct annotated metric as well as topological maps of the environment. All techniques have been implemented and thoroughly tested using real mobile robot in a variety of environments.

I. Introduction

The problem of learning maps is one of the fundamental problems in mobile robotics. Models are needed for a series of applications like transportation, cleaning, rescue, localization, and various other service tasks. Learning maps has therefore been a major research issue in the robotics community over the last decades.

Typically, one distinguishes between the type of model the mapping approach learns: metric or topological maps. Metric maps like, for example, occupancy, feature, or geometric maps model the objects observed by the sensor. These maps are often used to explicitely represent obstacles and driveable areas. Typically, the resulting model strongly depends on the sensors used by the robot to perceive its environment. Metric maps often bear resemblance to floor plans used in architecture.

For different robotic tasks, however, the robot can improve its capabilities or performance when sematic or topological information is available. In contrast to metric maps, topological maps model the structure of the environment using a graph. The different places in the environment are represented by nodes in that graph. Topological maps are quite popular in the robotics community because they are believed to be cognitively more adequate. Compared to metric maps, they can be stored in a compact manner and can facilitate the communication with the users.

While most other mapping approaches address metric or topological map learning, we focus in this paper on constructing a metric as well as a topological model of the environment. This enables a mobile robot to use the best suited model for the task it performs. Our approach consists of two steps. In the first one, we apply a highly efficient particle filter to solve the *simultaneous localization and mapping* (SLAM) problem. This step is based on grid maps and eliminates the pose uncertainty of the robot. In the second step, we use the grid resulting from the first step in order to learn the topology. Our technique estimates semantic information about local areas using the AdaBoost algorithm. It furthermore applies probabilistic relaxation labeling to smooth the semantic labels and then identifies distinct places based on that data. Finally, this allows a mobile robot to learn accurate metric models of the environment while at the same time constructing a consistent topological map.

The reminder of this paper is organized as follows. We describe the two steps of our algorithm in the next sections. Section II presents the first step in which a Rao-Blackwellized particle filter is applied to eliminate the pose uncertainty and to construct a metric grid map. Based on this result, Section III describes the second step of our technique which is the extraction of the topological information. Section IV presents results obtained by our approach and finally, we discuss related work in Section V.

II. Step 1: Efficient Metric Mapping

This section describes the first step of our mapping approach. The goal is to eliminate the pose uncertainty and to obtain a consistent grid representation. According to Murphy [24], the key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the map m and the trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot. This estimation is performed given the observations $z_{1:t} = z_1, \ldots, z_t$ and the odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$ obtained by the mobile robot. The Rao-Blackwellized particle filter for SLAM makes use of the following factorization

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}). \quad (1)$$

This factorization allows us to first estimate only the trajectory of the robot and then to compute the map given that trajectory. Since the map strongly depends on the pose estimate of the robot, this factorized posterior can be computed efficiently. This technique is often referred to as Rao-Blackwellization.

The posterior over maps $p(m \mid x_{1:t}, z_{1:t})$ in Eq. (1) can be computed analytically using "mapping with known poses" [23]

since $x_{1:t}$ and $z_{1:t}$ are known. To estimate the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ over the potential trajectories, one can apply a particle filter. Each particle represents a potential trajectory of the robot. Furthermore, an individual map is associated with each sample. The maps are built from the observations and the trajectory represented by the corresponding particle.

One of the most common particle filtering algorithms is the sampling importance resampling (SIR) filter. A Rao-Blackwellized SIR filter for mapping can be summarized by the following four steps:

1) *Sampling*: The next generation of particles $\{x_t^{(i)}\}$ is obtained from the generation $\{x_{t-1}^{(i)}\}$ by sampling from the so-called proposal distribution π. Often, a probabilistic odometry motion model is used as the proposal distribution.
2) *Importance Weighting*: An individual importance weight $w_t^{(i)}$ is assigned to each particle according to the importance sampling principle

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)} \mid z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)} \mid z_{1:t}, u_{1:t-1})}. \quad (2)$$

   The weights account for the fact that the proposal distribution π is in general not equal to the target distribution of successor states.
3) *Resampling*: Particles are drawn with replacement proportional to their importance weight. This step is necessary since only a finite number of particles is used to approximate a continuous distribution. Furthermore, resampling allows us to apply a particle filter in situations in which the target distribution differs from the proposal. After resampling, all the particles have the same weight.
4) *Map Estimation*: For each particle, the corresponding map estimate $p(m^{(i)} \mid x_{1:t}^{(i)}, z_{1:t})$ is computed based on the trajectory $x_{1:t}^{(i)}$ of that sample and the history of observations $z_{1:t}$.

A. Computing an Improved Proposal Distribution

The general framework for mapping with Rao-Blackwellized particle filters leaves open how the proposal distribution is computed. In general, the filter produces more accurate results the closer the proposal approximates the target distribution. The target distribution, however, is typically not available in a closed form solution suitable for sampling. In our case, the target distribution is given by

$$\begin{aligned} p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) &= \eta \cdot p(z_t \mid m_{t-1}^{(i)}, x_t) \cdot p(x_t \mid x_{t-1}, u_{t-1}) \\ &\quad \cdot p(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-2}), \end{aligned} \quad (3)$$

where η is a normalizing constant resulting from Bayes' rule.

Our approach uses the laser range observations of the robot in order to approximate the target as close as possible while being able to efficiently sample from that distribution. The advantage of this approach lies in the fact that the laser range observations are typically affected by significantly less noise compared to the odometry of the robot (which is used as the proposal in classical particle filter applications). This fact is illustrated in Figure 1. Since the resulting distribution is given by the product of the motion and the observation model, one can restrict the search to areas of high likelihood (called meaningful area $L^{(i)}$, see Figure 1). In the reminder of this section, we derive our proposal distribution. According to Doucet [4], the distribution

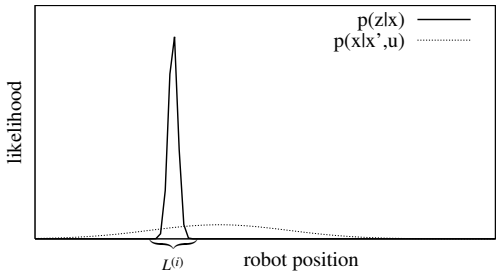


Fig. 1. The two components of the target distribution. Within the interval $L^{(i)}$ the product of both functions is dominated by the observation likelihood. The observation likelihood is therefore well-suited to focus the proposal to the interval $L^{(i)}$.

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \quad (4)$$

is the optimal proposal distribution with respect to the variance of the particle weights. When modeling the environment with grid maps, a closed form approximation of an informed proposal is also not directly available due to the unpredictable shape of the observation likelihood function.

One of our observations is that in the majority of cases the target distribution has only a limited number of maxima and it mostly has only a single one. This allows the robot to sample positions x_j covering only the area surrounding this maximum. Ignoring the less meaningful regions of the distribution saves a significant amount of computational resources since it requires less samples.

In our current approach, we consider both components of the proposal, the observation likelihood and the motion model within the meaningful interval $L^{(i)}$. We locally approximate the posterior $p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})$ around the maximum of the likelihood function reported by a scan registration procedure.

To efficiently draw the next generation of samples, we compute a Gaussian approximation based on that data. The main difference to previous approaches is that we first use a scan-matcher to determine the meaningful area of the observation likelihood function. We then sample in that meaningful area and evaluate the sampled points based on the target distribution. For each particle i, the parameters $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ are determined individually for K sampled points $\{x_j\}$ in the interval $L^{(i)}$. We furthermore take into account the odometry information when computing the mean $\mu^{(i)}$ and the variance $\Sigma^{(i)}$. We estimate the Gaussian parameters as

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} x_j \cdot p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \quad (5)$$

$$\begin{aligned} \Sigma_t^{(i)} &= \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \\ &\quad \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T \end{aligned} \quad (6)$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}). \quad (7)$$

In this way, we obtain a closed form approximation of the optimal proposal which enables the robot to efficiently obtain the next generation of particles. Using this proposal distribution, the weights can be computed as

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{\xi \cdot p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} & (8)\\ &\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} & (9)\\ &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) & (10)\\ &= w_{t-1}^{(i)} \cdot \int p(z_t \mid x') p(x' \mid x_{t-1}^{(i)}, u_{t-1})\, dx' & (11)\\ &\simeq w_{t-1}^{(i)} \cdot \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) & (12)\\ &= w_{t-1}^{(i)} \cdot \eta^{(i)}. & (13) \end{aligned}$$

Note that $\eta^{(i)}$ is the same normalization factor that is used in the computation of the Gaussian approximation of the proposal in Eq. (7) and ξ is a normalizer resulting from Bayes' rule. The computations presented in this section enable us to determine the parameters of a Gaussian proposal distribution for each particle individually. The proposal takes into account the most recent odometry reading and laser observation while at the same time allowing us efficient sampling. The resulting densities have a much lower uncertainty compared to situations in which the odometry motion model is used.

As explained above, we use a scan-matcher to determine the mode of the meaningful area of the observation likelihood function. In this way, we focus the sampling on the important regions. Most existing scan-matching algorithms maximize the observation likelihood given a map and an initial guess of the robot's pose. When the likelihood function is multi-modal, which can occur when, e.g., closing a loop, the scan-matcher returns for each particle the maximum which is closest to the initial guess. In general, it can happen that additional maxima in the likelihood function are missed since only a single mode is reported. However, since we perform frequent filter updates (after each movement of $0.5\,m$ or a rotation of $25°$) and limit the search area of the scan-matcher, we consider that the distribution has only a single mode when sampling data points to compute the Gaussian proposal. Note that in situations like a loop closure, the filter is still able to keep multiple hypotheses because the initial guess for the starting position of the scan-matcher when reentering a loop is different for each particle.

During filtering, it can happen that the scan-matching process fails because of poor observations or a too small overlapping area between the current scan and the previously computed map. In our system, this is detected by monitoring the observation likelihood in the matching process and by applying a threshold criterion. In this case, the raw motion model of the robot is used as a proposal. Note that such situations occur rarely in real datasets.

When modeling a mobile robot equipped with an accurate sensor like, a laser range finder, it is convenient to use such an improved proposal since the accuracy of the laser range finder leads to extremely peaked likelihood functions. In the context of landmark-based SLAM, Montemerlo *et al.* [20] presented a Rao-Blackwellized particle filter that uses a Gaussian approximation of the improved proposal. This Gaussian is computed for each particle using a Kalman filter that estimates the pose of the robot. This approach can be used when the map is represented by a set of features and if the error affecting the feature detection is assumed to be Gaussian. In this work, we transfer the idea of computing an improved proposal to the situation in which dense grid maps are used instead of landmark-based representations.

B. Adaptive Resampling

A further aspect that has a major influence on the performance of a particle filter is the resampling step. During resampling, particles with a low importance weight $w^{(i)}$ are typically replaced by samples with a high weight. On the one hand, resampling is necessary since only a finite number of particles are used to approximate the target distribution. On the other hand, the resampling step can remove good samples from the filter which can lead to particle impoverishment. Accordingly, it is important to find a criterion for deciding when to perform the resampling step. Liu [17] introduced the so-called effective sample size to estimate how well the current particle set represents the target posterior. In this work, we compute this quantity according to the formulation of Doucet *et al.* [6] as

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}^{(i)}\right)^2}, \quad (14)$$

where $\tilde{w}^{(i)}$ refers to the normalized weight of particle i.

The intuition behind N_{eff} is as follows. If the samples were drawn from the target distribution, their importance weights would be equal to each other due to the importance sampling principle. The worse the approximation of the target distribution, the higher is the variance of the importance weights. Since N_{eff} can be regarded as a measure of the dispersion of the importance weights, it is a useful measure to evaluate how well the particle set approximates the target posterior. Our algorithm follows the approach proposed by Doucet *et al.* [6] to determine whether or not the resampling step should be carried out. We resample each time N_{eff} drops below the threshold of $N/2$ where N is the number of particles. In extensive experiments, we found that this approach drastically reduces the risk of replacing good particles, because the number of resampling operations is reduced and they are only performed when needed.

III. Step 2: Estimating the Topology

In the previous section, we presented a way for learning accurate metric maps of the environment. The results of this first step are now used to build a topological representation

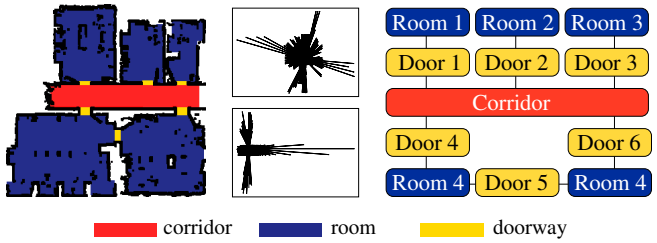


Fig. 2. The left image shows a geometric map of a typical indoor environment with rooms, doorways, and a corridor depicted in colors/grey levels. The middle images show two simulated range scans in the geometric map. The right image depicts the corresponding semantic-topological map.

of the environment. The approach is based on the assumption that indoor environments can typically be decomposed into areas with different functionalities such as rooms, corridors, and doorways. From these areas, we create the node of the topological graph. The edges between the nodes are then given by the neighborhood relation of the regions in the occupancy map. For example, a doorway is typically connected to two rooms, two corridors, or to a room and a corridor.

To obtain such a topology, our approach determines the semantic class for each unoccupied cell of the grid. This is achieved by simulating a range scan given the sensor is located in that particular cell and then classifying this scan into one of the semantic classes. Examples for typical simulated range scans obtained in an office environment are shown in the middle images of Figure 2. The classification is then done using a sequence of classifiers learned with the AdaBoost algorithm [27]. These classifiers are learned in a supervised fashion from simple geometric features that are extracted from range scans simulated in different, previously labeled maps of standard environment. To remove noise and clutter from the resulting classifications, we apply an approach denoted as probabilistic relaxation labeling [25]. From the resulting labeling, we construct a graph whose nodes correspond to the regions of identically labeled poses and whose edges represent the connections between them. Additionally, each node contains geometrical information about the region it represents, like the area, the centroid and the orientation. A typical topological map obtained with our approach is shown in the right image of Figure 2.

A. Semantic Classification of Locations

Boosting is a general method for creating an accurate strong classifier by combining a set of weak classifiers. The requirement to each weak classifier is that its accuracy is better than a random guessing. AdaBoost selects and arranges the best weak classifiers h_j and combines them to a strong classifier by weighted majority voting. In this work, we will use the boosting algorithm AdaBoost in its generalized form presented by Schapire and Singer [27]. The input to the algorithm is a set of labeled training examples $(e_n, y_n), n = 1, \ldots, N$, where each e_n is an example and each $y_n \in \{+1, -1\}$ is a value indicating whether e_n is positive or negative respectively. In our case, the training examples are composed by simulated laser observations.

Throughout this work, we will use the approach by Viola and Jones [32] in which each weak classifier h_j depends on a single-valued feature $f_j \in \mathbb{R}$

$$h_j(e) = \begin{cases} +1 & \text{if } p_j f_j(e) < p_j \theta_j \\ -1 & \text{otherwise,} \end{cases} \tag{15}$$

where θ_j is a threshold and p_j is either -1 or $+1$ and thus represents the direction of the inequality. The parameters θ_j and p_j are determined during the training process of AdaBoost.

The generalized AdaBoost is only able to predict the label of an example as positive or negative. To additionally estimate the probability of a particular label, we use the method suggested by Friedman *et al.* [8]. It uses the output of AdaBoost to determine a confidence value $C \in [0, 1]$ for a positive classification of an example $C = P(y = +1 \mid e)$.

AdaBoost distinguishes between two classes only. In practical applications, however, one often needs to distinguish between more than two classes. To create a multi-class classifier, we create a sequential multi-class classifier by using $K - 1$ binary classifiers, where K is the number of classes we want to recognize (see [18] for further details). Additionally, we use the method by Stachniss *et al.* [30], in which the classification output of the decision list is represented by a histogram z. These histograms can be seen as high level observations. Each bin of z stores the probability that the classified example belongs to the k-th class according to the sequence

$$z^{[k]} = C_k \prod_{j=1}^{k-1} (1 - C_j), \tag{16}$$

where $C_k = P_k(y = +1 \mid e)$ according to the approach of Friedman *et al.* [8].

In order to build such a classifier, one needs to extract features f_j which are used in the weak classifiers. In this work, we use geometric, single-valued features computed based on the laser range observation. These features are rotationally invariant and frequently used in shape analysis. Examples are the area covered by the scan or the average beam length. A full list of features can be found in [18].

B. Probabilistic Relaxation Labeling

One of the key problems that needs to be solved in order to learn accurate topological maps, in which the nodes correspond to the individual parts in the environment, is to eliminate classification errors. In this section, we describe probabilistic relaxation labeling [25] to smooth the AdaBoost classifications based on neighborhood relations.

Probabilistic relaxation labeling is defined as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph consisting of nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Let furthermore $\mathcal{L} = \{l_1, \ldots, l_L\}$ be a set of labels. We assume that every node v_i stores a probability distribution about its label which is represented by a histogram P_i. Each bin $p_i(l)$ of that histogram stores the probability that the node v_i has the label l. Thus, $\sum_{l=1}^{L} p_i(l) = 1$.

For each node v_i, $\mathcal{N}(v_i) \subset \mathcal{V}$ denotes its neighborhood which consists of the nodes $v_j \neq v_i$ that are connected to v_i. Each neighborhood relation is represented by two values. Whereas the first one describes the compatibility between the labels of two nodes, the second one represents the influence between the two nodes. The term $\mathcal{R} = \{r_{ij}(l, l') \mid v_j \in \mathcal{N}(v_i)\}$

defines the compatibility coefficients between the label l of node v_i and the label l' of v_j. And $C = \{c_{ij} \mid v_j \in \mathcal{N}(v_i)\}$ is the set of weights indicating the influence of node v_j on node v_i.

Given an initial estimation for the probability distribution over labels $p_i^{(0)}(l)$ for the node v_i, the probabilistic relaxation method iteratively computes estimates $p_i^{(r)}(l)$, $r = 1, 2, \ldots$, based on the initial probabilities $p_i^{(0)}(l)$, the compatibility coefficients $\mathcal{R}$, and the weights C in the form

$$p_i^{(r+1)}(l) = \frac{p_i^{(r)}(l)\left[1 + q_i^{(r)}(l)\right]}{\sum_{l'=1}^{L} p_i^{(r)}(l')\left[1 + q_i^{(r)}(l')\right]}, \tag{17}$$

where

$$q_i^{(r)}(l) = \sum_{j=1}^{M} c_{ij}\left[\sum_{l'=1}^{L} r_{ij}(l, l')p_j^{(r)}(l')\right]. \tag{18}$$

Note that the compatibility coefficients $r_{ij}(l, l') \in [-1, 1]$ do not need to be symmetric. A value $r_{ij}(l, l')$ close to -1 indicates that label l' is unlikely at node v_j when label l occurs at node v_i whereas values close to 1 indicate the opposite. A value of exactly -1 indicates that the relation is not possible and a value of exactly 1 means that the relation always occurs.

Probabilistic relaxation provides a framework for smoothing but does not specify how the compatibility coefficients are computed. We use the coefficients proposed by Yamamoto [33]

$$r_{ij}(l, l') = \begin{cases} \frac{1}{1-p_i(l)}\left(1 - \frac{p_i(l)}{p_{ij}(l|l')}\right) & \text{if } p_i(l) < p_{ij}(l \mid l') \\ \frac{p_{ij}(l|l')}{p_i(l)} - 1 & \text{otherwise,} \end{cases} \tag{19}$$

where $p_{ij}(l \mid l')$ is the conditional probability that node v_i has label l given that node $v_j \in \mathcal{N}(v_i)$ has label l'. Each of the values $p_i(l)$ and $p_{ij}(l \mid l')$ are pre-calculated only once and remain the same during the iterations of the relaxation process. Thus, the coefficients $\mathcal{R}$ remain the same as well.

So far, we described the general method for relaxation labeling. It remains to describe how we apply this method for spatial smoothing of the classifications obtained by our AdaBoost classifier. To learn a topological map, we assume a given two-dimensional occupancy grid map in which each cell $m_{(x,y)}$ stores the probability that it is occupied. We furthermore consider the eight-connected graph induced by such a grid. Let $v_i = v_{(x,y)}$ be a node corresponding to a cell $m_{(x,y)}$ from the map. Then we define a neighborhood $N_8(v_{(x,y)})$ using the 8-connected cells to $v_{(x,y)}$ as described in [9].

For the initial probabilities $p_{(x,y)}^{(0)}(l)$, we use the output z of the classifier as described in the beginning of this section. Our set of labels is $\mathcal{L} = \{\text{corridor}, \text{room}, \text{doorway}, \text{wall}\}$. For each node $v_{(x,y)}$ in the free space of the occupancy grid map, we calculate the expected laser scan by ray-casting in the map. We then classify the observation and obtain a probability distribution z over all the possible places according to Eq. (16). The classification output z for each pose (x, y) is used to initialize the probability distribution $P_{(x,y)}^{(0)}$ of node $v_{(x,y)}$. For the nodes lying in the free space, the probability $p_{(x,y)}^{(0)}(wall)$ of being a wall is initialized with 0. Accordingly, the nodes corresponding to occupied cells in the map are initialized with $p_{(x,y)}^{(0)}(wall) = 1$.

Each of the weights $c_{ij} \in C$ is initialized with the value $\frac{1}{8}$, indicating that all the eight neighbors v_j of node v_i are equally important. The compatibility coefficients are calculated using Eq. (19). The values $p_i(l)$ and $p_{ij}(l \mid l')$ are obtained from statistics in the training data.

C. Region Extraction and Topological Mapping

We define a region λ_l on a adjacency graph $\mathcal{A}$ as a set of 8-connected nodes with the same label l. For each label $l \in \{\text{corridor}, \text{room}, \text{doorway}\}$, regions are extracted from the adjacency graph using the algorithm by Rosenfeld and Pfaltz [26]. Each region λ_l is assigned a different identifier. The connections between regions are extracted in a similar way [9]. Finally, a topological graph $\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ is constructed in which each node $v_i \in \mathcal{V}_\mathcal{T}$ represents a region and each edge $\in \mathcal{E}_\mathcal{T}$ represents a connection. Additionally, we add to each node v_i information about the properties of the region λ_l which are the area, the centroid, and the major and minor axis of the ellipse approximation of λ_l. The major and minor axis are vectors which represent the elongation of the region and its orientation. The graph form the final topological map together with the region properties attached to the nodes. We finally apply a heuristic region correction to the topological map to increase the classification rate:

1) We mark each region corresponding to a room or a corridor whose size does not exceed a given threshold of 1m^2 compared to the training set as classification error and assign the label of one of its connected regions.
2) We mark each region labeled as doorway whose size does not exceed a given threshold of 0.1m^2 square meters or that is connected to only one region as false classification and assign the label of one of its connected regions.

IV. Experiments

The approach described above has been implemented and tested using real robots and datasets gathered with real robots. We first present the results of our SLAM approach and then illustrate how to extract the topology of the environment.

A. Mapping with Rao-Blackwellized Particle Filters

Our SLAM approach has been implemented and runs online on standard laptop computers. The first set of experiments is designed to show the accuracy of our solution to the SLAM problem. Most of the maps generated by our approach can be magnified up to a resolution of $1\,cm$, without observing considerable inconsistencies. Even in big real world datasets covering an area of approximately $250\,m$ by $250\,m$, our approach never required more than 80 particles to build accurate maps. Highly accurate grid maps have been generated with our approach from several datasets. These maps, raw data files, and an efficient implementation of our mapping system are available on the web [29].

A map of the Intel Research Lab is depicted in the left image of Figure 3 and has a size of $28\,m$ by $28\,m$. The dataset has been recorded with a Pioneer II robot equipped with a

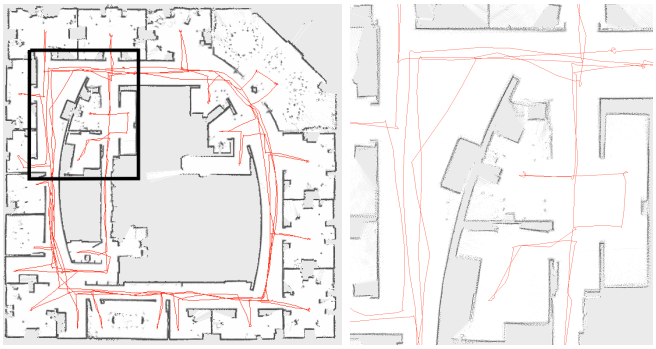


Fig. 3. The Intel Research Lab. The robot starts in the upper part of the circular corridor, and runs several times around the loop, before entering the rooms. The left image depicts the resulting map generated with 15 particles. The right image shows a cut-out with 1 *cm* grid resolution to illustrate the accuracy of the map in the loop closure point.

TABLE I

THE NUMBER OF PARTICLES NEEDED BY OUR ALGORITHM COMPARED TO THE APPROACH OF HÄHNEL *et al.* [11].

Proposal Distribution	Intel	MIT	Freiburg Campus
our approach	8	60	20
approach of [11]	40	400	400

SICK laser range finder. To successfully correct this dataset, our algorithm needed only 15 particles. As can be seen in the right image of Figure 3, the quality of the final map is so high that the map can be magnified up to 1 *cm* of resolution without showing any significant errors.

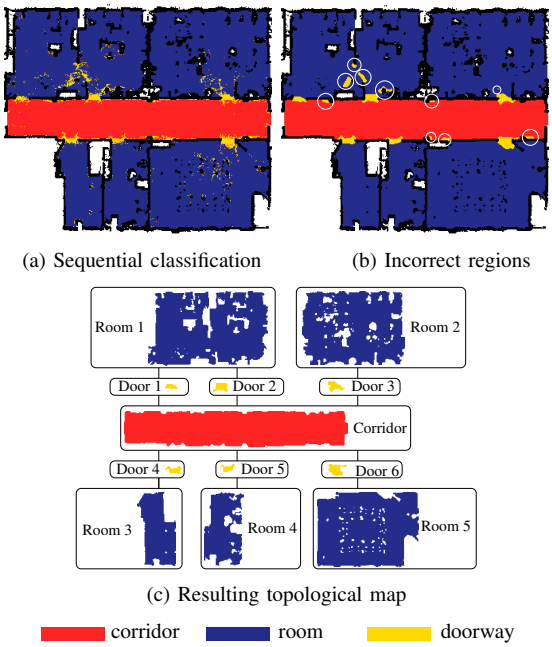
In order to measure the improvement in terms of the number of particles, we compared the performance of our system using the informed proposal distribution to the approach done by Hähnel *et al.* [11]. Table I summarizes the number of particles needed by a RBPF for providing a topologically correct map in at least 60% of all applications of our algorithm. In addition to the Intel dataset, we also corrected two other dataset which are not depicted for the sake of brevity. The corrected images of the MIT Killian Court dataset and the Freiburg Campus can be found on the web [29].

It turns out that in all of the cases, the number of particles required by our approach was approximately one order of magnitude smaller than the one required by the other approach. Moreover, the resulting maps are better due to our improved sampling process that takes the last reading into account. A more detailed discussion on the results of this Rao-Blackwellized approach to mapping can be found in [10].

B. Extraction of the Topology

The goal of this second set of experiments is to demonstrate that we can construct a topological maps based on grid map obtained with our SLAM approach. We also show that our method can be used to create a topological map of an environment for which no training data is available.

The environment depicted in Figure 4 is an office environment at the university of Freiburg. It contains rooms, doorways and a corridor. Our classifier correctly classifies 97.27% of the test examples. The classification is depicted as colors/grey levels in Figure 4(a). After the sequential classification, the



(a) Sequential classification (b) Incorrect regions

(c) Resulting topological map

Fig. 4. This figure shows a map of the building 79 at the University of Freiburg. (a) depicts the result of applying the sequential AdaBoost with a classification rate of 97.27%, (b) the result of applying relaxation and the detection of incorrect labeled regions (marked with circles), and (c) the final topological map with the corresponding regions.

probabilistic relaxation method explained in Section III-B is applied. This method generates more compact regions and eliminates noise. The result is illustrated in the Figure 4(b). Finally, the topological map is created using the connections between regions. As can be seen in Figure 4(b), some regions detected as doorways (marked with circles) do not correspond to real doorways. After applying the steps described in Section III-C on the corresponding topological map, these false doorways are eliminated. Furthermore, the two left rooms situated above the corridor are detected as only one region. That is due to the fact that the doorway in between was not completely detected. Thus, the two rooms remain connected and are classified as only one region. The final topological map, depicted in Figure 4(c), has a final classification rate of 98.95% of the data points.

C. Application to a New and Unknown Indoor Environment

In general, the result of the semantic classification depends on the data used to train the classifier. In case the data used for training the classifier is significantly different from the environment used for mapping, the quality of the classification decreases. This experiment analyzes whether our approach can be used to create a topological map of a new environment which is significantly different from the ones used for training. To carry out the experiment, we used the classifier from the environment shown in Figure 2 and 4. In order to obtain a classifier with a better generalization, we used it at different scales. The resulting classifier was then evaluated using the SDR dataset, recorded in an empty building in Virginia, USA. The process for obtaining the topological map is illustrated in Figure 5. As can be seen, some rooms are originally classified as parts of the corridor. The corridor is detected as only one

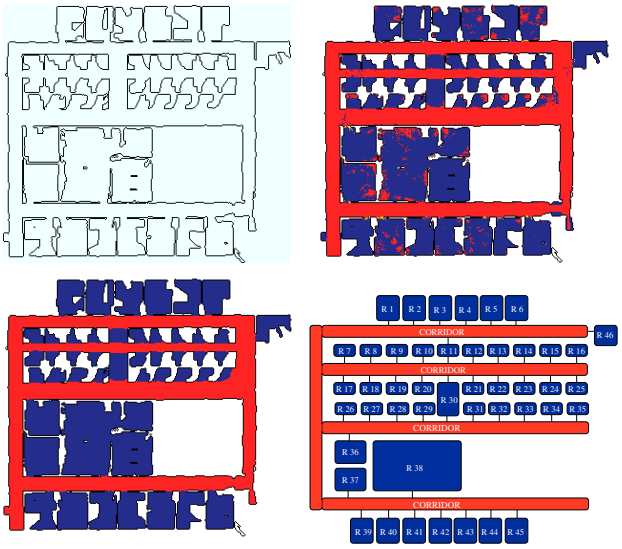


Fig. 5. This figure shows the original map of the building in the top left image and the results of applying the sequential AdaBoost classifier in the top right one. The lower left image depicts the resulting classification after the relaxation an region correction, and lower right one the final topological map with semantic information.

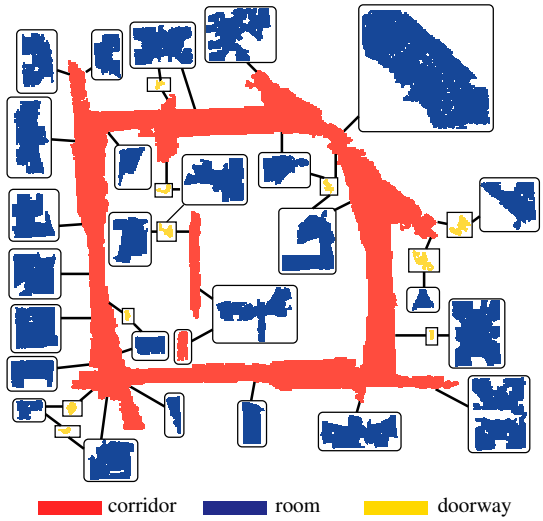


Fig. 6. The topological map learned from the Intel Research Lab.

region, although humans potentially would prefer to separate it into six different corridors: four horizontal and two vertical ones.

We also analyzed the results obtained without applying the relaxation process. This had several effects. First, omitting the relaxation procedure reduces the classification rate. Furthermore, the finally obtained regions typically are more sparse and do not represent the original ones as well as with relaxation. Finally, omitting the relaxation procedure increases the number of errors in the resulting topological map. For example, the map for the building in Virginia contained four incorrect nodes without relaxation, whereas there were only two incorrect nodes when we used the probabilistic relaxation.

Similar results can be obtained for the Intel dataset. Based on the results of the Rao-Blackwellized particle filter shown in Figure 3, we can construct the topological map. Note that this environment has significantly different structures like, for example, the round corridor and is therefore hard to classify. However, as depicted in Figure 6, the resulting topology represents the environment in a good way. The main error in the topology are missing doorways, since the doors in this environment look different to the environment in which the classifiers have been learned (Figure 2 and 4).

V. Related Work

Mapping techniques for mobile robots can be roughly classified according to the map representation and the underlying estimation technique. One popular map representation is the occupancy grid [23]. Whereas such grid-based approaches are computationally expensive and require a huge amount of memory, they are able to represent arbitrary objects. Feature-based representations are attractive because of their compactness. However, they rely on predefined feature extractors, which assumes that some structures in the environments are known in advance. Topological maps are typically the most compact model of the environment neglecting most of the metric information.

The work of Thrun [31] is closely related to our approach. He also learns a metric and a topological map of the environment. The metric map is build using an incremental scan-matching approach. This allows the robot to compensate for odometry errors and results in accurate maps learned from small-scale environments. Thrun furthermore constructs a Voronoi diagram to define interest points. In his approach, he selects voronoi points as topological nodes that describe a local minima in the obstacle clearance. In this way, he is able to separate, for example, two rooms that are connected by a doorway. In contrast to that, our approach first generates accurate maps from small- and large-scale environments by solving the simultaneous localization and mapping problem with a particle filter. Furthermore, we identify different areas according to their semantic information learned from previously seen environments. This allows us to distinguish, for example, between rooms and corridors and enables us to identify the boundaries of the area defining the topology.

In a work by Murphy, Doucet, and colleagues [5, 24], Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a possible robot trajectory and a map. The framework has been subsequently extended by Montemerlo et al. [21, 22] for approaching the SLAM problem with landmark maps. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [7] and Hähnel *et al.* [11]. Whereas the first work describes an efficient map representation, the second presents an improved motion model that reduces the number of required particles.

The SLAM technique described in Section II is an improvement of the algorithm proposed by Hähnel *et al.* [11]. Instead of using a fixed proposal distribution, our algorithm computes an improved proposal distribution on a per-particle basis on the fly. This allows us to directly use the information obtained from the sensors while evolving the particles. The computation of the proposal distribution is done in a similar way as in

FastSLAM-2 presented by Montemerlo *et al.* [20]. In contrast to FastSLAM-2, our approach does not rely on predefined landmarks and uses raw laser range finder data to acquire accurate grid maps. The advantage of our approach is twofold. Firstly, our algorithm draws the particles in a more effective way. Secondly, the highly accurate proposal distribution allows us to utilize the effective sample size as a robust indicator to decide whether or not a resampling has to be carried out. This further reduces the risk of particle depletion.

Bosse *et al.* [2] describe a generic framework for SLAM in large-scale environments. They use a graph structure of local maps with relative coordinate frames and always represent the uncertainty with respect to a local frame. In this way, they are able to reduce the complexity of the overall problem.

In the past, different algorithms for creating topological maps have been proposed. Kuipers and Byun [15] extract distinctive points in the map, which are defined as the local maximum of some measure of distinctiveness. Kortenkamp and Weymouth [13] fuse the information obtained with vision and ultrasound sensors to determine topologically relevant places. Shatkey and Kaelbling [28] apply a HMM learning approach to learn topological maps in which the nodes represent points in the plane. Additionally, Kuipers and Beeson [14] apply different learning algorithms to calculate topological maps of environments. These approaches only identify points in the map that have special properties but they do not include any means for extracting the types of places or even regions. In contrast to this, our approach presented in this paper is able to identify complete regions in the map like corridors, rooms or doorways, which have a direct relation with a human understanding of the environment.

In the context of learning topological map from noisy data, Modayil *et al.* [19] presented a technique which combines metrical SLAM with topological SLAM. The topology is utilized to solve the loop-closing problem, whereas metric information is used to build up local structures. Similar ideas have been realized by Lisien *et al.* [16], which introduce a hierarchical map in the context of SLAM.

Additionally, several authors considered the problem of identifying certain types of places. For example, Buschka and Saffiotti [3] describe a virtual sensor that is able to identify rooms from range data. Also Koenig and Simmons [12] use a pre-programmed routine to detect doorways from range data. Althaus and Christensen [1] use sonar data to detect corridors and doorways.

With respect to place classification, our approach is an extention of our previous work [18]. We additionally use a probabilistic variant of the classifier and apply a probabilistic relaxation labeling to incorporate similarity constraints between neighboring points and to eliminate false classifications.

The overall approach presented in this paper allows a mobile robot to learn a highly accurate metric occupancy grid map as well as a consistent topological model of the environment using noisy input data. The pose correction is done by applying a Rao-Blackwellized particle filter with an informed proposal distribution and adaptive resampling. The topological model is extracted based on the result of the filter and different learning algorithms. AdaBoost is used to estimate semantic labels of places and probabilistic relaxation labeling is applied smooth the results. Finally, the topology can be extracted and modeled in a graph structure.

VI. Conclusion

In this paper, we presented a method to learn accurate metric as well as topological maps under uncertainty. We described our algorithm that consists of two consecutive steps. First, it applies a Rao-Blackwellized particle filter to solve the SLAM problem and to create metric occupancy grid maps. We compute a highly accurate proposal distribution based on the observation likelihood of the most recent sensor information, the odometry, and a scan-matching process. This allows us to draw particles in a more accurate manner which seriously reduces the number of required samples and the quality of the resulting map. In the second step, we extract semantic place labels from the metric model for categorizing places into semantic classes such as rooms, doorways, and corridors. We apply a probabilistic relaxation process to reduce classification errors. We then extract regions and their connections which results in a topological representation of the environment. One advantage of this approach is that the nodes of the resulting graph correspond to the individual semantic regions. This links the metric and the topological representations. As a result, we obtained an accurate grid map modeling the metric information as well as a topological map representing the structure of the environment.

Our approach has been implemented and evaluated using real robots equipped with a laser range finder. Tests performed with our algorithm in different large-scale environments have demonstrated its robustness and the ability of generating high quality maps. The approach is well-suited to extract the topology from indoor environments even without training the classifier for each environment individually.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8, and by the EC under contract number FP6-004250-CoSy, FP6-IST-027140-BACS, and FP6-2005-IST-5-muFly. The authors would like to acknowledge Andrew Howard for providing us the SDR dataset and Dirk Hähnel for the Intel Research Lab and the Belgioioso dataset.

References

[1] P. Althaus and H.I. Christensen. Behaviour coordination in structured environments. *Advanced Robotics*, 17(7):657–674, 2003.

[2] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1899–1906, Taipei, Taiwan, 2003.

[3] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 637–642, Lausanne, Switzerland, 2002.

[4] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge, 1998.

[5] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 176–183, Stanford, CA, USA, 2000.

[6] A. Doucet, N. de Freitas, and N. Gordan, editors. *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[7] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[9] R.C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Inc., 1987.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[11] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, Las Vegas, NV, USA, 2003.

[12] S. Koenig and R. Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122. MIT Press, 1998.

[13] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proc. of the Twelfth National Conference on Artificial Intelligence*, pages 979–984, 1994.

[14] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, Edmonton, Canada, 2002.

[15] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics & Autonomous Systems*, 8:47–63, 1991.

[16] B. Lisien, D. Silver D. Morales, G. Kantor, I.M. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 448–453, Las Vegas, NV, USA, 2003.

[17] J.S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statist. Comput.*, 6:113–119, 1996.

[18] O. Martínez-Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1742–1747, Barcelona, Spain, 2005.

[19] J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1530–1536, Sendai, Japan, 2004.

[20] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

[21] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1985–1991, Taipei, Taiwan, 2003.

[22] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[23] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[24] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[25] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Systems. Man. Cybernet*, 6(6):420–433, 1976.

[26] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13(4):471–494, 1966.

[27] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn.*, 37(3):297–336, 1999.

[28] H. Shatkey and L.P. Kaelbling. Learning topological maps with weak local odometric information. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1997.

[29] C. Stachniss and G. Grisetti. Mapping results obtained with Rao-Blackwellized particle filters. http://www.informatik.uni-freiburg.de/~stachnis/research/rbpfmapper/, 2004.

[30] C. Stachniss, O. Martínez-Mozos, A. Rottmann, and W. Burgard. Semantic labeling of places. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, San Francisco, CA, USA, 2005.

[31] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[32] P. Viola and M.J. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Theories of Computer Vision*, Vancouver, Canada, 2001.

[33] H. Yamamoto. A method of deriving compatibility coefficents for relaxation operators. *Compt. Graph. Image Processing*, 10:256–271, 1979.

[J9] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and accurate slam with rao-blackwellized particle filters. *Robots and Autonomous Systems*, 55(1):30–38, 2007.

Fast and Accurate SLAM with Rao-Blackwellized Particle Filters

Giorgio Grisetti [a,b] Gian Diego Tipaldi [b] Cyrill Stachniss [c,a]
Wolfram Burgard [a] Daniele Nardi [b]

[a] *University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany*
[b] *Dipartimento Informatica e Sistemistica, Universitá "La Sapienza", I-00198 Rome, Italy*
[c] *Swiss Federal Institute of Technology (ETH) Zurich, CH-8092 Zurich, Switzerland*

Abstract

Rao-Blackwellized particle filters have become a popular tool to solve the simultaneous localization and mapping problem. This technique applies a particle filter in which each particle carries an individual map of the environment. Accordingly, a key issue is to reduce the number of particles and/or to make use of compact map representations. This paper presents an approximative but highly efficient approach to mapping with Rao-Blackwellized particle filters. Moreover, it provides a compact map model. A key advantage is that the individual particles can share large parts of the model of the environment. Furthermore, they are able to reuse an already computed proposal distribution. Both techniques substantially speed-up the overall filtering process and reduce the memory requirements. Experimental results obtained with mobile robots in large-scale indoor environments and based on published standard datasets illustrate the advantages of our methods over previous mapping approaches using Rao-Blackwellized particle filters.

Key words: SLAM, Rao-Blackwellized particle filter, grid map, informed proposal
PACS:

1 Introduction

Learning maps is a fundamental task of mobile robots and a lot of researchers focused on this problem. In the literature, the mobile robot mapping problem is often referred to as the *simultaneous localization and mapping (SLAM)* problem [1, 2, 3, 4, 5, 6, 7, 8]. In general, SLAM is a complex problem because for learning a map the robot requires a good pose estimate while at the same time a consistent map is needed to localize the robot. This dependency between the pose and the map estimate makes the SLAM problem hard and requires to search for a solution in a high-dimensional space.

Murphy, Doucet, and colleagues [7, 9] introduced Rao-Blackwellized particle filters (RBPFs) as an effective means to solve the SLAM problem. The main problem of Rao-Blackwellized particle filters lies in their complexity, measured in terms of the number of particles required to learn an accurate map. Reducing this quantity is one of the major challenges for this family of algorithms.

The contribution of this paper is a technique that reduces the computational and the memory requirements in the context of mapping with Rao-Blackwellized particle filters. In this way, it becomes feasible to maintain a comparably large set of particles online. This is achieved by enabling a subset of samples to share large parts of the map and to use the same proposal distribution. Our system allows a standard laptop computer to perform all computations necessary to learn accurate maps with more than one thousand samples online.

This paper is organized as follows. After the discussion of related work, we briefly introduce mapping with RBPFs. We then describe our technique for efficiently drawing particles from a proposal distribution. After this, we present our map representation and the concept of particle clusters. Finally, we show experiments illustrating the improvements of our approach to map learning with RBPFs.

2 Related Work

The estimation techniques for the SLAM problem can be classified according to their underlying basic principle. The most popular approaches are extended Kalman filters (EKFs), maximum likelihood techniques, sparse extended information filters (SEIFs), and Rao Blackwellized particle filters (RBPFs). The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior over landmark maps and robot poses [10, 11]. Their weakness lies in the strong assumptions that have to be made on the robot motion model and the sensor noise. Moreover, in the basic framework the landmarks are assumed to be uniquely identifiable. There exist techniques [12] to deal with unknown data association in the SLAM context, however, if certain assumptions are violated, the filter is likely to diverge [13].

Thrun *et al.* [8] proposed a SEIF method which is based on the inverse of the covariance matrix. In this way, measurements can be integrated efficiently. Eustice *et al.* [14] presented an improved technique to accurately compute the error-bounds within the SEIF framework and thus reduces the risk of becoming overly confident. Paskin [15] presented a solution to the SLAM problem using thin junction trees. This reduces the complexity compared to EKF-based approaches since thinned junction trees provide a linear-time filtering operation.

An alternative approach is to use a maximum likelihood algorithm that computes a map by constructing a network of relations. The relations represent the spatial

constraints between the poses of the robot [3, 16]. The main difference to RBPFs is that the maximum likelihood approach can only track a single mode of the distribution about the trajectory of the robot. It computes the solution by minimizing the least square error introduced by the constraints.

Lisien *et al.* [17] realized an hierarchical map model in the context of SLAM and reported that this improves loop-closing. Bosse *et al.* [18] describe a generic framework for SLAM in large-scale environments. They use a graph structure of local maps with relative coordinate frames similar to the work described in [19]. This approach is able to reduce the complexity of the overall problem and it better deals with the linearizations in the context of EKF-SLAM. Our approach is related to this framework since we also use local maps attached to a graph structure to model the environment. However, our motivation to use such a map representation is to allow multiple particles to share local maps and to compute the proposal distributions in an efficient way.

Murphy [7] introduced Rao-Blackwellized particle filters as an effective means to solve the SLAM problem. Each particle in a RBPF represents a potential trajectory of the robot and a map of the environment. The framework has been subsequently extended by Montemerlo *et al.* [5, 6] for approaching the SLAM problem with landmarks. To learn accurate grid maps, Hähnel *et al.* [4] presented an improved motion model that reduces the number of required particles. A combination of the approach of Hähnel *et al.* and Montemerlo *et al.* as been presented by Grisetti *et al.* [2], which extends the ideas of FastSLAM-2 [5] to the grid map case. We present in this paper an approximative solution to RBPF-based mapping which describes how to draw particles and how to represent the maps of the particles so that the system can be executed significantly faster and needs less memory resources.

There exist other approaches to mapping with RBPFs like DP-SLAM [1] that provide a compact map representation. This approach stores an ancestry tree of particles. Furthermore, each cell of their grid map maintains a tree of poses from which that cell has been observed. This allows the system to store the map hypotheses in an compact manner. Additionally, the resampling can be carried out more efficiently. In contrast to that, our map representation enables us to reuse already computed proposal distributions for multiple samples. This is done by carrying out a coordinate transformation between the reference frames stored in our graph structure.

The contribution of this paper is a computational and memory efficient Rao-Blackwellized particle filter for SLAM. Our approach allows the robot to efficiently determine the proposal distributions to sample the next generation of particles in an approximative manner. Additionally, we present a compact map model in which multiple particles share local maps. This enables us to maintain substantially more samples with less memory and computational requirements compared to state-of-the-art mapping approach using Rao-Blackwellized particle filters.

3 Learning Maps with Rao-Blackwellized Particle Filters

The key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot and the map m of the environment given the observations $z_{1:t} = z_1, \ldots, z_t$ and odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$. It does so by using the following factorization:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) p(m \mid x_{1:t}, z_{1:t}) \tag{1}$$

In this equation, the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ is similar to the localization problem, since only the trajectory of the vehicle needs to be estimated. This estimation is performed using a particle filter which incrementally processes the observations and the odometry readings as they are available. The second term $p(m \mid x_{1:t}, z_{1:t})$ can be computed efficiently since the poses $x_{1:t}$ of the robot are known when estimating the map m. Therefore, a Rao-Blackwellized particle filter for SLAM maintains an individual map for each sample and updates this map based on the trajectory estimate of the sample upon "mapping with known poses".

A mapping system that applies a RBPF requires a proposal distribution in order to draw the next generation of samples. The general framework leaves open which proposal should be used and how it should be computed. A proposal distribution typically used in the context of Monte-Carlo localization is the motion model $p(x_t \mid x_{t-1}, u_{t-1})$. This proposal, however, is sub-optimal since it does not consider the observations of the robot to predict its motion. As pointed out by several authors [20, 5], problem-specific proposal distributions are needed in order to build an efficient mapping system. The approach presented in this paper, makes use of our previously defined [2] proposal distribution. It transfers the ideas of FastSLAM-2 [5] to the grid map case. Under the Markov assumption, the optimal proposal distribution [20] is

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{\int p(z_t \mid m_{t-1}^{(i)}, x') p(x' \mid x_{t-1}^{(i)}, u_{t-1}) \, dx'}. \tag{2}$$

Whenever a laser range finder is used, one can observe that the observation likelihood $p(z_t \mid m_{t-1}, x_t)$ is much more peaked than the motion model $p(x_t \mid x_{t-1}, u_{t-1})$. The observation likelihood dominates the product in Eq. (2) in the meaningful area of the distribution. Therefore, we approximate $p(x_t \mid x_{t-1}, u_{t-1})$ by a constant k within this meaningful area $L^{(i)}$. Under this approximation, the proposal turns into

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \simeq p(z_t \mid m_{t-1}^{(i)}, x_t) \cdot \left[\int_{x' \in L^{(i)}} p(z_t \mid m_{t-1}^{(i)}, x') \, dx' \right]^{-1}. \tag{3}$$

Eq. (3) can be computed by evaluating $p(z_t \mid m_{t-1}^{(i)}, x_t)$ on a grid which is bounded by the maximum odometry error. Alternatively, one can use a set of sampled points

$\{x_j\}$ and then evaluate point-wise the observation likelihood. In order to efficiently sample the next generation of particles, one can approximate this distribution by a Gaussian. For each particle i, the parameters $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ of the Gaussian are computed as

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{K} x_j \cdot p(z_t \mid m_{t-1}^{(i)}, x_j) \tag{4}$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T. \tag{5}$$

Here $\eta = \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j)$ is a normalizer. Note that $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ are calculated for each particle individually which is computationally expensive but leads to an informed proposal distribution. This allows us to draw particles in an more accurate manner which seriously reduces the number of required samples.

4 Speeding Up the Computation of the Proposal

The problem of the method presented above is the computational complexity of the informed proposal distribution since it has to be done for each sample individually. As a result, such a mapping system runs online only for small particle sets. Furthermore, each particle maintains a full grid map which requires to store large grid structures in the memory. To overcome this limitation, we present a way to utilize intermediate results in order to efficiently determine the proposal for the individual samples. Our implementation extends the open-source implementation [21] of the mapping system of Grisetti *et al.* [2] which originally makes use of the proposal distribution presented in the previous section.

The proposal distribution is needed to model the relative movement of the vehicle under uncertainty. In most situations, this uncertainty is similar for all samples within one movement. It therefore makes sense to use the same uncertainty to propagate the particles. We derive a way to sample multiple particles from the same proposal. As a result, the time consuming computation of the proposal distribution can be carried out for a few particles that are representatives for groups of similar samples.

Furthermore, we observed that local maps which are represented in a particle-centered coordinate frame look similar for many samples. We therefore present a compact map model in which multiple particles can share their local maps. Instead of storing a full grid map, each sample maintains only a set of reference frames for the different local maps. This substantially reduces the memory requirements of the mapping algorithm.

4.1 Different Situations During Mapping

Before we derive our new proposal distributions, we start with a brief analysis of the behavior of a RBPF. One can distinguish three different types of situations during mapping:

- The robot is moving through *unknown* areas,
- is moving through *known* areas, or
- is *closing a loop*. Here, closing a loop means that the robot first moves through unknown areas and then reenters known terrain. It can be seen as moving along a so far non traversed shortcut from current pose of the robot to an already known area (see also [22]).

In each of those situations, the filter behaves differently. Whenever the robot is moving through unknown terrain, the uncertainty about the pose of the robot grows. This is due to the fact that the errors are accumulated along the trajectory. The resulting uncertainty can only be bounded by observations which cover a (partially) known region.

In the second case, a map of the surroundings of the robot is known and in this way the SLAM problem turns into a localization problem which is typically easier to handle. Whenever the robot is closing a loop, the particle cloud is often widely spread. By reentering known areas, the filter can typically determine which particles are consistent with their own map and which are not. As a result, such a situation leads to an unbalanced distribution of particle weights. The next resampling action then eliminates a series of unlikely hypotheses and the uncertainty decreases.

For each of these three situations, we will present a proposal distribution that needs to be computed only for a small set of representatives rather than for all particles. Throughout this paper, we make the following three assumptions.

Assumption 1 The current situation is known, which means that the robot can determine whether it is moving through unknown terrain, within a known area, or is closing a loop.

Assumption 2 The corresponding local maps of two samples are similar if considered in a particle-centered reference frame. In the following, we refer to this property as *local similarity* of the maps.

Assumption 3 An accurate algorithm for pose tracking is used and the observations are affected by a limited sensor noise.

4.2 Computing the Proposal for Unknown Terrain

For proximity sensors like laser range finders, the observations of the robot cover only a local area around the robot. As a result, we only need to consider the sur-

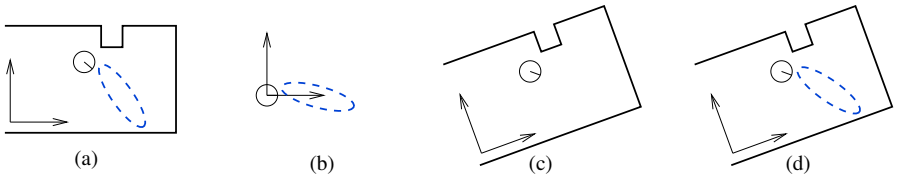


Figure 1. Image (a) depicts the pose of a particle, its local map, and the computed proposal which represented by the blue/dashed ellipse. Image (b) illustrates the proposal distribution represented in the ego-centric reference frame of that sample. Image (c) shows a second particle and its map. By carrying out a coordinate transform, the proposal of the first particle can be used by the second particle as long as their maps are (locally) similar (d).

roundings of the robot when computing the proposal distribution. Let $\tilde{m}_{t-1}^{(i)}$ refer to the local map of particle i around its previous pose $x_{t-1}^{(i)}$. In the surroundings of the robot, we can approximate

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \;\simeq\; p(x_t \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \tag{6}$$

Let $\oplus$ and $\ominus$ be the standard pose compounding operators (see [16]): $a \ominus b$ is an operator that translates all the points in the domain of the function a so that the new origin of the domain of a is b and $\oplus$ is its inverse. The local similarity between maps (Assumption 2) allows us to write $\tilde{m}_{t-1}^{(i)} \ominus x_{t-1}^{(i)} \simeq \tilde{m}_{t-1}^{(j)} \ominus x_{t-1}^{(j)}$. In this case, the proposal distribution for different particles are similar if transformed to an ego-centric reference frame

$$p(x_t \ominus x_{t-1}^{(j)} \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(x_t \ominus x_{t-1}^{(i)} \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \tag{7}$$

As a result, we can determine the proposal for a particle j by computing the proposal in the reference frame of particle i and then translating it to the reference frame of particle j

$$p(x_t \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(x_{t-1}^{(j)} \oplus (x_t \ominus x_{t-1}^{(i)}) \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \tag{8}$$

This computation is illustrated in Figure 1. It shows how to transform a proposal between particles. The complex proposal computation needs to be performed only once and can then be translated to the reference frame of the other particles.

4.3 Computing the Proposal for Already Visited Areas

Whenever the robot moves through known areas, each particle stays localized in its own map according to Assumption 3. To update the new pose of each particle while the robot moves, we choose the pose x_t that maximizes the likelihood of the observation around the pose predicted by odometry

$$x_t^{(i)} = \underset{x_t}{\operatorname{argmax}}\, p(x_t \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \tag{9}$$

Analog to Eq. (6)-(8), we can express the proposal of particle j using the one of particle i. The only difference is that we do not apply the $\oplus$ and $\ominus$ operators based on the poses of the samples. Instead, the operators are applied based on the particle dependent reference frames $l^{(i)}$ and $l^{(j)}$ of the local maps. These reference frames were established whenever the robot visits the corresponding area for the first time. This results in

$$p(x_t \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(l^{(j)} \oplus (x_t \ominus l^{(i)}) \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \quad (10)$$

Combining Eq. (9) and Eq. (10) leads to

$$x_t^{(j)} = \operatorname*{argmax}_{x_t} p(x_t \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \quad (11)$$

$$\simeq \operatorname*{argmax}_{x_t} p(l^{(j)} \oplus (x_t \ominus l^{(i)}) \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = l^{(j)} \oplus (x_t^{(i)} \ominus l^{(i)}). \quad (12)$$

Under the Assumptions 2 and 3, we can estimate the poses of all samples according to Eq. (12) (while moving through known areas). In this way, the complex computation of an informed proposal needs to be done only once.

4.4 Computing the Proposal When Closing a Loop

In contrast to the two situations described before, the computation of the proposal is more complex in case of a loop-closure. This is due to the fact that Assumption 2 (local similarity) is typically violated even for subsets of particles. Let us assume that the particle cloud is widely spread when the loop is closed. Typically, the individual samples reenter the previously mapped terrain at different locations. This results in different hypotheses about the topology of the environment and definitively violates Assumption 2. Dealing with such a situation, requires additional effort in the estimation process.

Whenever a particle i closes a loop, we consider that the map $\tilde{m}_{t-1}^{(i)}$ of its surroundings consists of two components. Let $m_{\text{loop}}^{(i)}$ refer to the map of the area, the robot seeks to reenter. Then, $m_{\text{local}}^{(i)}$ is the map constructed from the most recent measurements without the part of the map that overlaps with $m_{\text{loop}}^{(i)}$. Since those two maps are disjoint and under the assumption that the individual grid cells are independent, we can use a factorized form for our likelihood function

$$p(z_t \mid x_t, m_{\text{local}}^{(i)}, m_{\text{loop}}^{(i)}) \propto p(z_t \mid x_t, m_{\text{local}}^{(i)}) \cdot p(z_t \mid x_t, m_{\text{loop}}^{(i)}). \quad (13)$$

For efficiency reasons, we use only the local map $m_{\text{local}}^{(i)}$ to compute the proposal and do not consider $m_{\text{loop}}^{(i)}$. This procedure is valid but requires to adapt the weight computation. According to the importance sampling principle, this leads to

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot \frac{p(x_t^{(i)} \mid z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, m_{\text{loop}}^{(i)}, u_{t-1})}{p(x_t^{(i)} \mid z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, u_{t-1})} \tag{14}$$

$$= w_{t-1}^{(i)} \cdot \frac{\eta_1^{(i)} p(z_t \mid x_t^{(i)}, m_{\text{local}}^{(i)}) p(z_t \mid x_t^{(i)}, m_{\text{loop}}^{(i)})}{\eta_2^{(i)} p(z_t \mid x_t^{(i)}, m_{\text{local}}^{(i)})} \tag{15}$$

$$= w_{t-1}^{(i)} \cdot p(z_t \mid x_t^{(i)}, m_{\text{loop}}^{(i)}) \frac{\eta_1^{(i)}}{\eta_2^{(i)}}, \tag{16}$$

where η_1 and η_2 are normalization factors resulting from Bayes' rule.

Note that the computation of the proposal in case of a loop-closure is more expensive than in the two other situations. Fortunately, loop-closing situations occur rarely. The robot has to travel through unknown and eventually known terrain for a comparably long period of time before a loop-closure can occur.

4.5 Approximative Importance Weight Computation

We observed in practical experiments that the normalizing factors η_1 and η_2 in Eq. (16) have only a minor influence on the overall weight. In order to speed up the computation of the importance weights, we approximate Eq. (16) by

$$w_t^{(i)} \simeq w_{t-1}^{(i)} \cdot p(z_t | x^{(i)}, m_{\text{loop}}^{(i)}) \tag{17}$$

in which the normalizing factors are neglected. This is significantly faster to compute and as we will demonstrate in the experiments leads to almost identical importance weights.

5 Achieving Situation Estimation, Local Similarity, and Pose Tracking

All of the derivations made in the previous section require that the robot knows whether it is moving through unknown terrain, through a previously mapped area, or is currently closing a loop (Assumption 1). Here, we describe how to distinguish the different cases. Detecting the first two situations can be done in a straightforward way by comparing the area covered by the current observation given the particle pose and the map constructed so far.

In general, it is more difficult to decide whether or not the robot is closing a loop. To detect loop closures, we apply the approach proposed by Stachniss *et al.* [22]. We use a dual representation consisting of a topologic map that models the trajectory of the vehicle and a grid map. By comparing both representations, one is able to accurately determine whether or not a robot is closing a loop.

Assumption 2 (local similarity) typically holds only up to the first loop closure but is then violated. By explicitly modeling the different topological hypotheses, it is

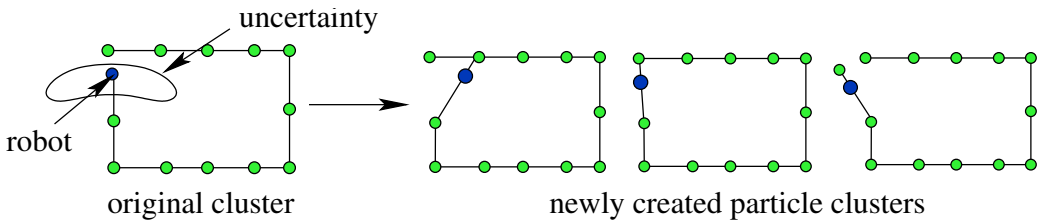


Figure 2. The left image depicts a cluster while the robot is approaching a loop-closure. The shown particle cluster splits up into three different clusters (topology hypotheses) as depicted in the right image.

still possible to represent the posterior in an appropriate way. To achieve local similarity, we introduce the notion of a *particle cluster* which describes a subset of particles for which the assumption of local similarity between maps holds. Ambiguities in the posterior can then be modeled using multiple particle clusters.

In the beginning of the mapping process, we start with a single cluster, but after closing a loop, multiple topology hypotheses typically occur. In this situation, the cluster needs to be split up. Therefore, we determine which particle belongs to which topological hypothesis in order to form new clusters. In our current implementation, we group the samples according to their Euclidian distance to the different nodes in their own graph structure of reference frames. For each particle, we first determine the list of nodes in the field of view of that sample. We order this list according to the Euclidian distance from the pose represented by the sample to the corresponding node. Then, a cluster is given by the samples which have the same list of nodes. An example which illustrates how new clusters arise in case of a loop-closure is depicted in Figure 2. Note that we currently do not merge clusters. Throughout our experiments, we observed that multiple particle clusters are created when closing a loop. The actual number ranges up to 50. However, after a few iterations only a small number of clusters (typically up to five) survive.

In our current implementation, we represent a map as a set of local maps also called patches. A global map for a given particle can be obtained by specifying the location of each patch within a global reference frame. Each sample therefore has to store only a list of reference frames $l_n^{(i)}$ for the patches. In this way, the individual patches $\mathcal{P}_1, \ldots, \mathcal{P}_N$ need to be stored only once per cluster. The map of particle i can be computed by $m^{(i)} = \bigcup_n l_n^{(i)} \oplus \mathcal{P}_n$.

Within one particle cluster, the local maps of each particle fulfills the assumption of local similarity. Therefore, they can share their patches. This results in a more compact representation compared to storing individual grid maps. In our current mapping system, we used a graph structure where each node is a reference to the corresponding patch. Furthermore, the state vector $s_t^{(i)}$ and the clusters $\mathcal{C}_k$ are rep-

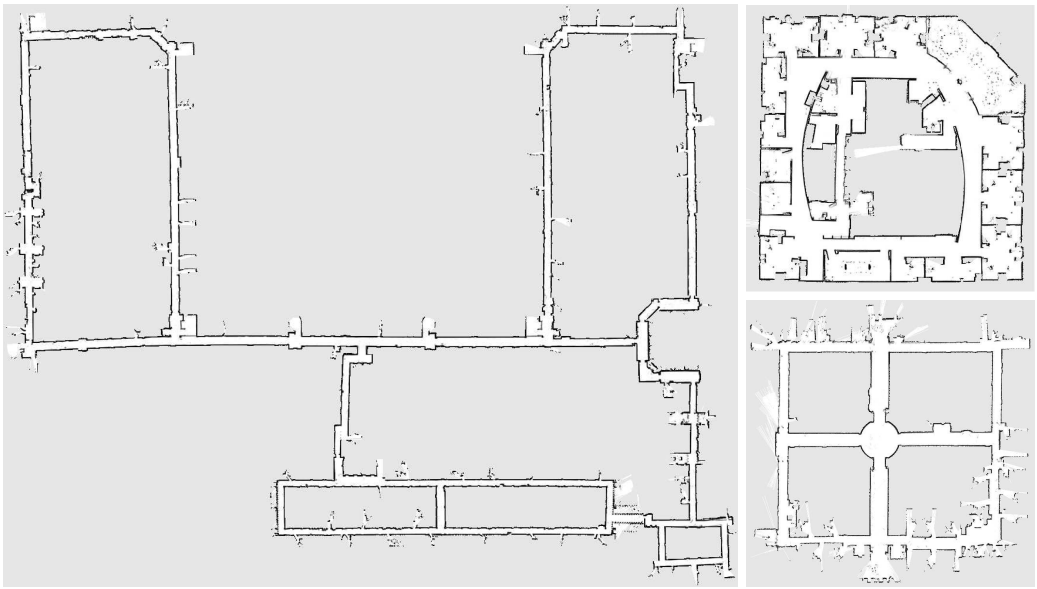


Figure 3. Learned map of the MIT Killian Court, the Intel Research lab, and the ACES dataset using our approach.

resented as

$$s_t^{(i)} = \left\langle \underbrace{x_t^{(i)}}_{\text{robot pose}} \ , \ \underbrace{k}_{\text{cluster ID}} \ , \underbrace{l_1^{(i)}, \ldots, l_{N_k}^{(i)}}_{\text{patch locations}} \right\rangle \quad \mathcal{C}_k = \left\langle \underbrace{\mathcal{P}_1, \ldots, \mathcal{P}_{N_k}}_{\text{pointers to patches}} \ , \ \underbrace{\{e_{l,m}\}}_{\text{graph edges}} \right\rangle . \tag{18}$$

Note that the number N_k of patches does not grow with the length of the trajectory traveled by the robot. It grows with the number of relevant patches which is related to the size of the environment.

To fulfill Assumption 3, we apply an incremental scan alignment technique based on laser range finder data. The experiments presented in this paper indicate that this setup/implementation is sufficient to satisfy the three assumptions. As a result, we obtain a mapping system which provides highly accurate maps in a fast and memory efficient manner.

6 Experiments

In this section, we present experiments performed on real robot datasets which are commonly used within the robotics community. In the first experiment, we corrected several log files using our approach. The left image of Figure 3 depicts the resulting map of the MIT Killian Court. This is a challenging dataset, since the environment is large (it took 2.5h to record this log file) and contains several nested loops which can rise the problem of particle depletion. As shown in the figure, the map does not contain any inconsistencies like for example double walls. Comparable results have been obtained using the Intel Research, the Austin ACES dataset, shown in the same figure.

Table 1
Comparison of memory and computational resources for the MIT dataset using a PC with a 1.3 GHz CPU.

	#particles	**execution time**	**max. memory**
our approach	2,000	51 min	210 MB
our approach	1,000	41 min	180 MB
our approach	500	30 min	165 MB
RBPF of [21]	150	(memory swapping)	2.9 GB
RBPF of [21]	80	300 min	1.5 GB
RBPF of [21]	50	190 min	1 GB

The second experiment is designed to show the advantages of our approach compared to a RBPF-based mapper without our optimizations. For this comparison, we used the open-source mapper provided in [21]. We compared the overall time, needed to correct the MIT Killian Court dataset and the memory used to store the maps. This was done using a (comparably slow) PC with a 1.3 GHz CPU and 1.5 GB RAM. The results of both mapping approaches are summarized in Table 1. In our current implementation, the filter update for *each cluster* takes in average 20 ms when moving through known terrain and 200 ms when moving through unknown terrain. When actually closing a loop, *each particle* requires approximatively 2 ms of execution time while the check for the closure takes around 0.3 ms per sample.

Since the approximated proposal is not as accurate as the original one, we need more particles to achieve the same robustness in filter convergence and quality of the resulting maps. However, we can maintain more than one order of magnitude more particles while requiring less runtime and memory. In all our experiments, this sufficiently accounted for the less accurately drawn samples.

The savings on runtime are mainly caused by transforming an already computed proposal distribution so that it can be used for several particles instead of computing it from scratch each time. The memory savings are due to the fact that all particles within a cluster can share a their local grid maps. Furthermore, the memory usage and runtime of our approach grows comparably slow when increasing the number of particles. The reason is that the complexity of our filter grows mainly with the number of topological hypotheses (particle clusters) which need to be maintained and only indirectly with the number of samples. Note that the *maximum* memory required by our approach is considerably higher than the amount of memory typically used. There exist a few peaks in the memory usage which arise from a loop closure where several clusters are temporarily created but deleted after a few steps (compare Figure 4). The typical memory usage is around 20% of the maximum usage.

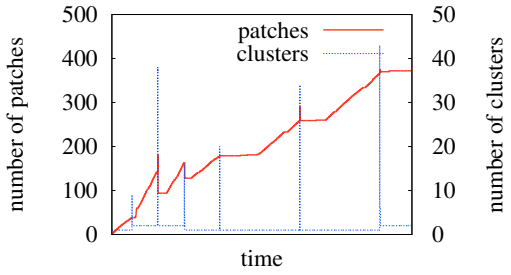


Figure 4. This plot depicts the number of patches in the memory and the number of clusters over time for the MIT dataset using 1.500 particles.

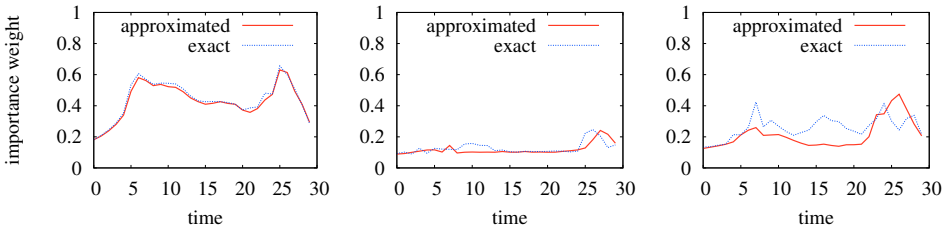


Figure 5. Difference in the particle weights caused the approximative computation for three different samples during a loop-closure. The left and middle image show typical results, the right one depicts the one of the worst results during our experiments.

Figure 4 depicts the number of patches that need to be stored and the number of clusters during the estimation process of the MIT dataset with 1,500 particles. As can be seen, the number of clusters is typically small until a loop closure occurs. At this point, the number of clusters increases. However, after a short period of time most of the clusters vanish.

The last experiment evaluates the error introduced by our approximative importance weight computation when closing a loop. As presented in Eq. (17), we ignore the normalization factors to achieve a faster estimation. We analyzed the loop-closing actions and in most situations the approximation error was small. Figure 5 depicts the differences between the sound computation and our approximation for three different particles during a loop closure. For a more quantitative evaluation between both methods, we computed the KL-divergence (KLD) between the distribution of the importance weights in both cases. It turned out, that the average KLD was only 0.02 (a KLD of 0 means that the distributions are equal and the higher the value the more different are the distributions). Substantiated by the good approximation quality, we ignore the evaluation of η_1 and η_2 when computing the particle importance weight.

7 Conclusion

In this paper, we presented efficient optimizations for Rao-Blackwellized particle filters applied to solve the SLAM problem on grid maps. We are able to update the complex posterior requiring substantially less computational and memory re-

sources. This is achieved by performing the computations only for a set of representatives instead of for all particles. We extended a state-of-the-art mapping system in a way that the computation of the proposal distribution is significantly faster and needs only a fraction of the memory resources. The key idea is that clusters of particles can share large parts of their map model as well as an informed proposal distribution used to draw the next generation of particles.

With our optimizations, we are able to maintain more than one order of magnitude more samples and at the same time require less memory and computational resources compared to other state-of-the-art mapping techniques using Rao-Blackwellized particle filters. With this comparably high number of particles that we are able to maintain, we can compensate for the errors introduced by our approximations. Our approach has been implemented, tested, and evaluated based on real robots and standard log files used within the SLAM community to demonstrate the accuracy and benefits of our system.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A3), and by the EC under contract number FP6-004250-CoSy and FP6-IST-027140-BACS. The authors would like to acknowledge Mike Bosse and John Leonard for providing us the dataset of the MIT Killian Court, Patrick Beeson for the ACES dataset, and Dirk Hähnel for the Intel Research Lab.

References

[1] A. Eliazar, R. Parr, DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks, in: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, 2003, pp. 1135–1142.

[2] G. Grisetti, C. Stachniss, W. Burgard, Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain, 2005, pp. 2443–2448.

[3] J.-S. Gutmann, K. Konolige, Incremental mapping of large cyclic environments, in: Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA), Monterey, CA, USA, 1999, pp. 318–325.

[4] D. Hähnel, W. Burgard, D. Fox, S. Thrun, An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2003, pp. 206–211.

[5] M. Montemerlo, S. T. D. Koller, B. Wegbreit, FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, in: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, 2003, pp. 1151–1156.

[6] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM: A factored solution to simultaneous localization and mapping, in: Proc. of the National Conference on Artificial Intelligence (AAAI), Edmonton, Canada, 2002, pp. 593–598.

[7] K. Murphy, Bayesian map learning in dynamic environments, in: Proc. of the Conf. on Neural Information Processing Systems (NIPS), Denver, CO, USA, 1999, pp. 1015–1021.

[8] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, H. Durrant-Whyte, Simultaneous localization and mapping with sparse extended information filters, Int. Journal of Robotics Research 23 (7/8).

[9] A. Doucet, J. de Freitas, K. Murphy, S. Russel, Rao-Blackwellized partcile filtering for dynamic bayesian networks, in: Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI), Stanford, CA, USA, 2000, pp. 176–183.

[10] J. Leonard, H. Durrant-Whyte, Mobile robot localization by tracking geometric beacons, IEEE Transactions on Robotics and Automation 7 (4) (1991) 376–382.

[11] R. Smith, M. Self, P. Cheeseman, Estimating uncertain spatial realtionships in robotics, in: I. Cox, G. Wilfong (Eds.), Autonomous Robot Vehicles, Springer Verlag, 1990, pp. 167–193.

[12] J. Neira, J. Tardós, Data association in stochastic mapping using the joint compatibility test, IEEE Transactions on Robotics and Automation 17 (6) (2001) 890–897.

[13] U. Frese, G. Hirzinger, Simultaneous localization and mapping - a discussion, in: Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle, WA, USA, 2001, pp. 17–26.

[14] R. Eustice, M. Walter, J. Leonard, Sparse extended information filters: Insights into sparsification, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Cananda, 2005, pp. 641–648.

[15] M. Paskin, Thin junction tree filters for simultaneous localization and mapping, in: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, 2003, pp. 1157–1164.

[16] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, Journal of Autonomous Robots 4 (1997) 333–349.

[17] B. Lisien, D. S. D. Morales, G. Kantor, I. Rekleitis, H. Choset, Hierarchical simultaneous localization and mapping, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2003, pp. 448–453.

[18] M. Bosse, P. Newman, J. Leonard, S. Teller, An ALTAS framework for scalable mapping, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan, 2003, pp. 1899–1906.

[19] C. Estrada, J. Neira, J. Tardós, Hierachical slam: Real-time accurate mapping of large environments, IEEE Transactions on Robotics 21 (4) (2005) 588–596.

[20] A. Doucet, On sequential simulation-based methods for bayesian filtering, Tech. rep., Signal Processing Group, Dept. of Engeneering, University of Cambridge (1998).

[21] C. Stachniss, G. Grisetti, Mapping results obtained with Rao-Blackwellized particle filters, http://www.informatik.uni-freiburg.de/∼stachnis/research/rbpfmapper/ (2004).

[22] C. Stachniss, D. Hähnel, W. Burgard, G. Grisetti, On actively closing loops in grid-based fastslam, Advanced Robotics 19 (10) (2005) 1059–1080.

[C1] J. Sturm, V. Predeap, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard. Learning kinematic models for articulated objects. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Pasadena, CA, USA, 2009.

Learning Kinematic Models for Articulated Objects

Jürgen Sturm[1] **Vijay Pradeep**[2] **Cyrill Stachniss**[1]
Christian Plagemann[3] **Kurt Konolige**[2] **Wolfram Burgard**[1]

[1] *Univ. of Freiburg, Dept. of Computer Science, D-79110 Freiburg, Germany*
[2] *Willow Garage, Inc., 68 Willow Road, Menlo Park, CA 94025*
[3] *Stanford University, CS Dept., 353 Serra Mall, Stanford, CA 94305-9010* *

Abstract

Robots operating in home environments must be able to interact with articulated objects such as doors or drawers. Ideally, robots are able to autonomously infer articulation models by observation. In this paper, we present an approach to learn kinematic models by inferring the connectivity of rigid parts and the articulation models for the corresponding links. Our method uses a mixture of parameterized and parameter-free (Gaussian process) representations and finds low-dimensional manifolds that provide the best explanation of the given observations. Our approach has been implemented and evaluated using real data obtained in various realistic home environment settings.

1 Introduction

Home environments are envisioned as one of the key application areas for service robots. Robots operating in such environments are typically faced with a variety objects they have to deal with or to manipulate to fulfill a given task. Furthermore, many objects are not rigid since they have moving parts such as drawers or doors. Understanding the spatial movements of parts of such objects is essential for service robots to allow them to plan relevant actions such as door-opening trajectories. In this paper, we investigate the problem of learning kinematic models of articulated objects from observations. As an illustrating example, consider Fig. 1 which depicts two examples for observations of the door of a microwave oven and a learned, one-dimensional description of the door motion.

Our problem can be formulated as follows: Given a sequence of locations from observed objects parts, learn a compact kinematic model describing the *whole* articulated object. This kinematic model has to define (i) the connectivity between the parts, (ii) the dimensionality of the latent (not observed) actuation space of the object, and (iii) a kinematic function between different body parts in a generative way allowing a robot to reason also about unseen configurations.

*This work has partly been supported by the DFG under contract number SFB/TR-8 and the EU under FP6-IST-045388 (INDIGO).

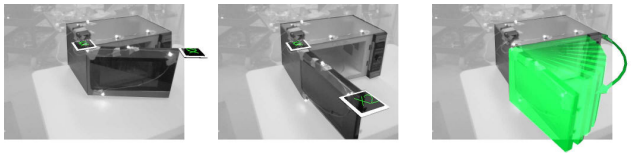


Figure 1: Left and middle: examples for observations of a moving door of a microwave oven. Right: visualization of the kinematic model of the door learned by our approach.

The contribution of this paper is a novel approach for learning such models based only on observations. Our method is able to robustly detect the connectivity of the rigid parts of the object and to estimate accurate articulation models from a candidate set. It allows for selecting the best model among parametric, expert-designed transformation templates (rotational and prismatic models), and non-parametric transformations that are learned from scratch requiring minimal prior assumptions. To obtain a parameter-free description, we apply Gaussian processes (GPs) [Rasmussen and Williams, 2006] as a non-parametric regression technique to learn flexible and accurate models. To find the low-dimensional description of the moving parts, we furthermore apply a local linear embedding (LLE) [Roweis and Saul, 2000], which is a non-linear dimensionality reduction technique. As the experiments described in this paper demonstrate, our technique allows to learn accurate models for different articulated objects from real data. We regard this as an important step towards autonomous robots understanding and actively handling objects in their environment.

Throughout this paper, we consider objects that are a collection of rigid bodies denoted as "object parts" in the 3D space and that they are *articulated*, which means that the configuration of their parts can be described by a finite set of parameters. The only required input are potentially noisy observations of the poses of object parts.

2 Related Work

Learning the kinematics of robots that can actively move their own body parts has been intensively investigated in the past: Dearden and Demiris [2005] learn a Bayesian network for a 1-DOF robot. Sturm *et al.* [2008a; 2008b] proposed an approach to infer probabilistic kinematic models by learning the conditional density functions of the individual joints and by

subsequently selecting the most likely topology. Their approach requires knowledge about the actions carried out by the robot or by the observed object—information which is not available when learning the models of arbitrary, articulated objects from observations only. Similarly, Taycher *et al.* [2002] address the task of estimating the underlying topology of an observed articulated body. Their focus lies on recovering the topology of the object rather than on learning a generative model with explicit action variables. Also, compared to their work, our approach can handle higher-dimensional transformations between object parts. Kirk *et al.* [2004] extract human skeletal topologies using 3D markers from a motion capture system. However, they assume that all joints are rotational.

Yan and Pollefeys [2006] present an approach for learning the structure of an articulated object from feature trajectories under affine projections. They first segment the feature trajectories by local sampling and spectral clustering and then build the kinematic chain as a minimum spanning tree of a graph constructed from the segmented motion subspaces.

Other researchers have addressed the problem of identifying different object parts from image data. Ross *et al.* [2008] use multi-body structure from motion to extract links from an image sequence and then fit an articulated model to these links using maximum likelihood learning. There also exist approaches for identifying humans that assume a known topology of the body parts. Ramanan [2006] perform pose estimation of articulated objects from images using an iterative parsing approach. They seek to improve the feature selection to better fit the model to the image.

There exist several approaches where tracking articulated objects is the key motivation and often an a-priori model is assumed. Comport *et al.* [2004], for example, describe a framework for visual tracking of parametric non-rigid multi-body objects based on an a-priori model of the object including a general mechanical link description. Chu *et al.* [2003] present an approach for model-free and marker-less model and motion capture from visual input. Based on volume sequences obtained from image data from calibrated cameras, they derive a kinematic model and the joint angle motion of humans with tree-structured kinematics.

Similar to our approach for identifying low-dimensional articulation actions, Tsoli and Jenkins [2009] presented an Isomap-based technique that finds a low-dimensional representation of complex grasp actions. This allows human operators to easily carry out remote grasping tasks.

Katz *et al.* [2008] learn planar kinematic models for articulated objects such as 1-DOF scissors or a 3-DOF snake-like toy. They extract features from a series of camera images, that they group together to coherently moving clusters as nodes in a graph. Two nodes are connected in the graph when they are rigid. Subsequently, rotational and prismatic joints are identified by searching for rotation centers or shifting movements. In contrast to their work, we use 3D information and are not restricted to prismatic and rotation joints. We additionally can model arbitrary movements including those of garage doors which are 1-DOF actions that cannot be described by a prismatic or rotational joint. The approach of Katz *et al.* [2008] is furthermore focused on manipulation actions whereas our approach is passive and only based on observations.

3 Learning Models of Actuated Objects

In this work, we consider articulated objects consisting of n rigid object parts, which are linked mechanically as an open kinematic chain. We assume that a robot, external to the object, observes the individual parts and that it has no prior knowledge about their connectivity.

To describe the kinematics of such an articulated object, we need to reason about (i) the connections of the object parts (the topology) and (ii) the kinematic nature of the connections. Our approach seeks to find the topology and the local models that best explain the observations. We begin with a discussion of how to model the relationship of *two* object parts. The extension towards an entire graph of parts and relations is then given in Section 3.3.

3.1 Modeling the Interaction between Two Parts

The state of an object part i can be described by a vector $x_i^t \in \mathbb{R}^6$ representing the position and orientation of the part $i \in 1, \ldots, n$ at time $t = 1, \ldots, T$. We assume that only their relative transformation $\Delta_{ij} = x_i \ominus x_j$ is relevant for estimating the model, where $\oplus$ and $\ominus$ are the motion composition operator and its inverse.

If the two object parts are not rigidly connected, we assume that the articulation can be described by a latent (not observed) action variable. Examples for a latent action variable are the rotation angle of a door or the translation of a drawer. The goal is now to describe the relative transformation between the object parts using such a latent variable $a_{ij} \in \mathbb{R}^d$, where d represents the intrinsic DOF of the connection between i and j.

Since we have no prior information about the nature of the connection, we do not aim to fit a single model but instead aim to fit a set of candidate template models representing different kinds of joints. This candidate set consists of parameterized models that occur in various objects including a rotational joint ($\mathcal{M}^{\text{rotational}}$), a prismatic joint ($\mathcal{M}^{\text{prismatic}}$), and a rigid transformation ($\mathcal{M}^{\text{rigid}}$). Additionally, there may be articulations that do not correspond to these standard motions, for which we consider parameter-free models ($\mathcal{M}^{\text{LLE/GP}}$). These are computed by using a combination of the local linear embedding (LLE) dimensionality reduction technique and a Gaussian process. A more detailed description of these models is given in Section 3.4

We use a sequence of T noisy observations $z_{ij}^{1:T} = z_{ij}^1, \ldots, z_{ij}^T$ of Δ_{ij} for fitting the candidate models and for evaluating which model appears to be the best one. This is done by performing 2-fold cross-validation. In the remainder of this paper, we refer to $\mathcal{D}_{ij}$ as the training data selected from the observations, where $\mathcal{D}_{ij} \subset z_{ij}^{1:T}$, and to $\mathcal{D}_{ij}^{\text{test}}$ as the (disjoint) set of test data.

3.2 Evaluating a Model

Let $\mathcal{M}_{ij}$ be an articulation model $p(\Delta_{ij} \mid a)$ describing the connection between the part i and j and learned from the training data $\mathcal{D}_{ij}$. To actually evaluate how well an observation z_{ij} can be explained by a model, we have to determine

$p(z_{ij} \mid \mathcal{M}_{ij})$ which corresponds to

$$p(z_{ij} \mid \mathcal{M}_{ij}) = \int_a p(z_{ij} \mid a, \mathcal{M}_{ij})\, p(a \mid \mathcal{M}_{ij})\, da. \quad (1)$$

The variable a is the latent action variable of the model that, for example, describes the opening angle of a door.

We assume that during the observations, there is no latent action state a that is more likely than another one, i.e., that $p(a \mid \mathcal{M}_{ij})$ is a uniform distribution. Note that this is an approximation since in our door example, one might argue that doors are more likely to be closed or completely opened compared to other states. This assumption simplifies Eq. 1 to

$$p(z_{ij} \mid \mathcal{M}_{ij}) = \int_a p(z_{ij} \mid a, \mathcal{M}_{ij})\, da. \quad (2)$$

To evaluate $p(z_{ij} \mid a, \mathcal{M}_{ij})$, that is, a measure for how well model $\mathcal{M}_{ij}$ parameterized by the action variable a explains the observation z_{ij} of the part transformation Δ_{ij}, we first compute the expected transform

$$\hat{\Delta}_{ij} = \mathbb{E}_{\mathcal{M}_{ij}}[\Delta_{ij} \mid a] = f_{\mathcal{M}_{ij}}(a) \quad (3)$$

using a model-specific transformation function $f_{\mathcal{M}_{ij}}(a)$. In Section 3.4, we will specify this transformation function for all model templates. Note that we reason about the *relative* configuration between object parts here and compare the result to the observed transformation under a Gaussian error assumption with variance σ^2:

$$p(z_{ij} \mid a, \mathcal{M}_{ij}) \propto \exp\left(-||\hat{\Delta}_{ij} - z_{ij}||^2/\sigma^2\right) \quad (4)$$

To actually compute $p(z_{ij} \mid \mathcal{M}_{ij})$ using Eq. 2, we need to compute the integral over the latent action variable a. In this paper, we solve this by performing Monte-Carlo integration by sampling multiple instances of the latent variable.

Since this procedure can be rather time-consuming, we also tested an alternative strategy to approximate the integral. If we assume that $p(z_{ij} \mid a, \mathcal{M}_{ij})$ is unimodal, we can think of evaluating it only at the most likely latent action variable and approximate Eq. 2 by

$$p(z_{ij} \mid \mathcal{M}_{ij}) \quad = \quad \max_a p(z_{ij} \mid a, \mathcal{M}_{ij}). \quad (5)$$

Depending on the realization of the model $\mathcal{M}_{ij}$, we can carry out the maximization step to compute $p(z_{ij} \mid \mathcal{M}_{ij})$ efficiently.

Finally, we can compute the data likelihood for the test data set

$$p(\mathcal{D}_{ij}^{\text{test}} \mid \mathcal{M}_{ij}) = \prod_{z_{ij} \in \mathcal{D}_{ij}^{\text{test}}} p(z_{ij} \mid \mathcal{M}_{ij}). \quad (6)$$

3.3 Finding the Connectivity

So far, we ignored the question of connectivity and described how to evaluate a model $\mathcal{M}_{ij}$ representing a connection between the parts i and j. If we consider the individual object parts as nodes in a graph and the connections as edges between nodes, then the set of possible acyclic object structures that connect all parts is given by all spanning trees of this graph. The endeavor of explicitly computing, evaluating, and reasoning with all possible topologies, however, results in an intractable complexity. We therefore seek to find the *spanning tree* $\mathbb{M}$ that results in a combined model for all object parts that both maximizes the expected data likelihood of a new observation, i.e.,

$$p(\mathcal{D}^{\text{test}} \mid \mathbb{M}) = \prod_{\mathcal{M}_{ij} \in \mathbb{M}} p(\mathcal{D}_{ij}^{\text{test}} \mid \mathcal{M}_{ij}), \quad (7)$$

while at the same time minimizing the overall complexity of the combined model. The latter is calculated in a fashion similar as with the Bayesian information criterion. In our case, we measure the model complexity by the dimensionality of the latent action space.

To find this topology (that is, the spanning tree of the local models), we fit for all tuples of rigid parts all models from the candidate template model set and add for each model a link to the graph. We then assign to each edge in the graph the cost of model $\mathcal{M}_{ij}^{\text{type}}$ that is equal to the negative expected data log-likelihood plus a complexity penalty of the model:

$$\text{cost}_{\mathcal{M}_{ij}^{\text{type}}} = -\frac{1}{\|\mathcal{D}^{\text{test}}\|} \log p(\mathcal{D}^{\text{test}} \mid \mathcal{M}_{ij}^{\text{type}}) + C(\mathcal{M}_{ij}^{\text{type}}) \quad (8)$$

Then, the task of finding the topology of local models which minimize this cost function is equivalent to finding the minimal spanning tree in this graph which can be done rather efficiently.

Please note that the resulting kinematic tree can be transformed into a Bayes net (BN) by replacing the edges by connected nodes representing local models $\mathcal{M}$ and latent action variables a and by adding nodes for (absolute) object part observations and relative observations z. The resulting BN naturally encodes all independence assumptions made in our work. Such a BN, however, is complex and hard to visualize. We therefore stick at this point to a graph-like visualization as shown in the left image of Fig. 2. Bold arrows indicate the selected models form the spanning tree structure. The plot in the middle illustrates the prediction error of all considered models during learning. The right image depicts the probability density function of the model $\mathcal{M}^{\text{LLE/GP}}$.

3.4 Model Templates

This section explains the instances of the set of candidate model templates.

Rigid Transformation Model

The simplest connection between two parts is a rigid transformation without any latent action variable. The model-specific transformation function of Eq. 3 for the rigid transform model $\mathcal{M}^{\text{rigid}}$ from training data $\mathcal{D}$ then reduces to the estimating the mean, i.e.,

$$f_{\mathcal{M}_{ij}^{\text{rigid}}} = \frac{1}{\|\mathcal{D}_{ij}\|} \sum_{z_{ij} \in \mathcal{D}_{ij}} z_{ij}. \quad (9)$$

Prismatic Joint Model

For modeling prismatic joints that can be, for example, found in a drawer, we assume a 1-DOF latent action variable that describes the motion between the object parts. Prismatic joints

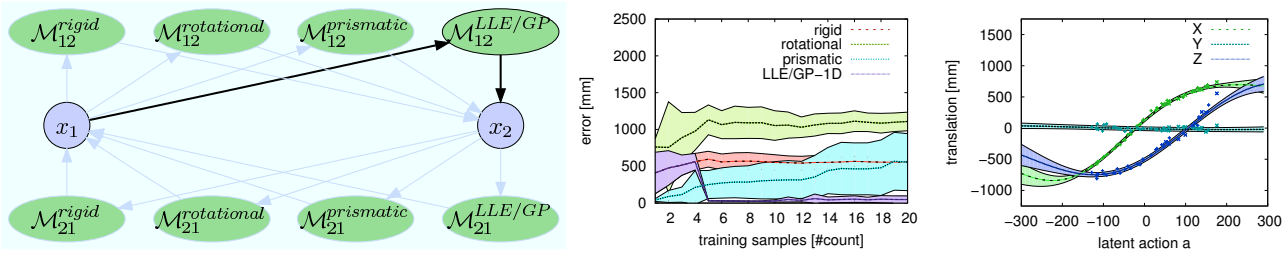


Figure 2: Learning the kinematic model for a garage door. Left: The fully connected graph contains instantiations of all possible template models and the selected models are indicated by bold arrows. Middle: evolution of the data likelihood of different models. Right: transformation function learned by the LLE/GP model.

move along a single axis, that can for example be found using principle component analysis. Internally, we model the action a_{ij}^t as the relative movement with respect to the first observation z^1 in $\mathcal{D}$ (therefore $a_{ij}^1 = 0$) along its principal axis e of unit length. Let $trans$ be the function that removes all rotational components, we obtain:

$$\hat{a}_{ij}^t = e \cdot trans(\Delta_{ij}^t - \Delta_{ij}^1) \tag{10}$$

The model-specific transformation function for the prismatic model $\mathcal{M}^{\text{prismatic}}$ then becomes

$$f_{\mathcal{M}_{ij}^{\text{prismatic}}}(a) = ae + \Delta^1. \tag{11}$$

Rotational Joint Model

In the case of a rotational joint, we compute the latent 1-DOF action variable from Eq. 5 by taking the first observation as a reference (similar as in the prismatic joint model). The rotational components describe a line, whose direction e can be found by principle component analysis. We computing the angular difference of all observations relative to the first one

$$\hat{a}_{ij}^t = e \cdot angle(\Delta_{ij}^t - \Delta_{ij}^1). \tag{12}$$

Here, $angle$ is a function that removes all non-rotational components.

Since our model assumes a 1-DOF latent action variable, the positions of the observed parts describe a circular arc or a single point in case the observed object part lies on the axis of rotation. By standard geometric operations, we estimate the axis of rotation $n \in \mathbb{R}^3$, the rotational center $c \in \mathbb{R}^3$, and the rigid transform $r \in \mathbb{R}^6$ carried out after the rotation. Then, the model-specific transformation function for the rotational model $\mathcal{M}^{\text{rotational}}$ becomes

$$f_{\mathcal{M}_{ij}^{\text{rotational}}}(a) = [c; n]^T \oplus rot_Z(a) \oplus r, \tag{13}$$

where $rot_Z(a)$ describes a rotation about the Z axis by a and $\oplus$ is the motion composition operator.

LLE/GP Joint Model

Although rigid transformations in combination with rotational and prismatic joints might seem at the first glance to be sufficient for a huge class of kinematic objects, it turns out that many real-world objects lack a clear shifting or rotation axis. One example for such objects is a garage door. Therefore, our candidate model template set contains one non-parametric model that is able to describe general transformations. This model is based on non-linear dimensionality reduction via local linear embedding for discovering the latent action manifold and a Gaussian process regression to learn a generative model.

Consider the manifold that is described by the observations of object poses in $\mathcal{D}_{ij} = \Delta_{ij}^1, \ldots, \Delta_{ij}^T$ for the link between rigid part i and j. Depending on the DOF d of this particular link, all data samples will lie on or close to a d-dimensional manifold with $1 \leq d \leq 6$ being non-linearly embedded in $\mathbb{R}^6$. There are many dimensionality reduction techniques such as PCA for linear manifolds or Isomap [Tenenbaum *et al.*, 2000] and LLE [Roweis and Saul, 2000] for non-linear manifolds. Our current implementation applies LLE but is not restricted to this method. LLE first expresses each data point as a linear combination of its neighbors, here in $\mathbb{R}^6$, and then computes a low-dimensional representation in $\mathbb{R}^d$ satisfying the identical linear relationships.

In more detail, LLE first finds the k-nearest neighbors of each data sample Δ^t in $\mathcal{D}$ (we neglect the indices i and j for a better readability here). For each data sample, LLE then computes a vector of weights that best reconstructs the data sample Δ^t from its neighbors. Let W be the weight matrix for all samples. LLE seeks for the weight matrix that minimizes the reconstruction error ε given by

$$\varepsilon(W) = \sum_t \|\Delta^t - \sum_{t'} W_{tt'} \Delta^{t'}\|^2. \tag{14}$$

By normalization, we require that the reconstruction weights for each data sample t to sum to one over its neighbors, i.e., $\sum_{t'} W_{tt'} = 1$. Minimizing Eq. 14 can be achieved via Lagrange minimization in closed form.

After determining the reconstruction weight matrix, LLE seeks for a point-wise mapping of each data sample Δ^t to a local coordinate a^t on the d-dimensional manifold. This mapping has to ensure that the weight matrix W reconstructs also the local coordinates of the data samples on the manifold. This is done by searching for the local coordinates $a^1, \ldots, a^T$ for $\Delta^1, \ldots, \Delta^T$ so that the reconstruction error Ψ

$$\Psi(a^1, \ldots, a^T) = \sum_t \|a^t - \sum_{t'} W_{tt'} a^{t'}\|^2, \tag{15}$$

on the manifold is minimized.

With a few additional constraints, the minimization of Eq. 15 can be solved as a sparse $T \times T$ eigenvector problem.

The local coordinates are then computed based on the eigenvectors. For further detail, we refer the reader to Roweis and Saul [2000].

The reconstructed latent action values can now be used for learning $p(z \mid a, \mathcal{M})$ from the training data $\mathcal{D}$. In our work, we employ Gaussian process regression, which is a powerful and flexible framework for non-parametric regression. For the sake of brevity, we refer the interested reader to Rasmussen and Williams [2006] for details about GP regression.

4 Experiments

To evaluate our approach, we recorded observations from two typical household objects, a microwave door and a cabinet with two drawers. To track the poses and orientations of the parts, we placed the objects in a PhaseSpace motion capture studio. For each object, we recorded 200 data samples while manually articulating the object. Additionally, we simulated a garage door as a typical object that cannot be described using a prismatic or rotational joint. We also estimated the model of a table moved on the ground plane to give an example of latent action variables with more than one dimension.

Our experiments are designed so that we can recover accurate transformation models for each link between parts along with the kinematic structure. In addition, we show that the range of the latent action space can be estimated and configurations of this range can be generated for visual inspection.

Model Selection

We evaluated the prediction accuracy and the expected data likelihood for each of the microwave, the drawer, and the garage door dataset for all models of out candidate set. For the evaluation, we carried out 10 runs and in each run, 40 observations were drawn independently and randomly from the data set, 20 of them were used for learning and 20 for testing. The quantitative results showing the prediction error of the models are depicted in Tab. 1. As can be seen, the flexible LLE/GP model can fit all objects well.

As can be seen from the table, the rotational model predicts best the opening movement of the microwave door while the prismatic model predicts best the motion of the drawer which is the expected result. It should be noted that the LLE/GP model is only slightly worse than the parametric models and is able to robustly predict the poses of the door and the drawer (1.1mm vs. 1.5mm for the microwave, and 0.7mm vs. 3.6mm for the drawer).

In the case of the garage door, however, all parametric models fail whereas the LLE/GP model, designed to describe general transformations, provides accurate estimates. Here we evaluated different levels of noise, and found that the LLE/GP model to be quite robust. Fig. 3 illustrates the motion of the garage door estimated by the non-parametric model. Note that our models also encode the range of the latent action variable a learned from observations.

This experiment shows that our system takes advantage of the expert-designed parametric models when appropriate while keeping the flexibility to also learn accurate models for unforeseen mechanical constructions.

dataset	error of $\mathcal{M}^{\text{rotational}}$		error of $\mathcal{M}^{\text{prismatic}}$		error of $\mathcal{M}^{\text{LLE/GP}}$	
microwave	**1.1mm** ±0.2mm	**0.1°** ±0.1°	65.9mm ±13.7mm	23.1° ±2.2°	**1.5mm** ±1.7mm	**0.2°** ±0.2°
drawer	67.2mm ±24.4mm	1.6° ±0.4°	**0.7mm** ±0.1mm	**0.9°** ±0.1°	**3.6mm** ±6.2mm	**0.6°** ±0.1°
garage door (no noise)	1059.4mm ±147.4mm	0.0° ±0.0°	382.3mm ±265.0mm	25.1° ±3.9°	**8.5mm** ±9.2mm	**0.4°** ±0.4°
(noise .1σ)	1052.6mm ±146.1mm	0.2° ±0.0°	507.7mm ±353.1mm	25.0° ±3.7°	**18.2mm** ±27.1mm	**0.9°** ±1.3°
(noise 1σ)	1108.3mm ±126.6mm	2.6° ±1.0°	555.9mm ±387.4mm	26.5° ±3.4°	**47.9mm** ±49.2mm	**2.8°** ±2.2°
(noise 10σ)	934.4mm ±289.4mm	27.4° ±11.8°	510.5mm ±239.2mm	28.9° ±1.8°	**248.3mm** ±24.1mm	**16.5°** ±1.8°

Table 1: Average prediction error and standard deviation of local models on test data over all runs. Gaussian noise with $\sigma = (10\text{mm}, 1^\circ)$ was added to the garage door data.

Structure Discovery

A typical articulated object consisting of multiple parts is a cabinet with drawers as illustrated in the left image of Fig. 4. In the experiment, we obtain pose observations of three rigid parts x_1, x_2, and x_3. First, we opened and closed only the lower drawer. Accordingly, a prismatic joint model is learned for link Δ_{13} (see top left image of Fig. 4). When also the upper drawer gets opened and closed, the rigid transform at Δ_{12} is replaced by a second prismatic joint model $\mathcal{M}^{\text{prismatic}}$, resulting in a kinematic tree. As a second multi-part object we present a yard stick, consisting of four consecutive elements with three rotational links, as depicted in Fig. 5. These experiments demonstrate that by using the data likelihood for selecting the minimum spanning tree we are able to infer the correct kinematic structure.

Multi-dimensional Latent Action Spaces

To illustrate that our approach is also able to find the models with a higher-dimensional latent action variable, we let the robot monitor a table that was moved on the ground plane. The robot is equipped with a monocular camera tracking a marker attached to the table. In this experiment, the table was only moved and was never turned, lifted, or tilted and therefore the action variable will have 2-DOF. Fig. 6 shows four snapshots during learning. Initially, the table is perfectly explained as a rigid part of the room (top left). Then, a prismatic joint model best explains the data since the table was moved in one direction only (top right). After moving sideways, the best model is a 1-DOF LLE/GP that follows a simple curved trajectory (bottom left). Finally full planar movement is explained by a 2-DOF LLE/GP model (bottom right).

Simplified Likelihood Computation

To evaluate the likelihood of a model one has to integrate over the latent variable a (see Eq. 2) which is done via Monte-Carlo integration. If we instead use the approximation shown in Eq. 5, we only need to evaluate one single action variable. In our current implementation, this speeds up the required computation time by a factor of 100 while both approaches select the same model. Even though the actual values for the likelihood differ slightly, we were unable to produce a dataset in which both strategies select different models.

5 Conclusions

In this paper, we presented a novel approach for learning kinematic models of articulated objects. Our approach infers the connectivity of rigid parts that constitute the object

including the articulation models of the individual links. To model the links, our approach considers both, parameterized as well as parameter-free representations. It combines nonlinear dimensionality reduction and Gaussian process regression to find low-dimensional manifolds that best explain the observations. Our approach has been implemented and tested using real data. In practical experiments, we demonstrated that our approach enables a robot to infer accurate articulation models for different everyday objects.

References

[Chu *et al.*, 2003] C.-W. Chu, O.C. Jenkins, and M.J. Matarić. Markerless kinematic model and motion capture from volume sequences. In *CVPR*, 2003.

[Comport *et al.*, 2004] A.I. Comport, E. Marchand, and F. Chaumette. Object-based visual 3d tracking of articulated objects via kinematic sets. In *Workshop on Articulated and Non-Rigid Motion*, 2004.

[Dearden and Demiris, 2005] A. Dearden and Y. Demiris. Learning forward models for robots. In *IJCAI*, 2005.

[Katz *et al.*, 2008] D. Katz, Y. Pyuro, and O. Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *Robotics: Science and Systems*, 2008.

[Kirk *et al.*, 2004] Adam Kirk, James F. O'Brien, and David A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *SIGGRAPH*, 2004.

[Ramanan, 2006] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2006.

[Rasmussen and Williams, 2006] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

[Ross *et al.*, 2008] David A. Ross, Daniel Tarlow, and Richard S. Zemel. Unsupervised learning of skeletons from motion. In *ECCV '08*, 2008.

[Roweis and Saul, 2000] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[Sturm *et al.*, 2008a] J. Sturm, C. Plagemann, and W. Burgard. Adaptive body scheme models for robust robotic manipulation. In *Robotics: Science and Systems*, 2008.

[Sturm *et al.*, 2008b] J. Sturm, C. Plagemann, and W. Burgard. Unsupervised body scheme learning through self-perception. In *ICRA*, 2008.

[Taycher *et al.*, 2002] L. Taycher, J.W. Fisher III, and T. Darrell. Recovering articulated model topology from observed rigid motion. In *NIPS*, 2002.

[Tenenbaum *et al.*, 2000] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[Tsoli and Jenkins, 2009] A. Tsoli and O.C. Jenkins. Neighborhood denoising for learning high-dimensional grasping manifolds. In *IROS*, 2009.

[Yan and Pollefeys, 2006] J. Yan and M. Pollefeys. Automatic kinematic chain building from feature trajectories of articulated objects. In *CVPR*, 2006.

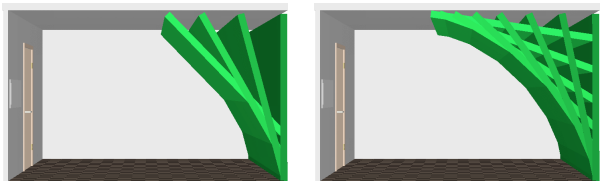


Figure 3: Motion of a garage door predicted by our non-parametric model. Left: Model after the first few observations. Right: after processing all observations.

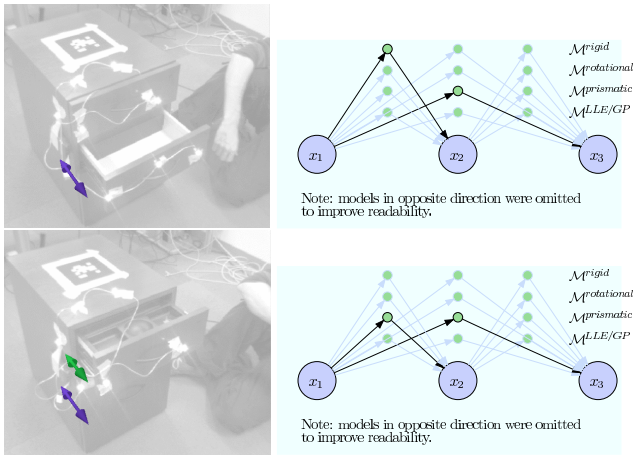


Figure 4: Estimating a model of two drawers of a cabinet. Top: initially, only the lower drawer is opened and closed and the corresponding kinematic structure is inferred. Bottom: both drawers are opened and closed independently.

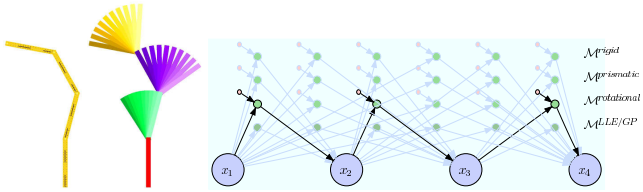


Figure 5: simulated yard stick consisting of 4 consecutive elements. Model selection correctly reveals the sequential chain.

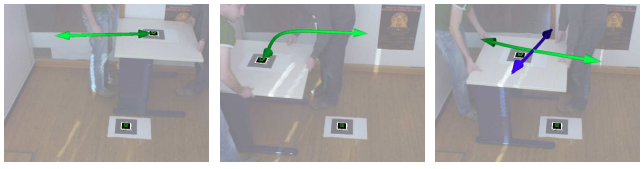


Figure 6: Learning a model for a table moving on the ground plane. Arrows indicate the dimensions of the latent action.

[C2] F. Enders, C. Plagemann, C. Stachniss, and W. Burgard. Scene analysis using latent dirichlet allocation. In *Proc. of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

Unsupervised Discovery of Object Classes from Range Data using Latent Dirichlet Allocation

Felix Endres Christian Plagemann Cyrill Stachniss Wolfram Burgard

***Abstract*— Truly versatile robots operating in the real world have to be able to learn about objects and their properties autonomously, that is, without being provided with carefully engineered training data. This paper presents an approach that allows a robot to discover object classes in three-dimensional range data in an unsupervised fashion and without a-priori knowledge about the observed objects. Our approach builds on Latent Dirichlet Allocation (LDA), a recently proposed probabilistic method for discovering topics in text documents. We discuss feature extraction, hypothesis generation, and statistical modeling of objects in 3D range data as well as the novel application of LDA to this domain. Our approach has been implemented and evaluated on real data of complex objects. Practical experiments demonstrate, that our approach is able to learn object class models autonomously that are consistent with the true classifications provided by a human. It furthermore outperforms unsupervised method such as hierarchical clustering that operate on a distance metric.**

I. Introduction

Home environments, which are envisioned as one of the key application areas for service robots, typically contain a variety of different objects. The ability to distinguish objects based on observations and to relate them to known classes of objects therefore is important for autonomous service robots. The identification of objects and their classes based on sensor data is a hard problem due to the varying appearances of the objects belonging to specific classes. In this paper, we consider a robot that can observe a scene with a 3D laser range scanner. The goal is to perform

- unsupervised learning of a model for object classes,
- consistent classification of the observed objects, and
- correct classification of unseen objects belonging to one of the known object classes.

Figure 1 depicts a typical point cloud of a scene considered in this paper. It contains four people, a box, and a balloon-like object. The individual colors of the 3D data points illustrate the corresponding object classes that we want our algorithm to infer.

An important distinction between different approaches to object detection and recognition is the way the objects or classes are modeled. Models can be engineered manually, learned from a set of labeled training data (supervised learning) or learned from unlabeled data (unsupervised learning). While the former two categories have the advantage that detailed prior knowledge about the objects can be included easily, the effort for manually building the model or labeling a significant amount of training data becomes infeasible with increasing model complexity and larger sets of objects to identify. Furthermore, in applications where the objects to distinguish are not known beforehand, a robot needs to build its own model, which can then be used to classify the data.

F. Endres, C. Stachniss, and W. Burgard are with the University of Freiburg, Germany. C. Plagemann is with Stanford University, CA, USA.

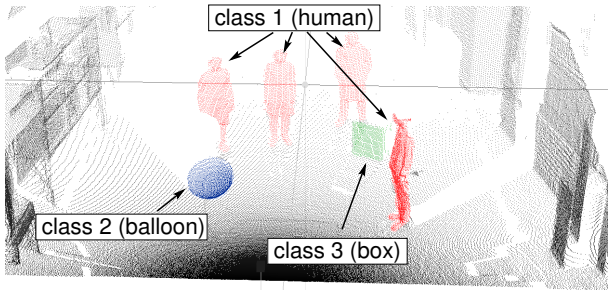


Fig. 1: Example of a scene observed with a laser range scanner mounted on a pan-tilt unit. Points with the same color resemble objects belonging to the same class (best viewed in color).

The contribution of this paper is a novel approach for discovering object classes from range data in an unsupervised fashion and for classifying observed objects in new scans according to these classes. Thereby, the robot has no a-priori knowledge about the objects it observes. Our approach operates on a 3D point cloud recorded with a laser range scanner. We apply Latent Dirichlet Allocation (LDA) [2], a method that has recently been introduced to seek for topics in text documents [9]. The approach models a distribution over feature distributions that characterize the classes of objects. Compared to most popular unsupervised clustering methods such as k-means or hierarchical clustering, no explicit distance metric is required. To describe the characteristics of surfaces belonging to objects, we utilize spin-images as local features that serve as input to the LDA. We show in practical experiments on real data that a mobile robot following our approach is able to identify similar objects in different scenes while at the same time labeling dissimilar objects differently.

II. Related Work

The problem of classifying objects and their classes in 3D range data has been studied intensively in the past. Several authors introduced features for 3D range data. One popular free-form surface descriptor are spin-images, which have been applied successfully to object recognition problems [13; 12; 14; 15]. In this paper, we propose a variant of spin-images that—instead of storing point distributions of the surface—stores the angles between the surface normals of points, which we found to yield better results in our experiments.

An alternative shape descriptor has been introduced by [18]. It relies on symbolic labels that are assigned to regions. The symbolic values, however, have to be learned from a labeled training set beforehand. Stein and Medioni [19] present a point descriptor that, similar to our approach, also relies on surface orientations. However, it focuses on the surface normals in a specific distance to the described point and models their change with respect to the angle in the tangent plane of the query point. Additional 3D shape descriptors are described in [5] and [6].

A large amount of work has focused on supervised algorithms that are trained to distinguish objects or object classes based on a labeled set of training data. For example, Anguelov *et al.* [1] and Triebel *et al.* [20] use supervised learning to classify objects and associative Markov networks to improve the results of the clustering by explicitly considering relations between the class predictions. In a different approach, Triebel *et al.* [21] use spin-images as surface descriptors and combine nearest neighbor classification with associative Markov networks to overcome limitations of the individual methods. Another approach using probabilistic techniques and histogram matching has been presented by Hetzel *et al.* [10]. It requires a complete model of the object to be recognized, which is an assumption typically not fulfilled when working on 3D scans recorded with a laser range finder. Ruhnke *et al.* [17] proposed an approach to reconstructing full 3D models of objects by registering several partial views. The work operates on range images from which small patches are selected based on a region of interest detector.

In addition to the methods that operate on 3D data, much research has also focused on image data as input. A common approach to locate objects in images is the sliding window method [4; 7]. Lampert *et al.* [16] proposed a new framework that allows to efficiently find the optimal bounding box without applying the classification algorithm explicitly to all possible boxes. Another prominent supervised detector is the face detector presented by Viola and Jones [22]. It computes Haar-like features and applies AdaBoost to learn a classifier.

In the domain of unsupervised classification of text documents, several models that greatly surpass mere counting of words have been proposed. These include probabilistic latent semantic indexing (PLSI) [11] and Latent Dirichlet Allocation [2], which both use the co-occurrence of words in a probabilistic framework to group words into topics. In the past, LDA has also been applied successfully to image data. In contrast to text documents [9], images often contain data of many different categories. Wang and Grimson [23], therefore, first perform a segmentation before applying LDA. Bosch *et al.* [3] used PLSI for unsupervised discovery of object distributions in image data. As shown in [8], LDA supersedes PLSI and it has been argued that the latter can be seen as a special case of LDA, using a uniform prior and maximum a posteriori estimation for topic selection. Fritz and Schiele [7] propose the sliding window approach on a grid of edge orientations to evaluate topic probabilities on subsets of the whole image. While the general approach of these papers is related to ours, to the best of our knowledge the algorithm described in this paper is the first to apply LDA on laser range data and which addresses the specific requirements of this domain.

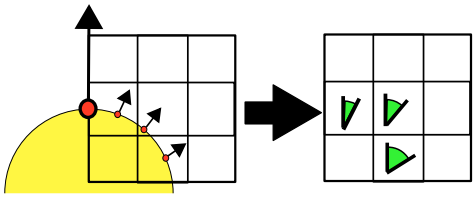


Fig. 2: Variant of spin-images used to compute a surface signature: the 3D object structure (yellow circle) is rotated around the surface normal of a query point (large red point) and a grid model accumulates the average angular distances between the surface normal at the query point and those of the points falling into the grid cells (small red points).

III. Data Pre-processing and Local Shape Features

As most approaches to object detection, identification, and clustering, we operate on local features computed from the input data. Our primary focus lies on the description of *shape* as this is the predominant feature captured in 3D range data. However, real-world objects belonging to the same class do not necessarily have the same shape and vice versa. Humans, for example, have a significant variability in shape. To deal with this problem, we model classes of objects as distributions of local shape features.

In the next sections, we first describe our local feature used to represent the characteristics of surfaces and after than, we address the unsupervised learning problem to estimate the distributions over local features.

A. Representation and Data Pre-processing

Throughout this work, we assume our input data to be a point cloud of 3D points. Such a point cloud can be obtained with a 2D laser range finder mounted on a pan-tilt unit, a standard setting in robotics to acquire 3D range data. An example point cloud recorded with this setup is shown in the motivating example in Figure 1 on the first page of this paper.

As in nearly all real world settings, the acquired data is affected by noise and it is incomplete due to perspective occlusions. The segmentation of range scans into a set of objects and background structure is not the key focus of this work. We therefore assume a ground plane as well as walls that can be easily extracted and assume the objects to be spatially disconnected. This allows us to apply a spatial clustering algorithm to create segments containing only one object.

B. Local Shape Descriptors

For characterizing the local shape of an object at a query point, we propose to use a novel variant of spin-images [12]. Spin-images can be seen as small raster images that are aligned to a point such that the upwards pointing vector of the raster image is the surface normal of the point. The image is then virtually rotated around the surface normal, “collecting” the

neighboring points it intersects. To account for the differences in data density caused by the distance between sensor and object, the spin-images are normalized.

To actually compute a normal for each data point, we compute a PCA using all neighboring points in a local region of 10cm. Then, the direction of the eigenvector corresponding to the smallest eigenvalue provides a comparably stable but smoothed estimate of the surface normal.

We have developed a variant of spin-images that does not count the points "collected" by the pixels of the raster image. Instead, we compute the average angle between the normal of the query point for which the spin-image is created and the normals of all collected points. See Figure 2 for an illustration. The average between the normals is then discretized to obtain a discrete feature space, as required in the LDA approach. As we will show in our experiments, this variant of spin-images provides better results, since they contain more information about the shape of the object.

IV. Probabilistic Topic Models for Object Shape

After segmenting the scene into a finite set of scan segments and transforming the raw 3D input data to the discrete feature space, the task is to group similar segments to classes and to learn a model for these classes. Moreover, we aim at solving the clustering and modeling problems simultaneously to achieve a better overall model. Inspired by previous work on topic modeling in text documents, we build on Latent Dirichlet Allocation for the unsupervised discovery of object classes from feature statistics.

Following this model, a multinomial distribution is used to model the distribution of discrete features in an object class. Analogously, another multinomial distribution is used to model the mixture of object classes which contribute to a scan segment. In other words, we assume a generative model, in which (i) segments generate mixtures of classes and (ii) classes generate distributions of features.

Starting from a prior distribution about these latent (i.e., hidden) mixtures, we update our belief according to the observed features. To do this efficiently, we express our prior $P(\theta)$ as a distribution that is conjugate to the observation likelihood $P(y \mid \theta)$. $P(\theta)$ being a conjugate distribution to $P(y \mid \theta)$ means that

$$P(\theta \mid y) \quad = \quad \frac{P(y \mid \theta)P(\theta)}{\int P(y \mid \theta)P(\theta)\, d\theta} \tag{1}$$

is in the same family as $P(\theta)$ itself. For multinomial distributions, the conjugate prior is the Dirichlet distribution, which we explain in the following.

A. The Dirichlet Distribution

The Dirichlet distribution is a distribution over multivariate probability distributions, i.e., a distribution assigning a probability density to every possible multivariate distribution. For the multinomial variable $\mathbf{x} = \{x_1, \ldots, x_K\}$ with K exclusive states x_i, the Dirichlet distribution is parameterized by a vector $\alpha = \{\alpha_1, \ldots, \alpha_K\}$. If $\alpha_i = 1$ for all i, the Dirichlet distribution

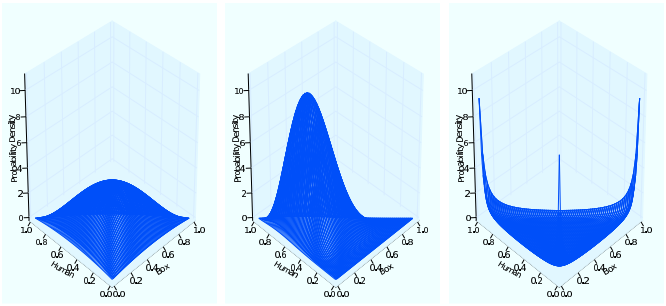


Fig. 3: Three Dirichlet distributions. On the left for the parameter vector $\alpha = \{2,2,2\}$, in the middle for $\alpha = \{3,6,3\}$ and on the right for $\alpha = \{0.1,0.1,0.1\}$.

is uniform. One can think of $(\alpha_i - 1)$ for $\alpha_i \in \mathbb{N}^{>0}$ as the number of observations of the state i. The Dirichlet distribution can be calculated as

$$f(\mathbf{x}) \quad = \quad \underbrace{\frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)}}_{\text{Normalization}} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{2}$$

where $\Gamma(\cdot)$ is the Gamma function and where the elements of $\mathbf{x}$ have to be positive and sum up to one.

Consider the following example: let there be three object classes "human", "box", and "chair" with a Dirichlet prior parameterized by $\alpha = \{2,2,2\}$. This prior assigns the same probability to all classes and hence results in a *symmetric* Dirichlet distribution. A 3D Dirichlet distribution $Dir(\alpha)$ can be visualized by projecting the the manifold where $\sum \alpha_i = 1$ to the 2D plane, as depicted in the left plot of Figure 3. Here the third variable is given implicitly by $\alpha_3 = 1 - \alpha_1 - \alpha_2$. Every corner of the depicted triangle represents the distributions where only the respective class occurs and the center point represents the uniform distribution over all classes. Now consider an observation of one human, four boxes, and a chair. By adding the observation counts to the elements of α, the posterior distribution becomes $Dir(\{5,8,5\})$ which is shown in the middle plot in Figure 3. The same result would of course occur when calculating the posterior using Eq. (1).

However choosing the values of α_i larger than 1 favors distributions that represent mixtures of classes, i.e. we expect the classes to occur together. To express a prior belief that either one or the other dominates we need to choose values smaller than 1 for all α_i. The shape of the distribution then changes in a way that it has a "valley" in the middle of the simplex and peaks at the corners. This is depicted in the right plot in Figure 3. In our setting, where a Dirichlet distribution is used to model the distribution of object classes, such a prior would correspond to the proposition that objects are typically assigned to one (or only a few) classes.

The calculation of the *expected probability distribution* over the states and can be performed easily based on α. The expected probability for x_i is given by

$$\mathbb{E}[x_i] \quad = \quad \frac{\alpha_i}{\sum_{i'} \alpha_{i'}}. \tag{3}$$

B. Latent Dirichlet Allocation

Latent Dirichlet allocation is a fully generative probabilistic model for semantic clustering of discrete data, which was developed by Blei *et al.* [2]. In LDA, the input data is assumed to be organized in a number of discrete data sets—these correspond to scan segments in our application. The scan segments contain a set of discretized features (a spin image for every 3D point). Obviously, a feature can have multiple *occurrences* since different 3D data points might have the same spin image. Often, the full set of data (from multiple scans) is referred to as "corpus". A key feature of LDA is that it does not require a distance metric between features as most approaches to unsupervised clustering do. Instead, LDA uses the co-occurrence of features in scan segments to assign them probabilistically to classes—called *topics* in this context.

Being a generative probabilistic model, the basic assumption made in LDA is that the scan segments are generated by random processes. Each random process represents an individual topic. In this work, we distinguish topics using the index j and scan segments are indexed by d. A random process generates the features in the segments by sampling them from its own specific discrete probability distribution $\phi^{(j)}$ over the features. A segment can be created by one or more topics, each topic having associated a distinct probability distribution over the features.

To represent the mixture of topics in a segment d, a multinomial distribution $\theta^{(d)}$ is used. For each feature in the segment, the generating topic is selected by sampling from $\theta^{(d)}$. The topic mixture $\theta^{(d)}$ itself is drawn from a Dirichlet distribution once for every segment in the corpus. The Dirichlet distribution represents the prior belief about the topic mixtures that occur in the corpus, i.e., whether the segments are generated by single topics or from a mixture of many topics. We express the prior belief with respect to the topic distribution using the Dirichlet parameter vector α.

Griffiths and Steyvers [9] extended LDA by additionally specifying a Dirichlet prior $Dir(\beta)$ on the conditional distributions $\phi^{(j)}$ over the features. This prior is useful in our application since it enables us to model a preference for selecting few characteristic features of a topic.

C. Learning the Model

In this section, we describe how to find the assignments of topics to 3D data points in range scans following the derivation of Griffiths and Steyvers [9]. Given the corpus $\mathbf{w} = \{w_1, w_2, ...w_n\}$ as the set of all feature occurrences, where each occurrence w_i belongs to exactly one scan segment. We are then looking for the most likely topic assignment vector $\mathbf{z} = \{z_1, z_2, ...z_n\}$ for our data $\mathbf{w}$. Here, each z_i is an index referring to topic j that generated w_i. Hence, we seek to estimate the probability distribution $P(\mathbf{z} \mid \mathbf{w})$. Based on $P(\mathbf{z} \mid \mathbf{w})$, we can then obtain the most likely topic assignment for each 3D data point. Using Bayes rule, we know that

$$P(\mathbf{z} \mid \mathbf{w}) = \frac{P(\mathbf{w} \mid \mathbf{z})P(\mathbf{z})}{P(\mathbf{w})}. \tag{4}$$

Unfortunately, the partition function $P(\mathbf{w})$ is not known and cannot be computed directly because it involves T^N terms, where T is the number of topics and N is the number of feature occurrences.

A common approach to approximate a probability distribution, for which the partition function $P(\mathbf{w})$ is unknown, is Markov chain Monte Carlo (MCMC) sampling. MCMC approximates the target distribution $P(\mathbf{z} \mid \mathbf{w})$ by randomly initializing the states of the variables—here the topic assignments. Subsequently, it samples new states using a Monte Carlo transition function leading to the target distribution. Therefore, the target distribution has to be the equilibrium distribution of the transition function. The transition function obeys the Markov property, i.e., it is independent of all states but the last. In our approach, we use Gibbs sampling as the transition function where the new state (the topic assignment) for each feature occurrence is sampled successively.

Gibbs sampling requires a *proposal distribution* to generate new states. Therefore, the next section describes how to obtain an appropriate proposal distribution for our problem.

D. Computing the Proposal Distribution for Gibbs Sampling

The proposal probability distribution over the possible topic assignments of a feature occurrence is calculated conditioned on the current assignments of the other feature occurrences. A new topic assignment is then sampled from this proposal distribution.

For estimating $P(\mathbf{z} \mid \mathbf{w})$, we successively sample from the distribution in the numerator on the right hand side of Eq. (4) the topic assignment z_i for each feature occurrence w_i given the topics of all other features. The distribution over the topics for sampling z_i is given by

$$P(z_i = j \mid \mathbf{z_{-i}}, \mathbf{w}) = \frac{\overbrace{P(w_i|z_i = j, \mathbf{z}_{-i}, \mathbf{w}_{-i})}^{\text{likelihood of } w_i} \overbrace{P(z_i = j|\mathbf{z}_{-i})}^{\text{prior of } z_i}}{\sum_{j=1}^{T} P(w_i|z_i = j, \mathbf{z}_{-i}, \mathbf{w}_{-i}) P(z_i|\mathbf{z}_{-i})}. \tag{5}$$

In Eq. (5), $\mathbf{w}_{-i}$ denotes the set $\mathbf{w}$ without w_i and $\mathbf{z_{-i}}$ the corresponding assignment vector. We can express the conditional distributions in the nominator of Eq. (5) by integrating over ϕ and θ, where ϕ denotes the feature distribution of all topics and θ denotes the topic distribution for each scan segment.

The likelihood of w_i in Eq. (5) depends on the probability of the distribution of topic j over features, so we need to integrate over all these distributions $\phi^{(j)}$:

$$P(w_i = w \mid z_i = j, \mathbf{z}_{-i}, \mathbf{w}_{-i}) = \int \underbrace{P(w_i = w \mid z_i = j, \phi^{(j)})}_{\phi_w^{(j)}} \underbrace{P(\phi^{(j)} \mid \mathbf{z}_{-i}, \mathbf{w}_{-i})}_{\text{posterior of } \phi^{(j)}} d\phi^{(j)} \tag{6}$$

Since the Dirichlet distribution is conjugate to the multinomials (to which $\phi^{(j)}$ belongs to), this posterior can be computed easily from the prior and the observations by adding the observations to the respective elements of the parameter vector β of the prior (see also Section IV-A). As a result, we obtain a Dirichlet posterior with parameter vector $\beta + n_{-i,j}^{(w)}$

where the elements of $n_{-i,j}^{(w)}$ are the number of occurrences of feature w assigned to topic j by the assignment vector $\mathbf{z_{-i}}$.

The first term on the right hand side of Eq. (6) is the probability for feature w under the multinomial $\phi^{(j)}$ and the second term denotes the probability of that multinomial. Therefore, solving this integral results in computing the expectation of $\phi_w^{(j)}$ which is the probability of w under $\phi^{(j)}$. According to Eq. (3), this expectation can be easily computed. The probability that an occurrence w_i is feature w is

$$P(w_i = w \mid z_i = j, \mathbf{z}_{-i}, \mathbf{w}_{-i}) = \mathbb{E}(\phi_w^{(j)}) = \frac{n_{-i,j}^{(w)} + \beta_w}{\sum_{w'} n_{-i,j}^{(w')} + \beta_{w'}}. \quad (7)$$

In the same way, we integrate over the multinomial distributions over topics θ, to find the prior of z_i from Eq. (5). With d_i being the index of the scan segment to which w_i belongs, we can compute the probability of a topic assignment for feature occurrence w_i as:

$$P(z_i = j \mid \mathbf{z}_{-i}) = \int \underbrace{P(z_i = j \mid \theta^{(d_i)})}_{\theta_j^{(d_i)}} \underbrace{P(\theta^{(d_i)} \mid \mathbf{z}_{-i})}_{\text{posterior of } \theta^{(d_i)}} d\theta^{(d_i)} \quad (8)$$

Let $n_{-i,j}^{(d_i)}$ be the number of features in the scan segment d_i that are assigned to topic j. Then, analogous to Eq. (7), the expected value of $\theta_j^{(d_i)}$ can be calculated by adding $n_{-i,j}^{(d_i)}$ to the elements of the parameter vector α of the prior:

$$P(z_i = j \mid \mathbf{z}_{-i}) = \mathbb{E}(\theta_j^{(d_i)}) = \frac{n_{-i,j}^{(d_i)} + \alpha_j}{\sum_{j'} n_{-i,j'}^{(d_i)} + \alpha_{j'}} \quad (9)$$

Combining the results of Eq. (7) and (9) in Eq. (5), we obtain the *proposal distribution* for the sampling of z_i as

$$P(z_i = j \mid \mathbf{z_{-i}}, \mathbf{w}) \propto \frac{n_{-i,j}^{(w)} + \beta_w}{\sum_{w'} n_{-i,j}^{(w')} + \beta_{w'}} \frac{n_{-i,j}^{(d_i)} + \alpha_j}{\sum_{j'} n_{-i,j'}^{(d_i)} + \alpha_{j'}} \quad (10)$$

Eq. (10) is the proposal distribution used in Gibbs sampling to obtain next generation of assignments.

After a random initialization of the Markov chain, a new state is generated by drawing the topic for each feature occurrence successively from the proposal distribution. From these samples, the distributions θ and ϕ can be estimated by using the sampled topic assignments $\mathbf{z}$.

Note that in our work, we restrict the Dirichlet priors to be symmetric. This implies that all topics and all features have the same initial prior occurrence probability. As a result, we only have to specify only value for the elements of the parameter vectors α and β which we denote by $\hat{\alpha}$ and $\hat{\beta}$. This leads to:

$$\phi_j^{(w)} \sim \frac{n_j^{(w)} + \hat{\beta}}{\left(\sum_{w'} n_j^{(w')}\right) + W\hat{\beta}} \qquad \theta_j^{(d)} \sim \frac{n_j^{(d)} + \hat{\alpha}}{\left(\sum_{j'} n_{j'}^{(d)}\right) + T\hat{\alpha}} \quad (11)$$

where T is the number of topics and W the number of features.

To summarize, we explained how to compute the proposal distribution in Eq. (10) used in Gibbs sampling during MCMC. The obtained samples can then be used to estimate the distributions ϕ and θ. Due to our restriction to symmetric priors, only two parameters ($\hat{\alpha}, \hat{\beta} \in \mathbb{R}$) have to be specified.

E. Unsupervised Topic Discovery and Classification of Newly Observed Objects

This section briefly summarizes how the components presented so far are integrated to perform the unsupervised discovery of object classes and the classification when new observations are made.

First of all, we preprocess the data according to Section III-A to extract the scan segments which correspond to objects in the scene and for which we aim to learn a topic model. For each data point in a scan segment, we compute our feature, a variant of the spin-image, according to Section III-B to describe the surfaces characteristics.

For the discovery of topics, we then compute the feature distributions ϕ of the object classes as well as the topic mixtures θ for the scan segments using MCMC as described in the previous section. The learned distributions θ denote a probabilistic assignment of objects to topics.

Class inference, that is, the classification of objects contained in *new* scenes can be achieved using the feature distribution ϕ. In this case, ϕ and θ can be used to compute the proposal distribution directly and are not updated.

Note that the approach presented here does not automatically determine the number of object classes. This is similar to other unsupervised techniques such as k-means clustering or EM-based Gaussian mixture models in which the number of object classes is assumed to be known. We experimentally evaluated settings in which the number of topics was higher or lower than the number of manually assigned classes in the data set. Our observation was that a higher number of topics leads to the detection of shape classes such as "corner", "edge", or "flat surface" and that the objects are modeled as mixtures of those.

F. The Influence of the Dirichlet Priors $\hat{\alpha}$ and $\hat{\beta}$

Two hyperparameters $\hat{\alpha} \in \mathbb{R}$ and $\hat{\beta} \in \mathbb{R}$ need to be provided as the input to the presented approach. They define the prior distributions for the mixture of object classes in a data set and for the mixture of features in an object class respectively.

As briefly discussed in Section IV-A, choosing $\hat{\alpha}$ larger than one favors the occurrence of many topics in each scan segment, while lower values result in less topics per scan segment. Similarly, the lower the hyperparameter $\hat{\beta}$ for the Dirichlet distribution over the features, the stronger the preference for fewer features per topic and unambiguous ones. Due to the segmentation in the preprocessing step, we assume that there are only few topics per scan segment and thus a low value for the hyperparameter is favored in this setting. For $\hat{\beta}$ holds: On the one hand different objects can yield the same individual features (yet in distinct distributions). On the other hand, we expect features to be related to specific topics.

From this intuitions about the Dirichlet parameters, a high performance can be expected if both parameters are selected between zero and one. This could be confirmed experimentally and the results are given in Section V-D, where we analyze the influence of the hyperparameters on manually labeled data sets.

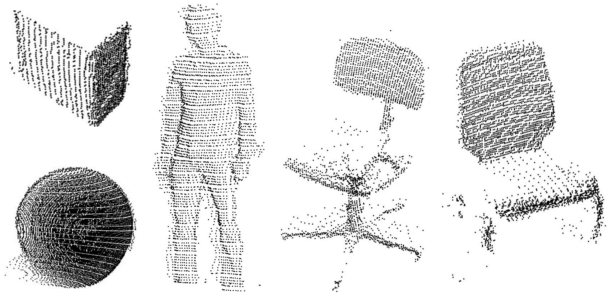


Fig. 4: Example point cloud segments of Corpus-A (box, balloon) and Corpus-B (box, balloon, human, swivel chair, chair)

V. Experimental Evaluation

In this section, we present experiments carried out to evaluate our approach on recorded data. All results are based on scans of real scenes acquired with an ActivMedia pioneer robot equipped with a SICK LMS range finder mounted on a Schunk pant-tilt unit. No simulator was involved in the evaluation.

The goal of the evaluation is to answer the following questions: (i) Are the proposed local shape features in conjunction with the topic model approach expressive enough to represent real-world objects? (ii) Is the approach able to discover object classes from unlabeled point clouds and are these classifications consistent with human-provided class labels? (iii) How does our LDA-based approach compare to conceptually simpler approaches for unsupervised clustering? (iv) How sensitive is the proposed algorithm w.r.t to the choice of parameters for the feature extraction step as well as of the Dirichlet priors?

A. Test Data

For the experimental evaluation, we prepared and rearranged indoor scenes containing five different object types: balloons, boxes, humans, and two types of chairs. In total, we recorded 51 full laser-range scans containing 121 object instances. The first part of this data set is termed *Corpus-A*. It contains 31 object instances of low geometric complexity (different boxes and balloons). The second and larger part comprising of 82 object instances, *Corpus-B*, additionally contains complex and variable shapes of chairs and humans. See Figure 4 for examples of such object segments represented as 3D point clouds.

The data was acquired and pre-processed as described in Section III-A. Some difficulties, inherent in 3D data recorded in this way, should be pointed out: Only one side of an object can be recorded and non-convex objects typically occlude themselves partially. Objects were scanned from different view points and thus different parts are observed. Different objects of the same class were scanned (different humans, different chairs, etc.). Metal parts, such as the legs of chairs, reflect the laser beams and, thus, are invisible to the sensor. Finally, local shape features extracted from the scans of humans are highly diverse compared to the simpler objects.

Figure 5 shows typical classification results achieved by our algorithm when applied to entire scans in three example scenes. Here, the points are color-coded according to their class assignments (elements of Corpus-A on the left and Corpus-B in the middle and on the right). The labels assigned to the individual points are taken from a sample of the posterior distribution $P(\mathbf{z} \mid \mathbf{w})$ as generated during the clustering process. It can be seen that the point labels are almost perfectly consistent within each object segment and, thus, the maximum likelihood class assignment per segment is unambiguous.

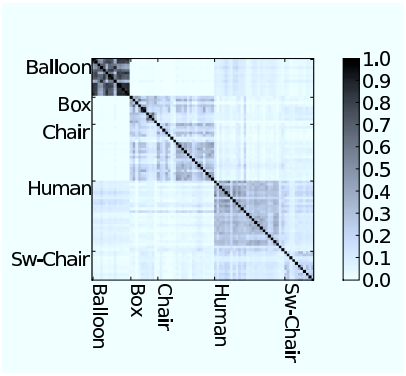


Fig. 7: Visualization of the confusion matrix of classification based on matching spin-image histograms.

In addition to that, Figure 6 gives a visual impression of the topics assigned by our approach to the 82 scan segments of Corpus-B. The labels in this diagram show the true object class. Each color in the diagram denotes one topic and the ratios of colors denote for each object segment the class assignment weight. As the diagram shows, except of one chair, all objects are grouped correctly when using the maximum likelihood assignment.

We furthermore analyzed the runtime requirements of our approach, disregarding the time for pre-processing and the computation of the spin images. In Corpus-B (82 objects from 39 different 3D scans, 300 000 spin image in total), it took less than 20 s to learn the topic distributions via MCMC and to classify the objects. Thus, the computation time per 3D scan is around 500 ms which is faster than the time needed to record a 3D scan.

B. Clustering by Matching Shape Histograms

In order to compare our LDA-based approach to an unsupervised clustering technique, we implemented hierarchical clustering (HC) using the similarity between spin-image histograms as the distance metric. In this implementation, we build a feature histogram for each object segment by counting the occurrences of the individual spin-images from the (finite) spin-image dictionary (see. Section III-B). To compare two scan segments, we first normalize their histograms to sum up to one over all bins. Among the popular measures for comparing histograms, namely histogram intersection [10], χ^2 distance, and the Kullback Leibler divergence (KL-D), histogram intersection appeared to provide the best results in our domain. This is due to the fact that the χ^2 distance and the KL-D are heavily influenced by features with few or no occurrences—an effect that can be observed frequently in our data sets. The quantitative results comparing LDA to

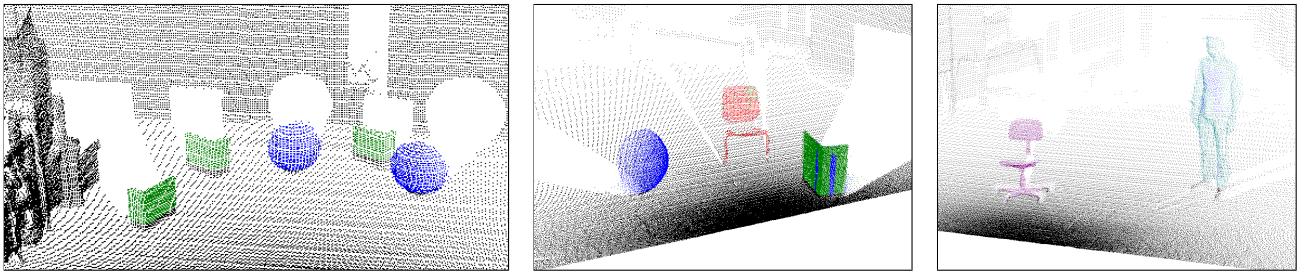


Fig. 5: Example classification results on test scans from Corpus-A (left) and Corpus-B (middle and right). The detected object classes are colored according to the LDA-assigned shape model.

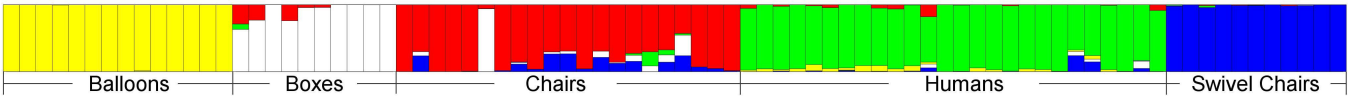


Fig. 6: Resulting topic mixtures θ for 82 segments of Corpus-B computed via LDA (the labels were not provided to the system).

HC are given in Table I. As can be seen for the simpler setting of Corpus-A, HC gives acceptable results but is still outperformed by LDA. In the more complex setting of Corpus-B, however, HC was not able to find a good clustering of the scene. In multiple runs using different setups, we found that the difference is statistically significant.

Figure 7 visualizes the similarity matrix between scan segments obtained using histogram intersection. Due to their rather uniform shape, balloons can be well distinguished from other objects. Objects with a more complex shape, however, are confused easily. This indicates that approaches working only based on such a distance metric are likely operate less accurately in more complex scenes. In contrast to that, LDA considers distributions of features and their dependencies and therefore perform substantially better.

C. Parameters of the Spin-Image Features

In this experiment, we analyzed the difference of the clustering performance when the regular spin-images (referred to as "Type 1") and our variant (referred to as "Type 2") is used. We also investigated the influence of the parameters used to create the features. These parameters are (i) the support distance, i.e., the size of the spinning image, (ii) the grid resolution, and (iii) the discretization of the stored values.

To compare the two alternative types of spin images, we collected statistics measuring the LDA clustering performance on a labeled test set, integrating over the three feature parameters. That way, we analyzed 10 780 different parameter settings—each for regular spin-images and for our variant. Figure 8 shows the results of this experiment as a histogram. The higher the bars on the right hand side of the histogram, the better the results. As can be seen, our approach outperforms regular spin-images.

TABLE I: Summary of the classification results on the test data sets. The percentages give the average correct classifications achieved by hierarchical clustering (HC) and the proposed model based on LDA.

Data set	No. of scenes	No. of segments	HC	LDA
Corpus-A	12	31	94.84%	99.89%
Corpus-B	39	82	71.19%	90.38%

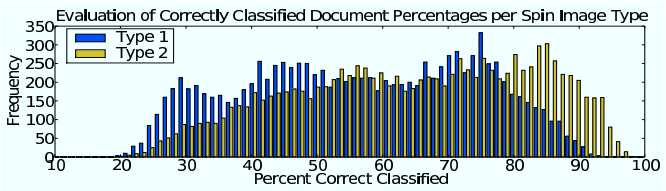


Fig. 8: Classification using standard spin-image features ("Type 1" shown in blue) generally labels less documents correctly than classification upon the features we proposed ("Type 2", yellow).

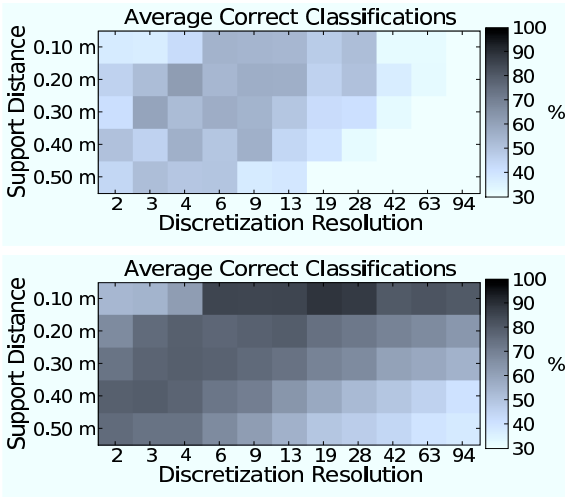


Fig. 9: Classification accuracy on Corpus-B for different discretization resolutions and respect to support distances for HC (top) and LDA (bottom).

In addition to that, we computed the clustering performance of our approach and HC for a wide variety of feature parameters using Corpus-B. Figure 9 shows the results for HC and LDA. Again, our approach clearly outperforms HC. The broad spread of high classification rates over the range of parameters demonstrates that the results presented in the previous section were not caused by selecting feature parameters that were suboptimal for HC.

We observe that for smaller support distances, a higher dis-

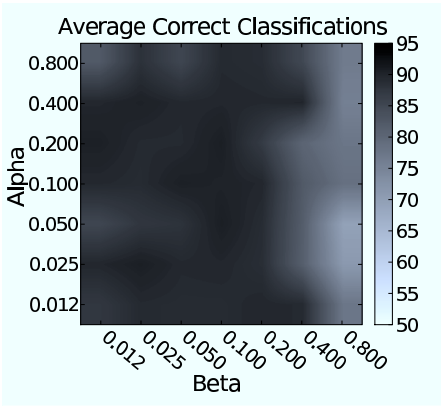


Fig. 10: Evaluation of classification accuracy for various values of alpha and beta.

cretization resolutions work well and vice versa. The intuition for this finding is that feature distributions with a large support and a very accurate discretization have overly detailed features, that do not match the distributions of other segments well.

The best results in our setting are obtained for features with a discretization resolution between 5 and 27 and a rather short support distance. In conclusion we see, that choosing such parameters for the feature generation, we can achieve over 90 % correct classifications (compare lower plot in Figure 9).

D. Sensitivity of the Dirichlet Priors

We furthermore evaluated how sensitive our approach is with respect to the choice of the parameters $\hat{\alpha}$ and $\hat{\beta}$ for the Dirichlet priors. Figure 10 depicts the average classification rates for varying parameters. In this plot, we integrate over the three feature parameters in a local region around the values determined in the previous experiment to illustrate how robust LDA performs. As can be seen from Figure 10, determining the hyperparameters is not a critical task since the performance stays more or less constant when varying them. Good values for $\hat{\alpha}$ lie between 0.1 and 0.8 and between 0.1 and 0.3 for $\hat{\beta}$. In these ranges, we always achieved close-to-optimal classification accuracies on labeled test sets.

VI. Conclusion

In this paper, we presented a novel approach for discovering object classes from laser range data in an unsupervised fashion. We use a feature-based approach that applies a novel variant of spin-images as surfaces representations but is not restricted to this kind of features. We model object classes as distributions over features and use Latent Dirichlet Allocation to learn clusters of 3D objects according to similarity in shape. The learned feature distributions can subsequently be used as models for the classification of unseen data. An important property of our approach is that it is unsupervised and does not need labeled training data to learn the partitioning.

We carried out experiments using 3D laser range data acquired with a mobile robot. Even for datasets containing complex objects with varying appearance such as humans, we achieve a robust performance with over 90% correctly grouped objects. We furthermore demonstrate that our approach clearly outperforms unsupervised clustering approaches such as hierarchical clustering. Not only does LDA achieve higher classification accuracy throughout the entire parameter range, it is also less sensitive to the choice of parameters.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *Proc. of the Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pages 169–176, 2005.

[2] D.M. Blei, A.Y. Ng, M.I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via plsa. In *In Proc. ECCV*, pages 517–530, 2006.

[4] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. of the ACM Int. Conf. on Image and Video Retrieval*, pages 401–408, 2007.

[5] B. Bustos, D.A. Keim, D. Saupe, T. Schreck, and D.V. Vranić. Feature-based similarity search in 3d object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.

[6] R.J. Campbell and P.J. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001.

[7] M. Fritz and B. Schiele. Decomposition, discovery and detection of visual categories using topic models. In *Proc. of the Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[8] M. Girolami and A. Kabán. On an equivalence between PLSI and LDA. In *Proc. of the Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 433–434, 2003.

[9] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5228–5235, 2004.

[10] G. Hetzel, B. Leibe, P. Levi, and B. Schiele. 3d object recognition from range images using local feature histograms. In *Proc. of the Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pages 394–399, 2001.

[11] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 50–57, 1999.

[12] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

[13] A. Johnson and M. Hebert. Recognizing objects by matching oriented points. Technical Report CMU-RI-TR-96-04, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1996.

[14] A.E. Johnson and M. Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16:635–651, 1998.

[15] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:433–449, 1999.

[16] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[17] M. Ruhnke, B. Steder, G. Grisetti, and W Burgard. Unsupervised learning of 3d object models from partial views. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009. To appear.

[18] S. Ruiz-Correa, L.G. Shapiro, and M. Meila. A new paradigm for recognizing 3-d object shapes from range data. *Computer Vision, IEEE International Conference on*, 2:1126, 2003.

[19] F. Stein and G. Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145, 1992.

[20] R. Triebel, K. Kersting, and W. Burgard. Robust 3d scan point classification using associative markov networks. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[21] R. Triebel, R. Schmidt, O. Martinez Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2d and 3d laser range data. In *Proc. of IJCAI*, pages 2225–2230, 2007.

[22] P. Viola and M.J. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.

[23] X. Wang and E. Grimson. Spatial latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[C3] K.M. Wurm, R. Kuemmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009.

Improving Robot Navigation in Structured Outdoor Environments by Identifying Vegetation from Laser Data

Kai M. Wurm Rainer Kümmerle Cyrill Stachniss Wolfram Burgard

***Abstract*— This paper addresses the problem of vegetation detection from laser measurements. The ability to detect vegetation is important for robots operating outdoors, since it enables a robot to navigate more efficiently and safely in such environments. In this paper, we propose a novel approach for detecting low, grass-like vegetation using laser remission values. In our algorithm, the laser remission is modeled as a function of distance, incidence angle, and material. We classify surface terrain based on 3D scans of the surroundings of the robot. The model is learned in a self-supervised way using vibration-based terrain classification. In all real world experiments we carried out, our approach yields a classification accuracy of over 99%. We furthermore illustrate how the learned classifier can improve the autonomous navigation capabilities of mobile robots.**

I. Introduction

Autonomous outdoor navigation is an active research field in robotics. In most of the outdoor navigation scenarios including autonomous cars, autonomous wheelchairs, surveillance robots, or transportation vehicles, the classification of the terrain plays an important role as most of the robots have been designed for navigation on streets or paved paths rather than on natural surfaces covered by grass or vegetation. The navigation outside of paved paths might be uncomfortable for passengers and might even introduce the risk of the robot getting stuck. Furthermore, driving on grass will in general increase wheel slippage and in this way increase potential errors in the odometry. Accordingly, the robust detection of vegetated areas is an important requirement for robots in any of the above-mentioned situations.

In this paper, we propose a novel laser-based classification approach that is especially suited for detecting low vegetation typically found in structured outdoor environments such as parks or campus sites. We classify surface terrain based on 3D scans of the surrounding of the robot in order to allow the robot to take the classification result into account during trajectory planning. It exploits an effect that is well known from satellite image analysis: Chlorophyll which is found in living plants strongly reflects near-IR light [13]. Often used laser scanners such as the SICK LMS 291-S05 scanner emit near-IR light and return the reflectivity of the object they hit. Our approach models this remission value of the laser scanner as a function of terrain class, incidence angle, and measured distance. Classification is done using a support vector machine. To integrate classification results, we apply a probabilistic mapping method similar to occupancy grid mapping [12]. The model is learned in a self-supervised way using a vibration-based classification approach to label training data. In our experiments, we demonstrate that our approach can be used to accurately map vegetated areas and do so with a higher accuracy than standard techniques that are solely based on range values (see Fig. 1). We furthermore present an application to autonomous navigation in structured outdoor environments in which a robot benefits from the knowledge about the vegetation in its surroundings.

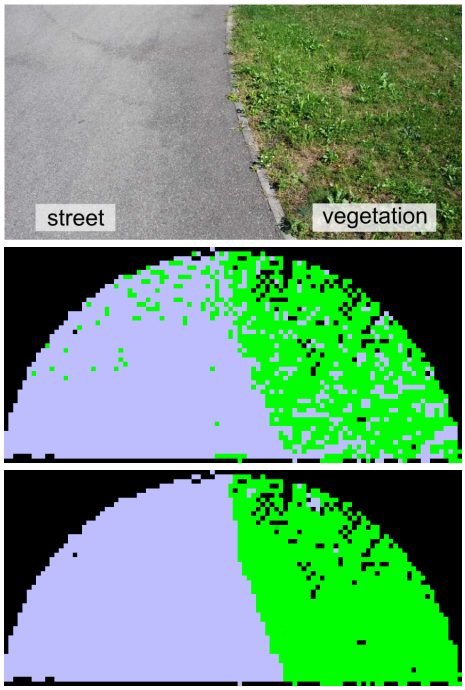


Fig. 1. Picture of a street and vegetation (top) and typical classification results obtained based on range differences (middle) and remission values (bottom). Shown is a bird's eye view of a 3D scan of the area depicted at the top with a maximum range of 5 m. Whereas points classified as street are depicted in blue, points corresponding to vegetation are colored in green.

This paper is organized as follows. After discussing related work, we will briefly describe Support Vector machines which are employed for classification in our approach. Sec. IV then discusses the properties of the remission values. In Sec. V we then present our approach for self-supervised learning of the terrain classification. Finally, in Sec. VI we describe the experimental results obtained with real data and with real robots navigating through our university campus.

All authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A3) and by the EC under contract number FP7-231888-EUROPA.

II. Related Work

There exist several approaches for detecting vegetation using laser measurements. Wolf et al. [23] use hidden Markov models to classify scans from a tilted laser scanner into navigable (e.g., street) and non-navigable (e.g., grass) regions. The main feature for classification is the variance in height measurements relative to the robot height. Other approaches analyze the distribution of 3D endpoints in a sequence of scans [8], [9], [10]. However, flat vegetation such as a freshly mowed lawn can not be reliably detected using this feature alone.

A combination of camera and laser measurements has been used to detect vegetation in several approaches [2], [6], [11], [22]. In a combined system, Wellington et al. [22] use the remission value of a laser scanner in addition to density features and camera images as a classification feature. However, they do not model the dependency between remission, measured range, and incidence angle. Probably due to this fact, they found the feature to be only "moderately informative".

The approach that is closest to our approach has been proposed by Bradley et al. [2]. Chlorophyll-rich vegetation is detected using a combination of laser range measurements, regular and near-infrared cameras. Vegetation is recognized by comparing measurements from the different types of cameras. 3D laser measurements of the environment are projected into the camera images. A classifier is then trained using the vegetation feature and features from the distribution of 3D endpoints. According to the authors the approach yields a classification accuracy of up to 95% but requires sophisticated camera equipment.

In contrast to those combined systems, our approach uses a laser scanner as its sole sensor. It is thus independent of lighting conditions and can be used on a variety of existing robot systems. Additionally, hand-labeling of training data is not required in our approach.

Terrain types have also been classified using vibration sensors on a robot [3], [7], [15], [21]. In these approaches, the robot traverses the terrain and the induced vibration is measured using accelerometers. The measurements are usually analyzed in the Fourier domain. Sadhukhan et al. [15] presented an approach based on neural networks. A similar approach is presented by DuPont et al. [7]. Brooks and Iagenemma [3] use a combination of principal component analysis and linear discriminant analysis to classify terrain. More recently, SVMs have been used by Weiss et al. [20], [21].

Self-supervised learning has previously been used by Dahlkamp et al. [5] in a vision-based road detection system. Here, laser measurements are used to identify nearby traversable surfaces. This information is then used to label camera image patches in order to train a classifier that is able to predict traversability in the far range. In our approach, we adopt the idea of self-supervision to generate labeled training data. We apply a vibration-based classifier to label laser measurements recorded by the robot. This labeled dataset is then used to train a laser-based vegetation classifier. Both classifiers used in our approach have been implemented using support vector machines which will be introduced in the following.

III. Support Vector Machines

Support vector machines (SVMs) are a kernel-based learning method which is widely used for classification and regression [16]. A SVM is essentially a hyperplane learning algorithm. Two classes of data points are separated by a hyperplane so that the margin between training points and the plane is maximized:

$$\max_{w \in H, b \in \mathbb{R}} \min\{||x - x_i|| \mid x \in H, \langle w, x \rangle + b = 0\}, \qquad (1)$$

where H is some dot product space, x_i are training points, and w is a weighting vector. The following decision function is used to determine the class label y:

$$f(x) = \langle w, x \rangle + b \qquad (2)$$

$$y = \text{sgn}(f(x)) \qquad (3)$$

The hyperplane can be constructed efficiently by solving a quadratic programming problem. To separate non-linear classes, the so-called kernel trick is applied. The training data is first mapped into a higher-dimensional feature space using a kernel function. A separating hyperplane is then constructed in this features space which yields a nonlinear decision boundary in the input space. In practice, the Gaussian Radial Basis Function (RBF) is often used as a kernel function given by

$$k(x, x') = e^{\frac{-(x-x')^2}{2l^2}}, \qquad (4)$$

with the so-called length-scale parameter l.

There exist derivatives of the basic SVM-formulation which allow for training errors. Among those, C-SVM is a popular method. An addition parameter commonly denoted as C has to be optimized which adjusts the trade-off between maximizing the margin and minimizing the training error.

Throughout this work, we use the SVM implementation of LibSVM [4].

IV. Using Remission Values for Terrain Classification

The goal of our terrain classification algorithm is to precisely classify the area containing vegetation. This classification has to be made early enough for the robot's planner to take the classification into account. For this reason, we focus on classifying three-dimensional scans of the environment surrounding the robot.

We are interested in distinguishing flat vegetation such as grass from drivable surfaces such as streets or paved paths. For sake of brevity, we will call those classes of materials "street" and "vegetation" in the following.

Compared to an approach based purely on the scan point distribution (see experiment in Sec. VI-A) a far better classification accuracy can be achieved when the remission value of laser measurements is used to classify endpoints.

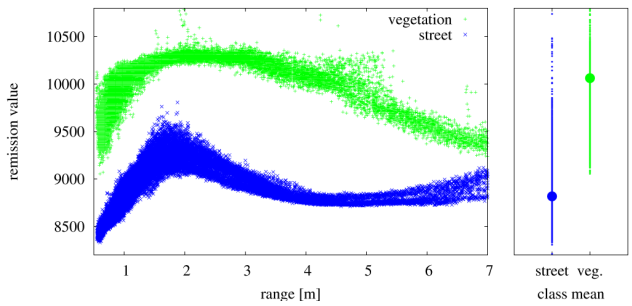


Fig. 2. Left: typical remission measurements of a SICK LMS 291-S05 for street (blue) and vegetation class (green). Right: mean remission values for both classes.

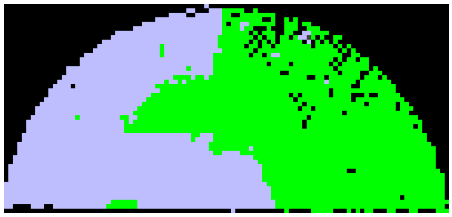


Fig. 3. Classification of the scene depicted in Fig. 1 using the mean remission value (see Fig. 2, right) to distinguish both classes.

The SICK LMS laser scanner uses light at a wavelength of 905 nm which is within the near-infrared range. Chlorophyll which is found in living plants strongly reflects near-IR light [13]. The remission values returned by the scanner depends on the material of the measured surface, on the distance at which it is hit, and on the angle of incidence [1]. Unfortunately, this relation is non-linear as the example of measurements in Fig. 2 shows. It is essential to take the measured range as well as the angle of the measurement into account. Averaging the remission value over all measured ranges and angles will lead to wrong classification results especially in the near range of up to three meters. This can be seen in Fig. 3. Here, the scene from Fig. 1 was classified using the mean remission value for both classes (see Fig. 2) ignoring the range and incidence angle.

In our approach, we apply a C-SVM to classify laser beams depending on the measured range, incidence angle, and remission using a RBF-kernel. Other classification methods might certainly be used instead of a SVM as long as they are able to classify data which is only separable in a non-linear way. The input to the learning algorithm is a labeled set of laser measurements. Each data point consists of a range measurement, incidence angle, and a remission value. By determining the separating hyperplane between both data sets, we implicitly model the remission functions for the street and vegetation class.

V. Self-Supervised Learning for Robust Terrain Classification

To avoid tedious hand-labeling of dense three-dimensional training data our algorithm uses a self-supervised training method [5]. While the robot is moving it generates a 3D model of the environment using the method described in Sec. V-B. Incidence angles are estimated using the surface normal in the 3D model as well as the angle of the laser beam relative to the robot.

For each measured scan point the remission value, incidence angle, and distance are stored. As the robot moves through the environment it traverses previously scanned surface patches. Those patches are labeled using the vibration classifier described below and are then fed to the SVM as training data. The training is done offline. To optimize the length-scale parameter l of the kernel as well as the soft-margin parameter C, we perform a systematic grid search on the grid $\log_2 l \in \{-15, ..., 3\}$ and $\log_2 C \in \{-5, ..., 15\}$. The optimal parameters are determined using 5-fold cross validation.

Once the classifier has been trained, the model can be used on any robot which is using the same sensor in a similar configuration. More specifically, the laser should be mounted so that the laser is measuring the surface at angles and distances similar to those observed during the training phase.

A. Terrain Classification based on the Vibration of the Vehicle

We use a vibration-based classifier to label training data. Different types of terrain induce vibrations of a different characteristic to the robot. These vibrations can be measured by an inertial measurement unit (IMU) and can be used to classify the terrain the robot traverses. Note that such classifiers *do not allow the prediction of terrain classes in areas which have not been traversed yet*. In our system, we only need to differentiate between vegetation and streets (non-vegetation) to generate training data for the SVM. In our experiments, we use a XSens MTi to measure the acceleration along the z-axis. We apply the Fourier transform to the raw acceleration data. In our algorithm, a 128-point FFT is used to capture the frequency spectrum of up to 25 Hz.

The frequency spectrum also depends on the speed of the robot. To account for this dependency, it has been suggested to train several classifiers at different speeds [20]. In our system, however, we decided to treat the forward velocity as a training feature instead. In addition to this, we also use the rotational velocity as a feature to account for vibrations which result from the skid-steering of our robot.

To classify the acceleration and velocity data, we again use a C-SVM with a RBF-kernel. Our feature vector x consists of the first 32 Fourier magnitudes $|F_m|$, the mean forward velocity $\bar{v_t}$ and the mean rotational velocity $\bar{v_r}$ of the robot over the sample period

$$|F_m| = \sqrt{(Re\ F_m)^2 + (Im\ F_m)^2}, m \in \{0,...,31\} \quad (5)$$

$$x = \{|F_0|, ..., |F_{31}|, \bar{v_t}, \bar{r_t}\}, \quad (6)$$

where F_m denotes the m-th Fourier component. The classifier was trained by recording short tracks of about 50 m at varying speeds both on a street and on vegetation. Parameter optimization was done using grid search and 5-fold cross validation.

In our experiments, we achieved a high classification accuracy and thus this data is well suited to label the training data for the laser-based classification.

B. Mapping of Vegetation

We use multi-level surface maps [18] to model the environment. This representation uses a 2D grid and stores in each cell a set of patches representing individual surfaces at different heights. In our approach, we additionally store the probability of each surface patch to contain vegetation. Let $P(\mathrm{v}^i)$ denote this probability of patch i. In general, surface patches will be observed multiple times. Therefore, we need to probabilistically combine results from several measurements. In this way, the uncertainty in classification is explicitly taken into account.

Let z_t be a laser measurement at time t. Analogous to Moravec [12], we obtain an update rule for $P(\mathrm{v}^i \mid z_{1:t})$. First we apply Bayes' rule and obtain

$$P(\mathrm{v}^i \mid z_{1:t}) = \frac{P(z_t \mid \mathrm{v}^i, z_{1:t-1}) P(\mathrm{v}^i \mid z_{1:t-1})}{P(z_t \mid z_{1:t-1})}. \quad (7)$$

We then compute the ratio

$$\frac{P(\mathrm{v}^i \mid z_{1:t})}{P(\neg\mathrm{v}^i \mid z_{1:t})} = \frac{P(z_t \mid \mathrm{v}^i, z_{1:t-1})}{P(z_t \mid \neg\mathrm{v}^i, z_{1:t-1})} \frac{P(\mathrm{v}^i \mid z_{1:t-1})}{P(\neg\mathrm{v}^i \mid z_{1:t-1})}. \quad (8)$$

Similarly, we obtain

$$\frac{P(\mathrm{v}^i \mid z_t)}{P(\neg\mathrm{v}^i \mid z_t)} = \frac{P(z_t \mid \mathrm{v}^i)}{P(z_t \mid \neg\mathrm{v}^i)} \frac{P(\mathrm{v}^i)}{P(\neg\mathrm{v}^i)},$$

which can be transformed to

$$\frac{P(z_t \mid \mathrm{v}^i)}{P(z_t \mid \neg\mathrm{v}^i)} = \frac{P(\mathrm{v}^i \mid z_t)}{P(\neg\mathrm{v}^i \mid z_t)} \frac{P(\neg\mathrm{v}^i)}{P(\mathrm{v}^i)}. \quad (9)$$

If we apply the Markov assumption that the current observation is independent of previous observations given we know that a patch contains vegetation

$$P(z_t \mid \mathrm{v}^i, z_{1:t-1}) = P(z_t \mid \mathrm{v}^i) \quad (10)$$

and utilize the fact that $P(\neg\mathrm{v}^i) = 1 - P(\mathrm{v}^i)$, we obtain

$$\frac{P(\mathrm{v}^i \mid z_{1:t})}{1 - P(\mathrm{v}^i \mid z_{1:t})} = \frac{P(\mathrm{v}^i \mid z_t)}{1 - P(\mathrm{v}^i \mid z_t)} \frac{P(\mathrm{v}^i \mid z_{1:t-1})}{1 - P(\mathrm{v}^i \mid z_{1:t-1})} \frac{1 - P(\mathrm{v}^i)}{P(\mathrm{v}^i)}. \quad (11)$$

This equation can be transformed into the following update formula:

$$P(\mathrm{v}^i \mid z_{1:t}) = \left[1 + \frac{1 - P(\mathrm{v}^i \mid z_t)}{P(\mathrm{v}^i \mid z_t)} \frac{1 - P(\mathrm{v}^i \mid z_{1:t-1})}{P(\mathrm{v}^i \mid z_{1:t-1})} \frac{P(\mathrm{v}^i)}{1 - P(\mathrm{v}^i)}\right]^{-1} \quad (12)$$

To perform the update step, we need an inverse sensor model $P(\mathrm{v}^i \mid z)$. In our system, this sensor model is based on the remission value of the laser. The SVM-classifier is used to model the measurement probabilities as described in the next section. Here, the prior probability of $P(\mathrm{v}^i)$ was set to 0.5.

Fig. 4. Robots used in our experiments.

C. Class Probabilities

The decision function of Support Vector Machines (see Eq. 3) produces a class label y which is not a probability (y is either 1 or -1).

Several methods have been proposed to map the output of SVMs to posterior class probabilities [14], [19]. Among those a popular approach is Platt's method [14]. It uses a parametric model to fit the posterior given by

$$P(y = 1 \mid f) = (1 + \exp(Af + B))^{-1}. \quad (13)$$

The parameters A and B are determined using maximum likelihood estimation from a training set (f_i, y_i), where f_i are the outputs of the function given in Eq. 2 and y_i are the corresponding class labels. For details of the optimization step see [14]. In our approach, this method is used to obtain the inverse sensor model $P(\mathrm{v}^i \mid z)$.

VI. Experiments

Our approach has been implemented and evaluated in several experiments. The experiments are designed to demonstrate that our approach is suitable to allow robots to reliably detect vegetation and thus improves robust navigation in structured outdoor environments.

We used two different robot systems (see Fig. 4). The self-supervised learning approach is evaluated using an ActivMedia Pioneer 2 AT, which is able to traverse low vegetation. For mapping large environments and for an autonomous driving experiment, we use an ActivMedia Powerbot platform. This robot cannot safely traverse grass since its castor wheels will block the robot due to its weight. Both robots are equipped with SICK LMS S291-S05 laser scanners on pan-tilt units. In addition to that, the Pioneer robot carries an XSens MTi IMU to measure vibrations. Three-dimensional scans are gathered by tilting the laser scanner from 50 degrees upwards to 30 degrees downwards. We use the raw (unnormalized) remission values provided by the scanner.

We limit the classification to scans with a range smaller than 5.0 meters. The approach itself is not limited in range. However, at a laser resolution of one degree and a low height of the sensor relative to the ground (approx. 0.5 m), long range data will be too sparse both to gather training data and to reliably detect drivable surfaces and obstacles.

A. Vegetation detection using range differences

In a first experiment, we implemented a vegetation detection algorithm based purely on the range differences of

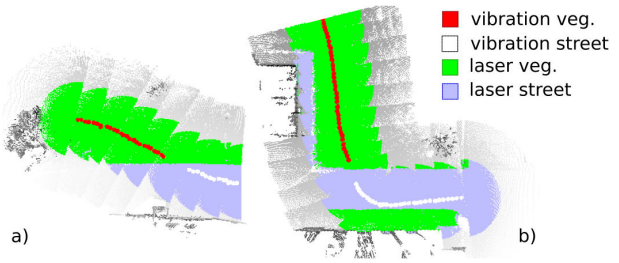


Fig. 5. Self-supervised learning. Left: the data set which is used to train the classifier. Right: test set recorded at a different location.

neighboring laser measurements similar to the one proposed by Wolf et al. [23]. This method is only used for a comparison with our proposed classification approach. Three-dimensional data is acquired by gathering a sweep of 2D scans. For each 2D scan the 2D endpoints $p_i = \langle x_i, y_i \rangle$ are computed from the range and angle measurements $\langle r_i, \alpha_i \rangle$. A local feature d_i is then determined for every scan point

$$d_i \quad = \quad x_i - x_{i-1}. \tag{14}$$

This feature captures the local roughness of the terrain. To cope with flat but tilted surfaces we classify scans based on the absolute difference in d_i between neighboring range measurements as suggested in [11]. Furthermore, we also use the measured range as a training feature to account for the varying data density from near to far scans. A SVM is used to train a classifier based on these features.

In our experiments, we achieve a classification accuracy of about 75% using the described method. An example of the classification results can be seen in Fig. 1. Similar results have been reported by other researchers [2].

B. Self-supervised Learning

To train our remission-based classifier, we manually steered the Pioneer AT robot through an outdoor environment consisting of a street and an area covered with grass. We acquired 3D scans approximately every 4 m. While the robot was driving the IMU measured the vibration induced to the robot. To correct odometry errors of the robot, we employed a state-of-the-art SLAM approach [17] and 3D scan-matching. We trained our classifier using the self-supervised approach described in Sec. V. The training set is visualized in Fig. 5a. The data recorded at the border region between street and vegetation were ignored since the precise location of the border cannot be determined using the vibration sensor. The model for the laser-based classifier was trained using 19,989 vegetation and 11,248 street samples.

To further evaluate the precision of the classifier, we recorded separate test data at a different location (see Fig. 5b and Fig. 6). The test set contains 36,304 vegetation and 28,883 street measurements. Again, the labeling of the data was carried out using the vibration-based classifier. The previously trained classifier reached a precision of 99.9 % on the test data; the recall is 99.6 %. The confusion matrix is given in Table I. Note that such accurate classification results are not due to overfitting. Fig. 2 illustrates that a non-linear function (as learned by the SVM in our approach) can clearly separate the classes.

TABLE I
CONFUSION MATRIX FOR EXP. VI-B (NUMBER OF DATA POINTS)

	vegetation	street
vegetation	36,300	138
street	4	28,745

Fig. 6. Aerial view of the computer science campus in Freiburg. Approximated robot trajectories are shown for the training (top, yellow) and the test set (bottom, red). Courtesy of Google Maps, Copyright 2008, DigitalGlobe.

C. 3D Mapping

In this experiment, the Powerbot robot was steered across the computer science campus at the University of Freiburg. The scanning laser of the robot was tilted to a fixed angle of 20 degrees downwards. In this way, a fairly large area could be mapped in less than 15 minutes. The length of the trajectory is 490 m. The vegetation classifier was used to map vegetation in the three-dimensional model of the environment. To properly integrate multiple measurements, we used the mapping approach described in Sec. V-B with a cell size of 0.1 m.

Due to a significantly different hardware setup than on the Pioneer AT, we were not able to use the model generated in Sec. VI-B. Instead, we recorded a training set of 12,153 grass and 10,448 street samples by placing the robot in front of flat areas containing only street and only vegetation. This method is only applicable if such example data can be gathered and thus should be considered inferior to the self-supervised approach described in Sec. V.

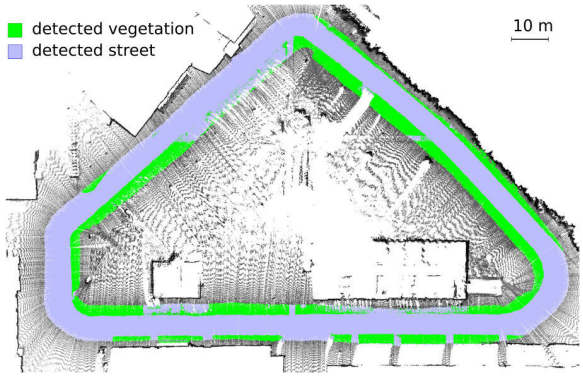


Fig. 7. Mapping of a large outdoor environment. The laser was tilted at a fixed angle of 20 degrees while the robot was moving. The figure shows a 2D projection of the 3D map.

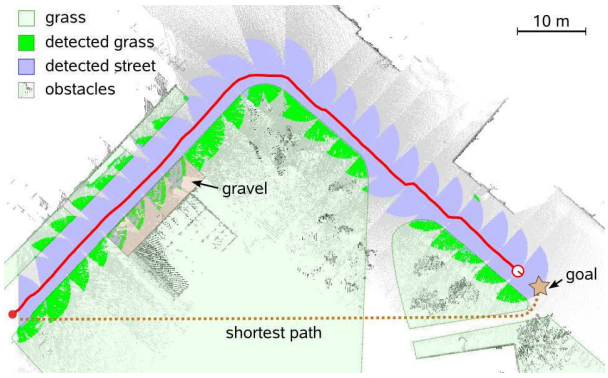


Fig. 8. Autonomous navigation experiment. Although the shortest obstacle-free path from the start to the goal position led over grass, the robot could reliably avoid the vegetated areas by using our vegetation classifier and traveled over the paved streets to reach its goal.

Compared to the aerial image of the campus site in Fig. 6, the mapping result shown in Fig. 7 is highly accurate. Even small amounts of vegetation, for example between tiles on a path, can be identified. To evaluate the accuracy of the map created during this experiment, we manually marked wrongly classified cells. Of a total of 271,638 cells (75,622 vegetation, 196,016 street), we found 547 false positives and 194 false negatives. This corresponds to a precision of 99.23 % and a recall of 99.74 %.

D. Autonomous Navigation

As mentioned above, the Powerbot cannot safely traverse vegetated areas. In the experiment depicted in Fig. 8 (see also our video attachment), the robot was told to navigate to a goal position 80 m in front while avoiding vegetation. Since the robot did not have a map, it explored the environment in the process of reducing its distance to the goal location. Thereby, it used the vegetation classifier to detect vegetation. The environment was represented as described in Sec. V-B. Without knowledge about the specific terrain, the shortest obstacle-free path would have led the robot across a large area containing grass. By considering the classification results in the path costs, however, the planner chose a safe trajectory over the street.

VII. Conclusion

In this paper, we proposed a new approach to vegetation detection using the remission values of a laser scanner. By predicting vegetation in the surrounding of a robot, our approach improves robot navigation in structured outdoor environments. The laser classifier is learned in a self-supervised fashion by means of a support vector machine using a vibration-based terrain classifier to gather training data. The approach has been implemented and evaluated in several real-world experiments. The experiments show that our approach is able to accurately detect low, grass-like vegetation with an accuracy of more than 99%. We also demonstrated that the terrain classification can be used to improve the navigation behavior of a robot.

Our current approach is limited to detecting flat vegetation due to the self-supervised training method. In future work, we will investigate whether the described approach can also be applied to classify tall vegetation such as trees or bushes. We will also look into using remission values provided by the recently introduced Hokuyo UTM-30LX. With a weight of 370 g this sensor could allow vegetation detection on an even broader range of robots including humanoids and small scale robots.

References

[1] R. Baribeau, M. Rioux, and G. Godin. Color reflectance modeling using a polychromatic laser range sensor. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):263–269, 1992.

[2] D. Bradley, R. Unnikrishnan, and J. Bagnell. Vegetation detection for driving in complex environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[3] C.A. Brooks, K. Iagnemma, and S. Dubowsky. Vibration-based terrain analysis for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3415–3420, 2005.

[4] C-C. Chang and C-J. Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/čjlin/libsvm, 2001.

[5] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, Philadelphia, USA, 2006.

[6] B. Douillard, D. Fox, and F. Ramos. Laser and vision based outdoor object mapping. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, 2008.

[7] E.M. DuPont, R.G. Roberts, C.A. Moore, M.F. Selekwa, and E.G. Collins. Online terrain classification for mobile robots. In *Proc. of the Int. Mechanical Engineering Congress and Exposition Conference (IMECE)*, Orlando, USA, 2005.

[8] M. Hebert and N. Vandapel. Terrain classification techniques from ladar data for autonomous navigation. In *Proc. of the Collaborative Technology Alliances conference*, College Park, MD., 2003.

[9] J-F. Lalonde, N. Vandapel, D. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839 – 861, 2006.

[10] J. Macedo, R. Manduchi, and L. Matthies. Ladar-based discrimination of grass from obstacles for autonomous navigation. In *ISER 2000: Experimental Robotics VII*, London, UK, 2001.

[11] R. Manduchi, A. Castano, A. Talukder, and L.Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots, 18*, pages 81–102, 2003.

[12] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, 1988.

[13] R.B. Myneni, F.G. Hall, P.J. Sellers, and A.L. Marshak. The interpretation of spectral vegetation indexes. *IEEE Transactions on Geoscience and Remote Sensing*, 33(2):481–486, 1995.

[14] J.C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[15] D. Sadhukhan, C. Moore, and E. Collins. Terrain estimation using internal sensors. In *Proc. of the IASTED Int. Conf. on Robotics and Applications*, Honolulu, Hawaii, USA, 2004.

[16] B Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[17] C. Stachniss and G. Grisetti. GMapping project at OpenSLAM.org. http://openslam.org, 2007.

[18] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[19] V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[20] C. Weiss, N. Fechner, M. Stark, and A. Zell. Comparison of different approaches to vibration-based terrain classification. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

[21] C. Weiss, H. Frohlich, and A. Zell. Vibration-based terrain classification using support vector machines. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[22] C. Wellington, A. Courville, and A. Stentz. A generative model of terrain for autonomous navigation in vegetation. *International Journal of Robotics Research*, 25(12):1287 – 1304, 2006.

[23] D.F. Wolf, G. Sukhatme, D. Fox, and W. Burgard. Autonomous terrain mapping and classification using hidden markov models. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[C4] W. Burgard, B. Steder, R. Kuemmerle, M. Ruhnke, G. Grisetti, C. Stachniss, C. Dornhege, A. Kleiner, and Juan D. Tardos. How to compare the results of slam algorithms. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009.

A Comparison of SLAM Algorithms Based on a Graph of Relations

Wolfram Burgard Cyrill Stachniss Giorgio Grisetti Bastian Steder
Rainer Kümmerle Christian Dornhege Michael Ruhnke Alexander Kleiner Juan D. Tardós

***Abstract*— In this paper, we address the problem of creating an objective benchmark for comparing SLAM approaches. We propose a framework for analyzing the results of SLAM approaches based on a metric for measuring the error of the corrected trajectory. The metric uses only relative relations between poses and does not rely on a global reference frame. The idea is related to graph-based SLAM approaches in the sense that it considers the energy needed to deform the trajectory estimated by a SLAM approach to the ground truth trajectory. Our method enables us to compare SLAM approaches that use different estimation techniques or different sensor modalities since all computations are made based on the corrected trajectory of the robot. We provide sets of relative relations needed to compute our metric for an extensive set of datasets frequently used in the SLAM community. The relations have been obtained by manually matching laser-range observations. We believe that our benchmarking framework allows the user an easy analysis and objective comparisons between different SLAM approaches.**

I. Introduction

Models of the environment are needed for a wide range of robotic applications including transportation tasks, guidance, and search and rescue. Learning maps has therefore been a major research focus in the robotics community over the last decades. In the literature, the mobile robot mapping problem under pose uncertainty is often referred to as the *simultaneous localization and mapping* (SLAM) or *concurrent mapping and localization* (CML) problem [26]. SLAM is considered to be a complex problem because to localize itself a robot needs a consistent map and for acquiring the map the robot requires a good estimate of its location.

Whereas dozens of different techniques to tackle the SLAM problem have been proposed, there is no gold standard for comparing the results of different SLAM algorithms. In the community of feature-based estimation techniques, researchers often measure the Euclidean or Mahalanobis distance between the estimated landmark location and the true location (if this information is available). As we will illustrate in this paper, comparing results based on an absolute reference frame can have shortcomings. In the area of grid-based estimation techniques, people often use visual inspection to compare maps or overlays with blueprints of buildings. This kind of evaluation becomes more and more difficult as new SLAM approaches show increasing capabilities and thus large scale environments are needed for evaluation. Therefore, there is a strong need for methods allowing meaningful comparisons of different approaches. Ideally, such a method is capable of performing comparisons between mapping systems that apply different estimation techniques and operate on different sensing modalities. We argue that meaningful comparisons between different SLAM approaches require to have a common performance metric. This metric should allow to compare the outcome of different mapping approaches when applying them on the same dataset.

In this paper, we propose a novel technique for comparing the output of SLAM algorithms. It is based on an idea that is actually similar to the concept of the graph-based SLAM approaches [19], [12], [22]. It uses the energy that is (virtually) needed to deform the trajectory estimated by a SLAM approach into the ground truth trajectory as a quality measure.

Our metric that operates only on relative geometric relations between poses along the trajectory of the robot. This is inspired by the fact used in most Rao-Blackwellized particle filter approaches, namely that estimating the map becomes trivial given the robot's trajectory [10], [20], [25]. Our metric enables a user to establish a benchmark for objectively comparing the performance of a mapping system to existing approaches. Our approach allows for making (approximative) comparisons between algorithms even if a perfect ground truth information is not available. This enables us to present benchmarks based on frequently used datasets in the robotics community such as the MIT Killian Court, or the Intel Research Lab dataset. The disadvantage of our method is that it requires manual work to be carried out by a human that knows the topology of the environment. The manual work, however, has to be done only once for a dataset and then allows other researchers to evaluate their methods with low effort. Together with this paper, we provide a web page that hosts such manually matched relations for existing log files [17]. We furthermore provide evaluations for the results of three different mapping techniques, namely scan-matching, SLAM using Rao-Blackwellized particle filter [10], and a maximum likelihood SLAM approach based on the graph formulation [11], [21].

II. Related Work

Learning maps is a frequently studied problem in the robotics literature. SLAM techniques for mobile robots can

All authors are with the University of Freiburg, Dept. of Computer Science, Georges Koehler Allee 79, 79110 Freiburg, Germany except of Juan D. Tardós who is with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, María de Luna 1, E-50018, Zaragoza, Spain.

The authors gratefully thank Mike Bosse, Patrick Beeson, and Dirk Haehnel for providing the MIT Killian Court, the ACES, and the Intel Research Lab datasets. This work has partly been supported by the DFG under contract number SFB/TR-8 and the European Commission under contract numbers FP6-2005-IST-6-RAWSEEDS, FP6-IST-045388-INDIGO, and FP7-231888-EUROPA.

be classified according to the underlying estimation technique. The most popular approaches are extended Kalman filters (EKFs) [18], [23] and its variants [15], sparse extended information filters [8], [28], particle filters [20], [10], least square error minimization approaches [19], [12], [22] and several others including also techniques for learning local maps only [13], [27], [30].

The approach of finding maximum likelihood maps using a graph or network of constraints is strongly related to our approach for evaluating SLAM methods presented in this paper. Lu and Milios [19] introduced the concept of graph-based or network-based SLAM using a kind of brute force method for optimization. Gutmann and Konolige [12] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Olson *et al.* [22] presented an optimization approach that applies stochastic gradient descent for resolving relations in a network efficiently.

Activities related to performance metrics for SLAM methods, as the work described in this paper, can roughly be divided into three major categories: First, competitions where robot systems are competing within a defined problem scenario (such as playing soccer), second, collections of publicly available datasets that are provided for comparing algorithms on specific problem, and third, related publications that introduce methodologies and scoring metrics for comparing different methods.

To perform comparisons between robots, numerous robot competitions have been initiated in the past, evaluating the performance of cleaning robots [6], robots in simulated Mars environments at the ESA Lunar Robot Challenge[7], robots playing soccer or rescuing victims after a disaster at RoboCup, and cars driving autonomously at the DARPA Urban Challenge. However, competition settings are likely to generate additional noise due to differing hardware and software settings. Depending on the competition, approaches are often tuned to the settings addressed in the competitions.

In the robotics community, there exist some well-known web sites providing datasets such as Radish [14] or [3] and algorithms [24] for mapping. However, they neither provide ground truth data nor recommendations on how to compare different maps in a meaningful way. Whereas Zivkovic *e*t al. [31] provide a labeled dataset containing information useful for human-robot interaction, Frese [9] generated a dataset of the DLR building with manually obtained ground truth data associations.

Some steps towards benchmarking navigation solutions have been presented in the past. Amigoni *e*t al. [1] presented a general methodology for performing experimental activities in the area of robotic mapping. They suggested a number of issues that should be addressed when experimentally validating a mapping method. If ground truth data is available, they suggest to utilize the Hausdorff metric for map comparison.

Wolf *e*t al. [29] proposed the idea of using manually supervised Monte Carlo Localization (MCL) for matching 3D scans against a reference map. They suggested to generate the reference maps from independently created CAD data, which can be obtained from the land registry office. The comparison between the generated map and the ground truth has been carried out by computing the Euclidean distance and angle difference of each scan, and plotting these over time. We argue here that comparing the absolute error between two tracks might not yield a meaningful assertion.

Balaguer *e*t al. [2] utilize the USARSim robot simulator and a real robot platform for comparing different open source SLAM approaches and they propose that the simulator engine could be used for systematically benchmarking different approaches of SLAM. However, it has also been shown that noise is often but not always Gaussian in the SLAM context [25]. Gaussian noise, however, is typically used in most simulation systems. In addition to that, Balaguer *et al.* do not provide a quantitative metric for comparing generated maps with ground truth. As many other approaches, their comparisons were carried out by visual inspection.

III. Metric for Benchmarking SLAM Algorithms

We propose a metric for measuring the performance of a SLAM algorithm not by comparing the map itself but by considering the poses of the robot during data acquisition. In this way, we gain two important properties: First, it allows us to compare the result of algorithms that generate different types of metric map representations, such as feature-maps or occupancy grid maps. Second, the method is invariant to the sensor setup of the robot. Thus, a result of a graph-based SLAM approach working on laser range data can be compared, for example, with the result of vision-based FastSLAM. The only property we require is that the SLAM algorithm estimates the trajectory of the robot given by a set of poses. All benchmark computations will be performed on this set.

A. The Metric

Let $x_{1:T}$ be the poses of the robot estimated by a SLAM algorithm from time step 1 to T, $x_t \in SE(2)$ or $SE(3)$. Let $x^*_{1:T}$ be the reference poses of the robot during mapping, ideally the true poses. A straightforward error metric could be defined as

$$\varepsilon(x_{1:T}) \quad = \quad \sum_{t=1}^{T} (x_t \ominus x_t^*)^2, \tag{1}$$

where $\oplus$ is the standard motion composition operator and $\ominus$ its inverse as defined by Lu and Milios [19] or its analogous definition for $SE(3)$, respectively. Thus, $\delta_{i,j} = x_j \ominus x_i$ is the relative transformation that moves the node x_i onto x_j. Let $\delta^*_{i,j}$ be the transformation based on x^*_i and x^*_j accordingly. Eq. 1 can be rewritten as

$$\begin{aligned} \varepsilon(x_{1:T}) \quad = \quad & \sum_{t=1}^{T} \big((x_1 \oplus \delta_{1,2} \oplus \ldots \oplus \delta_{t-1,t}) \\ & \ominus (x_1^* \oplus \delta^*_{1,2} \oplus \ldots \oplus \delta^*_{t-1,t}) \big)^2. \end{aligned} \tag{2}$$

We claim that this metric is suboptimal for comparing the result of a SLAM algorithm. To illustrate this, consider Figure 1. Here, a robot travels along a straight line. Let the robot make perfect pose estimates in general but a rotational

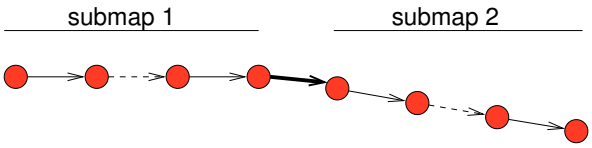


Fig. 1. This figure illustrates a simple example where the metric in Eq. 1 is suboptimal. The robot moves along a straight line and after n poses, it makes a small angular error (bold arrow) but then continues without any further error. Both parts (labeled submap 1 and submap 2) are perfectly mapped and only the connection between both submaps contains an error. According to Eq. 1, the error of this estimates increases with every node added to submap 2 although the submap itself is perfectly estimated. Thus, the error depends on the point in time where the robot made an estimation error without considering that it might not introduce any (further) error.

error somewhere along the line, let us say in the middle. Both submaps (before and after the estimation error) are perfectly mapped. According to Eq. 1, the error of this estimate increases with every node that is added to submap 2 although the submap itself is perfectly estimated. Thus, the error depends on the point in time where the robot made an estimation error without considering that it might not introduce any (further) error. The reason for this is the fact that the metric in Eq. 1 operates on global coordinates and considers the trajectory and thus the map as a rigid body that has to be aligned with the ground truth.

In this paper, we propose to use a metric that considers the deformation energy that is needed to transfer the estimate into the ground truth. This can be done – similar to the ideas of the graph mapping introduced by Lu and Milios [19] – by considering the poses as masses and connections between them as springs. Thus, our metric is based on the *relative* displacement between poses. Instead of comparing x to x^* (in the global reference frame), we do the operation based on δ and δ^* as

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j}(\delta_{i,j} \ominus \delta^*_{i,j})^2 \quad (3)$$

$$= \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \quad (4)$$

where N is the number of relative relations and $trans(\cdot)$ and $rot(\cdot)$ are used to separate the translational and rotational components. We suggest to provide both quantities individually. In this case, the error (or transformation energy) in the above-mentioned example will be consistently estimated as the single rotational error no matter where the error occurs in the space or in which order the data is processed. Note that this score is a metric since it satisfies the four properties of non-negativity, identity of indiscernibles, symmetry, and triangle inequality.

Our error metric, however, leaves open which relative displacements $\delta_{i,j}$ are included in the summation in Eq. 4. Evaluating two approaches based on a different set of relative pose displacements will obviously result in two different scores. As we will show in the remainder of this section, the set δ (and thus δ^*) can be defined to highlight certain properties of an algorithm.

Note that some researchers prefer the absolute error (absolute value, not squared) instead of the squared one. We prefer the squared one since it comes from the motivation that the metric measures the energy needed to transform the estimated trajectory into ground truth. However, one can also use the metric based on the non-squared error instead of the squared one. In the experimental evaluation, we actually provide both values.

It should be noted that the metric presented here also has drawbacks. First, the metric, as we defined it, only evaluates the mean estimate of the SLAM algorithm and does not consider its estimate of the uncertainty. Second, it misses a probabilistic interpretation as the Fisher information would realize, see, for example, Censi's work [4] on the achievable accuracy for range finder-based localization.

B. Selecting Relative Displacements for Evaluation

Benchmarks are designed to compare different algorithms. In the case of SLAM systems, however, the task the robot finally has to solve should define the required accuracy and this information should be considered in the benchmark.

For example, a robot generating blueprints of buildings should reflect the geometry of a building as accurately as possible. In contrast to that, a robot performing navigation tasks requires a map that can be used to robustly localize itself and to compute valid trajectories to a goal location. To carry out this task, it is sufficient in most cases, that the map is topologically consistent and that its observations can be locally matched to the map. A map having these properties is often referred to as locally consistent.

By selecting the relative displacements $\delta_{i,j}$ used in Eq. 4 for a given dataset, the user can highlight certain properties and thus design a benchmark for evaluating an approach given the application in mind.

For example, by adding only known relative displacements between nearby poses based on visibility, a local consistency is highlighted. In contrast to that, by adding known relative displacements of far away poses, for example, provided by an accurate external measurement device or by background knowledge, the accuracy of the overall geometry of the mapped environment is enforced. In this way, one can incorporate the knowledge into the benchmark that, for example, a corridor has a certain length and is straight.

IV. Obtaining Reference Relations

In practice, the key question regarding Eq. 4 is how to determine the *true relative displacements* between poses. Obviously, the true values are available only in simulation. If ground truth information is available, it is trivial to derive the exact relations. However, we can also determine close-to-true values by using the information recorded by the mobile robot and the background knowledge of the human recording the datasets. This, of course, involves manual work, but from our perspective it is the best method for obtaining such relations if no ground truth is available.

Please note, that the metric proposed above is independent of the actual sensor used. In the remainder of this paper, however, we will concentrate on laser range finders which are highly popular sensors in robotics at the moment. To evaluate an approach operating on a different sensor modality, one has

two possibilities. Either one temporarily mounts a laser range finder on the robot (if this is possible) or has to provide a method for accurately determining the relative displacement between two poses from which an observation has been taken that observes the same part of the space.

A. Initial Guess

In our work, we propose the following strategy. First, one seeks for an initial guess about the relative displacement between poses. Based on the knowledge of the human, a wrong initial guess can be easily discarded since the human "knows" the structure of the environment. In a second step, a refinement is proposed based on manual interaction.

In most cases, researchers in robotics will have SLAM algorithms at hand that can be used to compute such an *initial guess*. By manually inspecting the estimates of the algorithm, a human can accept or discard a match. It is important to note that the output is not more than an initial guess and it is used to estimate the visibility constraints which will be used in the next step.

B. Manual Matching Refinement and Scan Rejection

Based on the initial guess about the pose of the robot for a given time step, it is possible to determine which observations in the dataset should have covered the same part of the space or the same objects. For a laser range finder, this can easily be achieved. Between each visible pair of poses, one adds a relative displacement into a candidate set.

In the next step, a human processes the candidate set to eliminate wrong hypotheses by visualizing the observation in a common reference frame. This requires manual interaction but allows for eliminating wrong matches and outliers with high precision. Since we aim to find the best possible relative displacement, we perform a pair-wise registration procedure to refine the estimates of the observation registration method. It furthermore allows the user to manually adjust the relative offset between poses so that the pairs of observations fit perfectly. Alternatively, the pair can be discarded.

This approach might sound work-intensive but with an appropriate user interface, this task can be carried out within a reasonable amount of time. For example, for a standard dataset with 1700 relations, it took an unexperienced user approximately four hours to extract the relative translations needed to serve as the input to the error calculation.

C. Adding Additional Relations

In addition to the relative transformations added upon visibility and matching of observations, one can directly incorporate additional relations resulting from other sources of information, for example, given the knowledge about the length of a corridor in an environment. By adding a relation between two poses – each at one side of the corridor – one can easily incorporate knowledge about the global geometry of an environment if available. An alternative approach, for example, is to use aerial imagery [16].

V. Benchmarking of Algorithms without Trajectory Estimates

A series of SLAM approaches estimate the trajectory of the robot as well as a map. However, in the context of the EKF, researchers often exclude an estimate of the full trajectory to lower the computational load.

We see two solutions to overcome this problem: First, depending on the capabilities of the sensor, one can recover the trajectory as a post processing step given the feature locations and the data association estimated by the approach. This procedure could be quite easily realized by a localization run in the built map with given data association (the data association of the SLAM algorithm). Second, in some settings this strategy can be difficult and one might argue that a comparison based on the landmark locations is more desirable. In this case, one can apply our metric as well by operating on the landmark locations instead of on the poses of the robot. The relations $\delta^*_{i,j}$ can then be determined by measuring the relative distances between landmarks using, for example, a highly accurate measurement device and a triangulation based on the landmark location.

The disadvantage of this approach is that the data association between estimated landmarks and ground truth landmarks is not given. Depending on the kind of observations, a human can manually determine the data association for each observation of an evaluation dataset as done by Frese [9]. This, however, might get intractable for SIFT-like features obtained with high frame rate cameras. Note that all metrics measuring an error based on landmark locations require such a data association as given. Furthermore, it becomes impossible to compare significantly different SLAM systems using different sensing modalities. Therefore, we recommend the first option to evaluate techniques such as the EKF.

VI. Datasets for Benchmarking

To validate the metric, we selected a set of datasets representative for different kinds of environments from the publicly available datasets. We extracted relative relations between robot poses using the methods described in the previous sections by manually validating every single observation between pairs of poses.

As a challenging indoor corridor-environment with a non-trivial topology including nested loops, we selected the MIT Killian Court dataset and the dataset of the ACES building at the University of Texas, Austin. As a typical office environment with a significant level of clutter, we selected the dataset of building 079 at the University of Freiburg, the Intel Research Lab dataset, and a dataset acquired at the CSAIL at MIT. To give a visual impression of the corresponding environments, Figure 2 illustrates maps obtained by executing state-of-the-art SLAM algorithms. All datasets, the manually verified relations, and map images are available online [17].

VII. Experimental Evaluation

This evaluation is designed to illustrate the properties of our method. We selected three popular mapping techniques

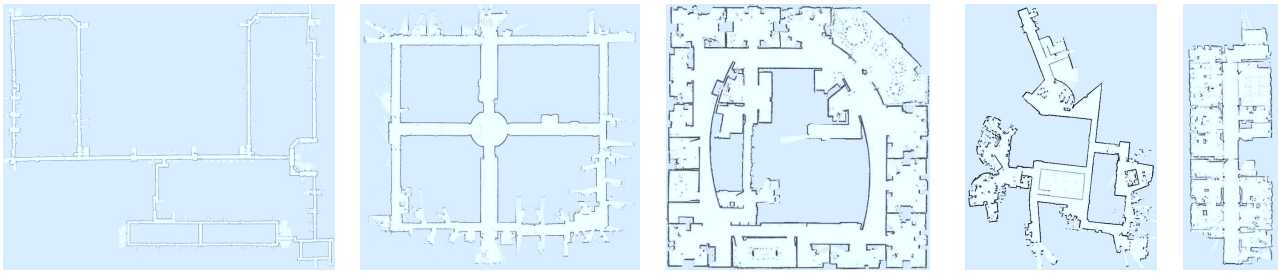


Fig. 2. Maps obtained by the reference datasets used to validate our metric. From left to right: MIT Killian Court, ACES Building at the University of Texas, Intel Research Lab Seattle, MIT CSAIL Building, and building 079 University of Freiburg.

and processed the datasets discussed in the previous section. We provide the obtained scores from the metric for all combinations of SLAM approach and dataset. This will allow other researchers to compare their own SLAM approaches against our methods using the provided benchmark datasets.

A. Evaluation of Existing Approaches

In this evaluation, we considered the following mapping approaches and present results obtained with these techniques using the datasets briefly described in the previous section.

First, we applied incremental scan matching as a kind of baseline approach. Scan matching, here using the approach of Censi [5], incrementally computes an open loop maximum likelihood trajectory of the robot by matching consecutive scans.

Second, we used GMapping which is a mapping system based on a Rao-Blackwellized Particle Filter (RBPF) for learning grid maps. We used the RBPF implementation described in [10] and available online [24]. It estimates the posterior over maps and trajectories by means of a particle filter. Each particle carries its own map and a hypothesis of the robot pose within that map.

Third, we selected an approach that addresses the SLAM problem by graph optimization. The idea is to construct a graph out of the sequence of measurements. Every node of the graph is labeled with a robot pose and the measurement taken at that pose. Then, a least square error minimization approach is applied to obtain the most likely configuration of the graph. The approach described by Olson [21] is used to determine constraints and the optimizer TORO available online [24] and described in [11] is applied.

For our evaluation, we manually extracted the relations for all datasets mentioned in the previous section (the data is available online). We then carried out the mapping approaches and used the corrected trajectory to compute the error according to our metric. Note, that the error computed according to our metric (as well as for most other metrics too) can be separated into two components: a translational error and a rotational error. Often, a "weighting-factor" is used to combine both error terms into a single number. In our evaluation here, however, we provide both terms separately for a better transparency of the results.

We processed all benchmark datasets mentioned in Section VI using the algorithms listed above. A condensed view of each algorithm's performance is given by the averaged

TABLE I

QUANTITATIVE RESULTS OF DIFFERENT APPROACHES/DATASETS.

[1] SCAN MATCHING HAS BEEN APPLIED AS A PREPROCESSING STEP.

Trans. error m / m^2	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs)	0.173 ± 0.614	0.060 ± 0.049	0.044 ± 0.044
Aces (sqr)	0.407 ± 2.726	0.006 ± 0.011	0.004 ± 0.009
Intel (abs)	0.220 ± 0.296	0.070 ± 0.083	0.031 ± 0.026
Intel (sqr)	0.136 ± 0.277	0.011 ± 0.034	0.002 ± 0.004
MIT (abs)	1.651 ± 4.138	0.122 ± 0.386[1]	0.050 ± 0.056
MIT (sqr)	19.85 ± 59.84	0.164 ± 0.814[1]	0.006 ± 0.029
CSAIL (abs)	0.106 ± 0.325	0.049 ± 0.049[1]	0.004 ± 0.009
CSAIL (sqr)	0.117 ± 0.728	0.005 ± 0.013[1]	0.0001 ± 0.0005
FR 79 (abs)	0.258 ± 0.427	0.061 ± 0.044[1]	0.056 ± 0.042
FR 79 (sqr)	0.249 ± 0.687	0.006 ± 0.020 [1]	0.005 ± 0.011

Rot. error deg / deg^2	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs)	1.2 ± 1.5	1.2 ± 1.3	0.4 ± 0.4
Aces (swr)	3.7 ± 10.7	3.1 ± 7.0	0.3 ± 0.8
Intel (abs)	1.7 ± 4.8	3.0 ± 5.3	1.3 ± 4.7
Intel (sqr)	25.8 ± 170.9	36.7 ± 187.7	24.0 ± 166.1
MIT (abs)	2.3 ± 4.5	0.8 ± 0.8[1]	0.5 ± 0.5
MIT (sqr)	25.4 ± 65.0	0.9 ± 1.7[1]	0.9 ± 0.9
CSAIL (abs)	1.4 ± 4.5	0.6 ± 1.2[1]	0.05 ± 0.08
CSAIL (sqr)	22.3± 111.3	1.9 ± 17.3[1]	0.01 ± 0.04
FR 79 (abs)	1.7 ± 2.1	0.6 ± 0.6[1]	0.6 ± 0.6
FR 79 (sqr)	7.3 ± 14.5	0.7 ± 2.0[1]	0.7 ± 1.7

error over all relations. In Table I (top) we give an overview on the translational error of the various algorithms, while Table I (bottom) shows the rotational error. As expected, it can be seen that the more advanced algorithms (Rao-Blackwellized particle filter and graph mapping) usually outperform scan matching. This is mainly caused by the fact that scan matching only optimizes the result locally and will introduce topological errors in the maps, especially when large loops have to be closed. A distinction between RBPF and graph mapping seems difficult as both algorithms perform well in general. On average, graph mapping seems to be slightly better than a RBPF for mapping.

To visualize the results and to provide more insights about the metric, we do not provide the scores only but also plots showing the error of each relation. In case of high errors in a block of relations, we label the relations in the maps. This enables us to see not only where an algorithm fails, but might also provide insights why it fails. Inspecting those situations in correlation with the map helps to understand the properties of an algorithm and gives valuable insights on its capabilities. For two datasets, a detailed analysis using these

plots is presented in the following sections.

B. MIT Killian Court

In the MIT Killian Court dataset (also called the infinite corridor dataset), the robot mainly observed corridors with only few structures that support accurate pose correction. The robot traversed multiple nested loops – a challenge especially for the RBPF-based technique. We extracted close to 5,000 relations between nearby poses that are used for evaluation. Figure 3 shows three different results and the corresponding error distributions to illustrate the capabilities of our method. Regions in the map with high inconsistencies correspond to relations having a high error. The absence of significant structure along the corridors results in a small or medium re-localization error of the robot in all compared approaches. In sum, we can say the graph-based approach outperforms the other methods and that the score of our metric reflects the impression of a human about map quality obtained by visually inspecting the mapping results (the vertical corridors in the upper part are supposed to be parallel).

C. Freiburg Indoor Building 079

The building 079 of the University of Freiburg is an example for a typical office environment. The building consists of one corridor which connects the individual rooms. Figure 4 depicts the results of the individual algorithms (scan matching, RBPF, graph-based). In the first row of Figure 4, the relations having a translational error greater than 0.15 m are highlighted in dark blue.

In the left plot showing the scan matching result, the relations plotted in blue are generated when the robot revisits an already known region. These relations are visible in the corresponding error plots (Figure 4 first column, second and third row). As can be seen from the error plots, these relations with a number greater than 1,000 have a larger error than the rest of the dataset. The fact that the pose estimate of the robot is sub-optimal and that the error accumulates can also be seen by the rather blurry map and by double-occurrences of some walls. In contrast to that, the more sophisticated algorithms, namely RBPF and graph mapping, are able to produce consistent and accurate maps in this environment. Only very few relations show an increased error (illustrated by dark blue relations).

D. Summary of the Experiments

Our evaluation illustrates that the proposed metric provides a ranking of the results of mapping algorithms that is likely to be compatible with a ranking made by humans. Inconsistencies yield increased error scores since in the wrongly mapped areas the relations obtained from manual matching are not met. By visualizing the error of each constraint as done in the plots in this section, one can identify regions in which algorithms fail and we believe that this helps to understand where and why different approaches have problems to build accurate maps.

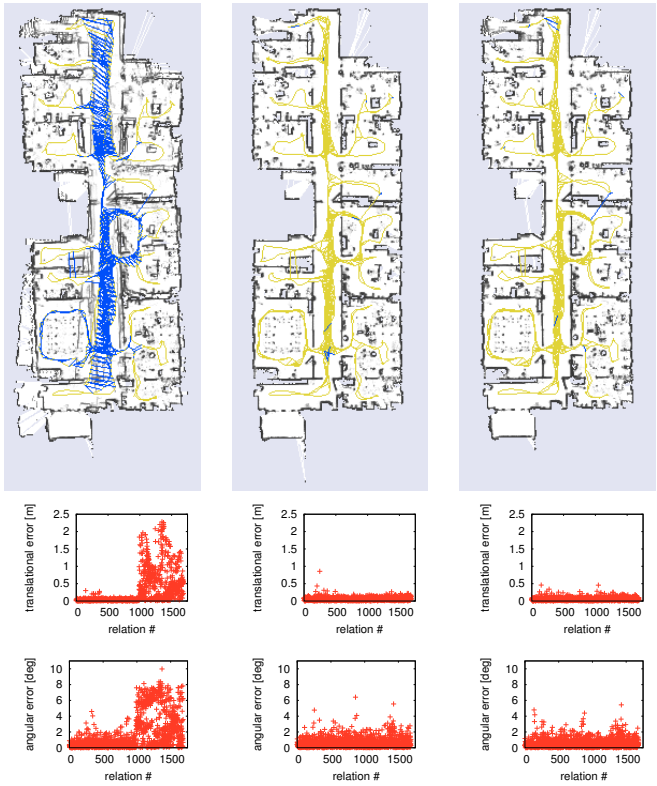


Fig. 4. This figure shows the Freiburg Indoor Building 079 dataset. Each column reports the results of one approach. Left: scan-matching, middle: RBPF and right a graph based algorithm. Within each column, the top image shows the map, the middle plot is the translational error and the bottom one is the rotational error.

VIII. Conclusion

In this paper, we presented a framework for comparing the results of SLAM approaches with the goal to create objective benchmarks. We proposed a metric for measuring the error of a SLAM system based on the corrected trajectory. Our metric uses only relative relations between poses and is motivated by the energy needed to transform an estimate into ground truth. This overcomes serious shortcomings of approaches using a global reference frame to compute the error. Our metric even allows the comparison of SLAM approaches that use different estimation techniques or different sensor modalities. In addition to the proposed metric, we provide robotic datasets together with relative relations between poses for benchmarking. These relations have been obtained by manually matching observations and yield a high matching accuracy. Finally, we provide an error analysis for three mapping systems using the metric and datasets. We believe that our results are a valuable benchmark for SLAM researchers since we provide a framework that allows for an objective and comparably easy analysis of the results of SLAM systems.

References

[1] F. Amigoni, S. Gasparini, and M. Gini. Good experimental methodologies for robotic mapping: A proposal. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[2] B. Balaguer, S. Carpin, and S. Balakirsky. Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems. In *IROS 2007 Workshop*, 2007.

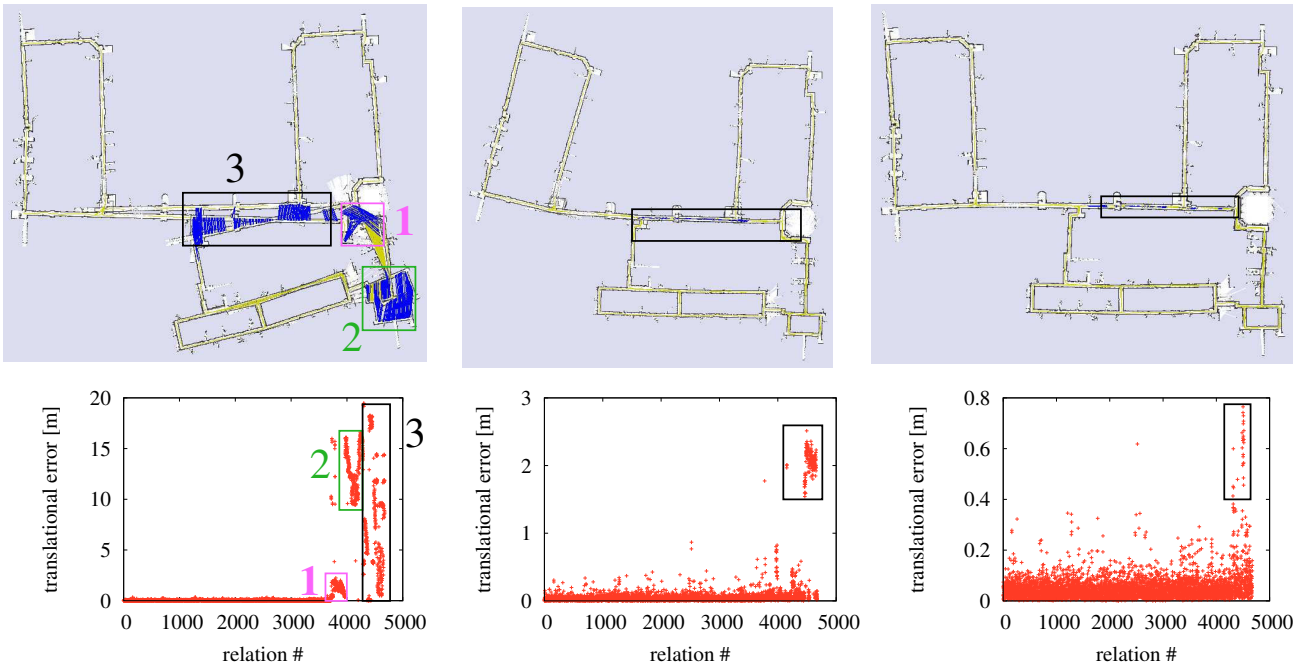


Fig. 3. The MIT Killian Court dataset. The reference relations are depicted in light yellow. The left column shows the results of scan-matching, the middle column the result of a GMapping using 50 samples, and the right column shows the result of a graph-based approach. The regions marked in the map (boxes and dark blue relations) correspond to regions in the error plots having high error. The rotational error is not plotted due to space reasons.

[3] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos. Rawseeds a project on SLAM benchmarking. In *Proc. of the IROS WS on Benchmarks in Robotics Research*, 2006.

[4] A. Censi. The achievable accuracy for range finder localization. *IEEE Transactions on Robotics*. Under review.

[5] A. Censi. Scan matching in a probabilistic framework. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2291–2296, 2006.

[6] EPFL and IROS. Cleaning Robot Contest, 2002. http://robotika.cz/competitions/cleaning2002/en.

[7] ESA. Lunar robotics challenge, 2008. http://www.esa.int/esaCP/SEM4GKRTKMF_index_0.html.

[8] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, 2005.

[9] U. Frese. Dlr spatial cognition data set. http://www.informatik.uni-bremen.de/agebv/en/DlrSpatialCognitionDataSet, 2008.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[11] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[12] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1999.

[13] J. Hermosillo, C. Pradalier, S. Sekhavat, C. Laugier, and G. Baille. Towards motion autonomy of a bi-steerable car: Experimental issues from map-building to trajectory execution. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[14] A. Howard and N. Roy. Radish: The robotics data set repository, standard data sets for the robotics community, 2003. http://radish.sourceforge.net/.

[15] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, 1995.

[16] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 2009. Condtionally accepted for publication.

[17] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. Slam benchmarking webpage. http://ais.informatik.uni-freiburg.de/slamevaluation, 2009.

[18] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[19] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[20] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[21] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.

[22] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[23] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[24] C. Stachniss, U. Frese, and G. Grisetti. OpenSLAM.org – give your algorithm to the community. http://www.openslam.org, 2007.

[25] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard. Evaluation of gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[26] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[27] S. Thrun and colleagues. Winning the darpa grand challenge. *Journal on Field Robotics*, 2006.

[28] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7/8):693–716, 2004.

[29] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Benchmarking urban six-degree-of-freedom simultaneous localization and mapping. *Journal of Field Robotics*, 25(3):148–163, 2008.

[30] M. Yguel, C.T.M. Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[31] Z. Zivkovic, O. Booij, B. Krose, E.A. Topp, and H.I. Christensen. From sensors to human spatial concepts: An annotated data set. *IEEE Transactions on Robotics*, 24(2):501–505, 2008.

[C5] A. Schneider, J. Sturm C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009.

Object Identification with Tactile Sensors using Bag-of-Features

Alexander Schneider Jürgen Sturm Cyrill Stachniss
Marco Reisert Hans Burkhardt Wolfram Burgard

***Abstract*— In this paper, we present a novel approach for identifying objects using touch sensors installed in the finger tips of a manipulation robot. Our approach operates on low-resolution intensity images that are obtained when the robot grasps an object. We apply a bag-of-words approach for object identification. By means of unsupervised clustering on training data, our approach learns a vocabulary from tactile observations which is used to generate a histogram codebook. The histogram codebook models distributions over the vocabulary and is the core identification mechanism. As the objects are larger than the sensor, the robot typically needs multiple grasp actions at different positions to uniquely identify an object. To reduce the number of required grasp actions, we apply a decision-theoretic framework that minimizes the entropy of the probabilistic belief about the type of the object. In our experiments carried out with various industrial and household objects, we demonstrate that our approach is able to discriminate between a large set of objects. We furthermore show that using our approach, a robot is able to distinguish visually similar objects that have different elasticity properties by using only the information from the touch sensor.**

I. Introduction

Touch is one of the five traditional senses that were already described by Aristotele. Humans use and rely on the sensor information from the skin while manipulating objects for a variety of sub-tasks, such as object localization, identification, and grip estimation. Additionally, there are many everyday objects that appear visually similar but can be easily distinguished using tactile sensing such as ripe versus unripe fruits. Also blind people heavily rely on their touch sense, using it to read and manipulate objects.

Accordingly, it seems very desirable to also have robots equipped with tactile sensors. Over the past years, several promising approaches have been developed on the technological or sensor side. Artificial skins that measure orthogonal pressure at comparably high spatial and temporal resolutions are typically composed of elastic, conductive, or resistive polymers, which change their electrical properties depending on the applied pressure. They can, in principle, be manufactured to cover larger parts of a robot at relatively low cost. Several research groups reported [9], [10], [11] to have successfully wrapped substantial parts of the surface of their robots using such sensors, for example, to ease human-machine interaction or to improve the robustness of object manipulation tasks.

In this paper, we show how a robotic manipulator can identify various industrial and household objects purely from the observations of its touch-sensitive fingers. Given low-resolution intensity images recorded with the artificial skin, we apply k-means as unsupervised clustering on the training dataset to create a vocabulary for our bag-of-features classifier. Based on the vocabulary, we learn a codebook histogram. This histogram is a distribution over the occurrence of cluster centroids in the observed dataset. The robot is then able to use these distributions for robustly recognizing a large set of different objects requiring only a small number of grasp actions carried out at different positions. We also present an approach based on a decision-theoretic framework to minimize the number of required grasp actions. In particular, our approach efficiently estimates the expected information gain of potential future grasp actions based on the observations made during training. Experiments carried out with a large set of different objects demonstrate that our approach is able to reliably discriminate between objects. It is even possible to differentiate objects that are visually similar.

Fig. 1: **Left:** A manipulation robot with touch-sensitive finger tips learns to distinguish a large set of objects (here: a coffee mug) solely by using its touch sense. In the image, both the cup and (at the bottom end) the two fingers of the robot's gripper are visible. **Right:** Tactile images of the sensor array in the left and the right finger. The robot is perceiving the handle of the cup.

II. Related Work

Tactile sensors [12], [19] are commonly defined as "a device that can measure a given property of an object or contact event through physical contact between sensor and object" that is able to sense one or more of the following modalities: pressure, normal and sheer forces, torques, slip, vibrations, or temperature. Important properties of a sensor are its spatial and temporal resolution, noise, hysteresis, creep, and aging. Different mechatronic principles have been explored in the past, such as pressure-sensitive conduc-

All authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany
{schneida, sturm, stachnis, reisert, hans.burkhardt, burgard}@informatik.uni-freiburg.de

tive polymers [20], piezo-resistive sensors [7], piezo-electric vibration sensors [13], and capacitive sensors which can additionally measure the sheer forces [4] or temperature [3].

Tactile sensors have been used in the past to explore the 3D shape of objects [2]. Others have used tactile sensors to detect ridges and bumps in the material [14] by sliding the robotic finger over an object. Sensors based on piezo-electric vibration have been used to determine the hardness/softness of probed (biological) objects [15]. Force-sensitive fingers have been used to control the robot's position [6], i.e., to continuously keep the finger in physical contact while moving the object. It has also been shown that tactile sensors can be used to estimate the 3D pose of objects with known shapes [16]. Notably, little information is recovered from the tactile sensor in this work, resulting in multi-modal distributions due to ambiguities during the first grasps, which is a problem we are also dealing with in our work. A work relatively close to ours is that of Russel *et. al.*, who used tactile sensors for object classification [17]. Their approach extracts geometric features like point, line, or area contacts and integrates them time to classify the objects into generic classes such as boxes, spheres, cylinders etc. Later, Russel [18] showed that a similar approach can also be used for object classification using an 8-whisker tactile sensor on a robotic gripper. In contrast to their work, our method is not restricted to pre-defined geometric shapes. Rather, our method is able to recognize typical real-world objects with arbitrary shapes.

III. Touch Sensor Observations

A. Sensor principle

The robot used for gathering the data and carrying out the experiments is a RWI B21r robot equipped with a 7-DOF manipulator. The robot's end-effector is a 1-DOF gripper consisting of two fingers which both are equipped with a Weiss Robotics sensor DSA 9205 for gathering tactile images [20]. Each tactile sensor array contains 84 sensor cells arranged in 6 columns and 14 rows with a size of 24 mm by 51 mm. The maximum scanning rate for the sensor is 240 fps. Each sensor cell measures the conductivity of an elastic rubber foam above it. When a force is applied to the rubber foam, the binding polymer gets compressed thus lowering the electrical resistance of the material. The calibration of the sensor array turned out to be difficult in consequence of the sensor principle. For example, due to memory effects of the rubber foam, we took a reference measurement before the experiments were started (with no pressure on all cells). Furthermore, we normalized all measurements to the sensor's maximum response, such that we obtained for each finger a measurement $Z \in [0,1]^{6\times 14}$.

B. Notation

In the remainder of this paper, we use the following notation for a single touch observation $\mathbf{z}$ as

$$\mathbf{z} =< Z^{left}, Z^{right}, h, w >,$$

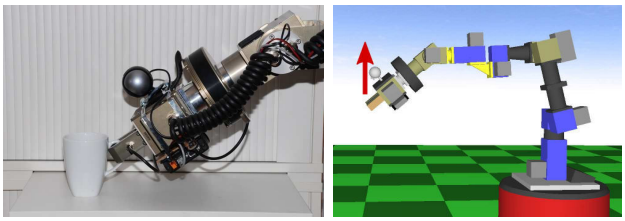


Fig. 2: Experimental setup. The robot grasps an object o at different positions. Each tactile observation $\mathbf{z}_i$ is then stored together with the object label in the database $\mathcal{D} = \{< \mathbf{z}_1, o_1 >, \ldots, < \mathbf{z}_N, o_N >\}$.

where $Z^{left}, Z^{right} \in [0,1]^{6\times 14}$ are the observations of the sensor matrix of the left and right finger, while $h \in \mathbb{R}$ refers to the current height of the gripper and $w \in \mathbb{R}$ refers to the current opening width of the fingers.

C. Data Acquisition

To acquire the training data, we present a set of n different objects denoted by $O = \{1, \ldots, n\}$ to the robot, including industrial objects like metal cuboids or cylinders and household objects like a cup, a toy, and a bottle. The robot grasps each object m times at different heights. This results in a set of $N = nm$ observations $\mathcal{D} = \{< \mathbf{z}_i, o_i >\}_{i=1}^{N}$. From this set, we sample training sets $\mathcal{D}_{\text{training}}$ for our experiments, including the true object labels, and a disjoint test sets $\mathcal{D}_{\text{test}}$ without the object labels for evaluation.

D. Distance Metric for Tactile Observations

Two images R, S (here $R, S \in [0,1]^{6\times 14}$) can be compared by computing the Euclidean distance pixel by pixel:

$$d(R,S) = \sum_x \sum_y |r_{xy} - s_{xy}|. \tag{1}$$

To allow for small translations of the object in the robot's fingers, we do not discount vertical shifts, i.e.,

$$\mathit{dist}(R,S) = \min_{\tau=1,\ldots,k} (d(\mathbf{R}, \mathit{shift}(\mathbf{S}, \tau)). \tag{2}$$

From there, we can define a distance function for the difference between observations $\mathbf{z}_1, \mathbf{z}_2$ as

$$\begin{aligned} \mathit{dist}(\mathbf{z}_1, \mathbf{z}_2) &= \alpha \left(\mathit{dist}(Z_1^{\text{left}}, Z_2^{\text{left}}) + \mathit{dist}(Z_1^{\text{right}}, Z_2^{\text{right}}) \right) \\ &+ (1-\alpha)|w_1 - w_2|, \end{aligned} \tag{3}$$

where $\alpha \in [0,1]$ is a weighting factor determining the influence of differences in touch and finger distance. In order to circumvent scaling issues between both distance measures, we normalized both of them to have unit variance on our training dataset.

IV. The Bag-of-Features Approach

As the finger of the robot is much smaller than all of our objects, the tactile observations the robot perceives of these objects are generally only partial views. To perform the classification based on these local image patches, we apply a variant of the so-called bag-of-features approaches [21],

Fig. 3: Application of the bag-of-features approach with 3 objects described using 5 features. The objects are: bottle, beer glass, and coffee mug. The features in the vocabulary are thick, medium, thin vertical, thin horizontal, and thin diagonal feature. The robot grasps each object multiple times at different positions, indicated by the highlighted rectangles. This results in a characteristic histogram per object, containing the occurrence frequency of each feature in the object.

[5], [1] which have been successfully applied in the area of computer vision. Bag-of-features techniques are appealing because of both, their simplicity and power. The key idea of the bag-of-features approach is to describe the observations with a common vocabulary of features. For tactile perception, the vocabulary might include features such as "straight", "round", and "thin" observations. Given that the feature vocabulary is rich enough, the resulting feature histograms are well suited for object classification. For this purpose, a so-called codebook needs to be learned that contains the feature histograms of the trained objects. Figure 3 graphically illustrates the process of the codebook generation over objects given a vocabulary.

A. Unsupervised Creation of the Tactile Vocabulary

In practice, the appropriate vocabulary strongly depends on the objects that the robot is supposed to grasp so that predefined vocabularies will not suffice in general. Therefore, our approach is to learn a set of characteristic features automatically from the observed training data. To achieve this, our approach applies the k-means clustering algorithm directly on the observed training data $\mathbf{z} \in \mathcal{D}_{\text{training}}$. This results in k cluster centers (or centroids) $\mathbf{c}_1, \ldots, \mathbf{c}_k$, computed according to Alg. 1. The centroids are the individual words of our vocabulary. During k-means clustering, we use the distance function as defined in Eq. 3, thereby allowing the tactile images to have small vertical displacements by Eq. 2. In the remainder of this paper, we consider the set of clusters/centroids as the vocabulary that we use to describe the tactile observations.

B. Codebook Generation

As already mentioned above, the vocabulary described in the previous section is used to generate a codebook. A codebook entry $\mathbf{h}^o$ for an object o describes the distribution over centroids calculated from the training data. Each $\mathbf{h}^o$ is a histogram with k bins, $\mathbf{h}^o \in \mathbb{R}^k$. The overall set of such histograms for the codebook is denoted by H.

To build up a codebook, we initialize $\mathbf{h}^o = \mathbf{0}$ and update each bin h_i^o of $\mathbf{h}^o$ according to the observations $\mathbf{z}$ of object o in $\mathcal{D}_{\text{training}}$ by

$$h_i^o \leftarrow h_i^o + \exp(-\mathit{dist}(\mathbf{c}_i, \mathbf{z})/l), \tag{4}$$

where l is the length scale parameter in the observation distance space. After processing all observations, the individual $\mathbf{h}^o$ must be normalized.

Initialize $c_i, i = 1, \ldots, k$ to k random $\mathbf{z}_t \in \mathcal{D}$
repeat
  forall $z_t \in \mathcal{D}$ **do**
    $b_i^t \leftarrow \begin{cases} 1 & \text{if } \mathit{dist}(\mathbf{z}_t, \mathbf{c}_i) = \min_j \mathit{dist}(\mathbf{z}_t, \mathbf{z}_j) \\ 0 & \mathit{otherwise} \end{cases}$
  end
  forall $c_i, i = 1, \ldots, k$ **do**
    $\mathbf{c}_i \leftarrow \sum_t b_i^t \mathbf{z}^t / \sum_t b_i^t$
  end
until $\mathbf{c}_i$ *converge* ;

Algorithm 1: The k-means clustering algorithm is used to generate a vocabulary for the bag-of-features approach.

The key idea of the codebook is to have a compact representation of the objects that allows us to efficiently compute the likelihood that a new observation (in $\mathcal{D}_{\text{test}}$) is generated by touching a specific object o. In the next section, we explain how to compute this likelihood.

C. Observation Model

To compute the distribution over potential object classes based on an observation, we proceed as follows. By applying Bayes rule, we can write

$$p(o \mid \mathbf{z}) \quad = \quad \eta p(\mathbf{z} \mid o) p(o), \tag{5}$$

where η is a normalizing constant ensuring that the left-hand side sums up to one over all o. The term $p(o)$ is the prior over the objects. In practice, this can be estimated from the training data or alternatively assumed to be uniformly distributed.

To compute the observation model $p(\mathbf{z} \mid o)$, we generate a histogram $\mathbf{h}^{\mathbf{z}}$ of a single observation $\mathbf{z}$ according to Eq. 4. As a result, we have two distributions over feature occurrences, and thus, we can express $p(\mathbf{z} \mid o)$ by computing the similarity between the feature histogram of current observation and the histogram stored in the codebook.

In the literature, there exist multiple ways for computing the similarity between histograms. Among the popular measures for comparing histograms [8] are the histogram intersection, the χ^2 distance, and the Kullback Leibler divergence (KLD). In our experiments, the histogram intersection turned out to yield the best results. This is probably due to the fact that the χ^2 distance and the KLD are heavily influenced by features with low support – an effect that can be observed frequently in our dataset. Thus, the observation model, which is based on the histogram intersection, is given by

$$p(\mathbf{z} \mid o) \quad \propto \quad \sum_{i=1}^{k} \min(\mathbf{h}_i^z, \mathbf{h}_i^o). \tag{6}$$

Fig. 4: Various objects used for the experiments. **Top row:** Visual image of each object **Bottom row:** Tactile images of left and right finger of each object **From left to right:** cuboid, triangle, t-object, handle, cylinder, door key, large cup, small cup, goofy, tiger, figure, mobile, bottle, kaleidoscope, tennis ball, and soft ball.

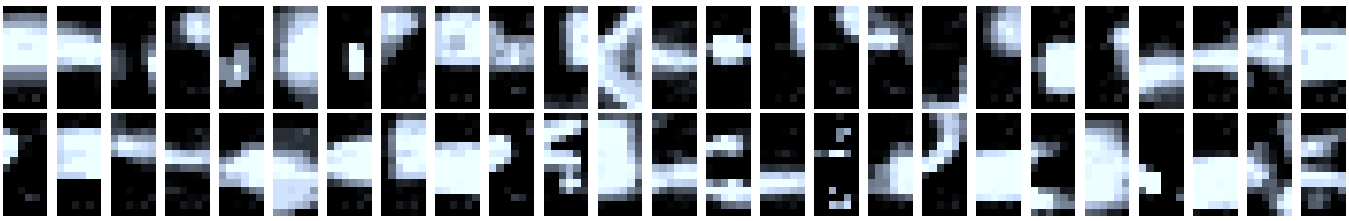


Fig. 5: Vocabulary $\mathbf{c}_1, \ldots, \mathbf{c}_k$ created using unsupervised clustering from the training data with $k = 50$ clusters. For visualization purposes, only the centroid corresponding to the tactile image of the left finger is depicted.

V. Selecting Observation Actions

To identify an object, the robot has to carry out multiple grasping actions at different height levels. Intuitively, it seems that an uninformed grasping strategy is not optimal. For example, a large number of grasps might be needed to distinguish kitchen utensils that have similar shafts. We therefore propose an informed technique based on concepts from information theory. Our approach seeks to determine the action which will provide the highest expected information gain, that is, the highest reduction of uncertainty in the posterior about potential object identity. Here, the expected information gain is the expected change of entropy in the posterior $p(o)$ of the identity o of a grasped object. The entropy is defined as

$$H(p(o)) \quad = \quad \int_o p(o) \log p(o) \, do. \tag{7}$$

Let $a_{1:t}$ be the actions carried out until the current time step t and let $\mathbf{z}_{1:t}$ be the corresponding observations. The robot then has to select the action a_{t+1} that provides the highest expected reduction in entropy. Let $\hat{a}$ be an action under consideration and $\hat{\mathbf{z}}$ be the corresponding observation that is obtained when carrying out $\hat{a}$.

Then, the information gain is given by

$$I(\hat{\mathbf{z}}, \hat{a}) \quad = \quad H(p(o \mid \mathbf{z}_{1:t})) - H(o \mid \mathbf{z}_{1:t}, \hat{\mathbf{z}})). \tag{8}$$

In general, we do not know which measurements the robot will obtain while executing action $\hat{a}$. Therefore, we have to integrate over all possible measurements $\hat{\mathbf{z}}$ to compute the expected information gain

$$E[I(\hat{a})] \quad = \quad \int_{\hat{\mathbf{z}}} p(\hat{\mathbf{z}} \mid \hat{a}, z_{1:t}) I(\hat{\mathbf{z}}, \hat{a}) \, d\hat{\mathbf{z}}. \tag{9}$$

Unfortunately, reasoning about all potential observations is intractable for real world applications since the number of potential measurements grows exponentially in the dimension of the measurement space. A practical approximation, however, is to sum over observations stored in the training set instead of integrating over the whole observation space:

$$E[I(\hat{a})] \quad \approx \quad \sum_{\hat{\mathbf{z}} \in \mathcal{D}_{\text{training}}} p(\hat{\mathbf{z}} \mid \hat{a}, z_{1:t}) I(\hat{\mathbf{z}}, \hat{a}) \, d\hat{\mathbf{z}} \tag{10}$$

Depending on the size of the training database, this sum might still be expensive to compute. To further reduce the complexity, one can easily down-sample the training set.

This approach allows us to approximate the posterior efficiently since we can directly utilize the discrete posterior about the identity of an object. The approximations substantially reduce the number of potential observations that have to be estimated by simulation compared to the number of possible measurements the sensor can generate. The ability to carry out such computations efficiently is an important prerequisite for informed action selection.

After having computed the expected information gain for each action under consideration, we select the action a_{t+1} with the highest expected utility

$$a_{t+1} = \operatorname*{argmax}_{\hat{a}} E[I(\hat{a})]. \tag{11}$$

Every time the robot has to make the decision of where to grasp next, it uses Eq. 11 to determine the action a_{t+1} with the highest expected information gain and executes it. As soon as no action provides an expected reduction of uncertainty or the robot reached a given level of certainty, the identification task is completed.

In addition to the expected reduction of the entropy, one typically has to consider a second quantity when selecting the next action. This quantity is the actual cost of carrying out an action, which needs to be traded off with the expected information gain. In our setting, however, the cost measured

k	10	20	30	40	50
% correct	58.2%	72.8%	71.7%	84.4%	83.0%

TABLE I: Parameter study on the number of clusters k.

α	0.00	0.25	0.50	0.75	1.00
% correct	66.9%	84.3%	81.0%	78.3%	76.0%

TABLE II: Parameter study on the weight α between (normalized) image distance and (normalized) gripper distance.

in terms of time needed to carry out an action can be assumed to be identical for all actions since the movements of the manipulator are carried out quickly without major differences in the time. Thus, we ignore the time needed by the manipulator for changing the height. Considering such a cost, however, can be done in a straightforward manner by adding a cost term to Eq. 11.

VI. Experimental Results

A. Raw Data from the Touch Sensor

For testing our approach, we recorded tactile data for 16 different objects as shown in Fig. 4. The first 5 objects are industrial parts with a relatively similar shape (like a metal cube, a cylinder, and a triangle), while the latter 13 objects were household objects such as cups, toys, and bottles. We created a database of tactile observations by grasping each object on a pre-defined path (from bottom to top). Some objects were included twice in the dataset, both under $0°$, and $90°$ rotation. We obtained a set of $|\mathcal{D}| = 830$ tactile observations for 21 object labels. All experiments were then carried out on this dataset using randomized, 2-fold cross validation, resulting in two disjoint sets $\mathcal{D}_{\text{training}} \subset \mathcal{D}$ and $\mathcal{D}_{\text{test}} \subset \mathcal{D}$ of $|\mathcal{D}_{\text{training}}| = |\mathcal{D}_{\text{test}}| = 415$ samples for each run.

B. Vocabulary Creation

Before each run of our experiments, a vocabulary was created from the training data $\mathcal{D}_{\text{training}}$ by running the k-means algorithm. An example of the resulting centroids is given in Fig. 5. We tried different choices for k empirically, and found by evaluating the resulting recognition rates that $k = 50$ was a reasonable choice for the number of clusters (see Tab. I). Alternatively, one could try to find k automatically, for example, by using the Bayes information criterion (BIC). Further, we studied the influence of the weighting factor α in the distance metric of Eq. 3 (see Tab. II). For all subsequent experiments, we chose $\alpha = 0.5$ such that both the tactile images and the finger distance were considered being equally important.

C. Recognition Rates

For measuring the recognition rate, we chose an object o at random, and selected $T = 10$ random grasp observations $\mathbf{z}_{1:T}$ of that object from $\mathcal{D}_{\text{test}}$. Starting from a uniform prior $p(o)$ over all object classes, we computed the posterior $p(o \mid \mathbf{z}_{1:T})$ according to Eq. 5 and Eq. 6. From this posterior, we then selected the maximum-a-posteriori (MAP) object class

$$\hat{o} \quad = \quad \operatorname*{argmax}_{o} p(o \mid \mathbf{z}_{1:T}) \tag{12}$$

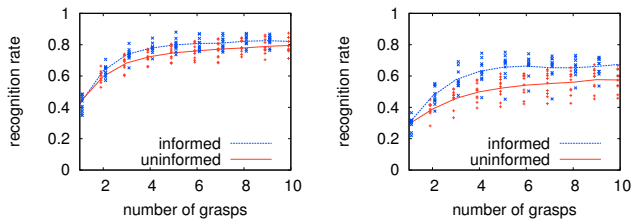


Fig. 6: Comparison of the uninformed and the informed grasping strategy depending on the number of grasp actions. **Left:** household and industrial objects (full dataset). **Right:** industrial objects only.

and compared it to the true object class o. In this way, we obtained a recognition rate of 84.6% over all 21 objects. In particular, we found that the household objects among each other were hardly ever confused (96.2%), in contrast to the industry objects (58.0%), that look very similar. The confusion matrices of these experiments are depicted in Fig. 7.

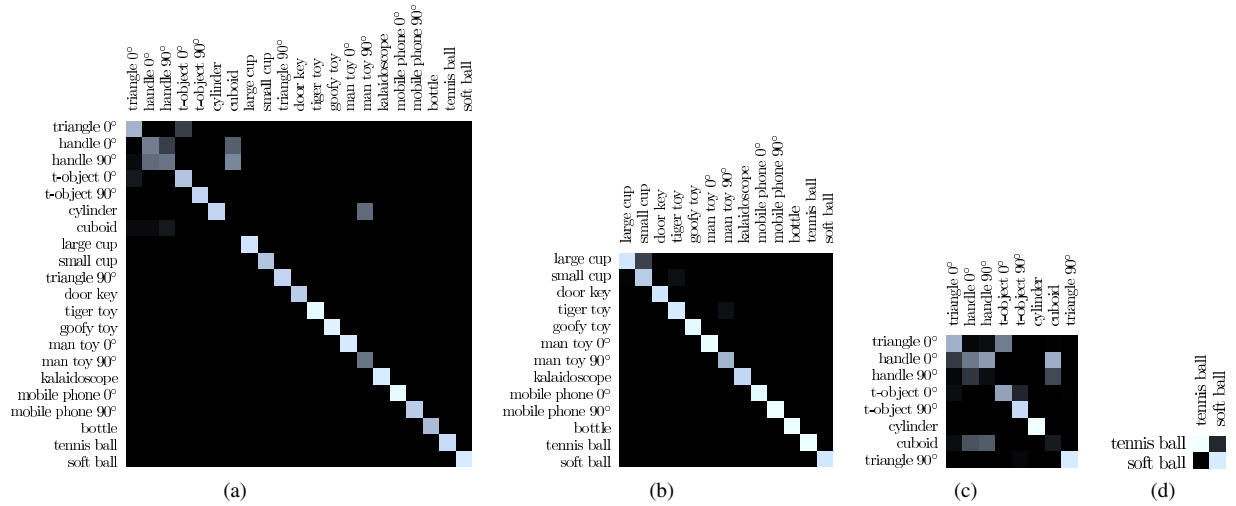
In our dataset, we also have a tennis ball and a soft ball, two objects that appear visually almost completely similar. In our experiments, we measured that these two objects could be separated from each other very well, with a recognition rate of 93.8%.

D. Active perception

We also measured whether objects can be recognized with fewer grasps when the robot selects the grasping height based on the expected information gain. We evaluated the recognition rates after each test grasp in 10 independent runs using both the uninformed and the informed grasping strategy. The results are summarized in Fig. 6. On our full dataset (household and industrial objects), using the information gain strategy performs on average 5.0% better than random grasping. In particular, one would expect that a better grasping strategy improves the recognition rates on the more difficult dataset of industrial objects. Indeed, we measured a performance gain of 18.9% of the informed strategy over the random one, raising the average recognition rate from 58.0% to 67.5%. In both experiments, a paired t-test showed significantly improved recognition rates when choosing the position that maximizes the expected information gain.

VII. Conclusion

In this paper, we presented a novel approach for object recognition using tactile observations obtained from the touch-sensitive fingers of a manipulation robot. Our work belongs to the class of bag-of-features techniques and maintains a probabilistic belief about the object that is currently grasped. We create a feature vocabulary for the tactile observations using k-means clustering. To recognize objects, we learn distributions over the feature vocabulary and use this to create a codebook. Furthermore, we implemented a decision-theoretic approach to active grasping that considers the expected information gain of future actions which significantly improved the recognition performance. We validated



(a) (b) (c) (d)

Fig. 7: Confusion matrices of object recognition after 500 object recognition trials with 10 test grasps each on different subsets of objects. (a) household & industrial objects, recognition rate: 84.6% (b) household objects only, recognition rate: 96.2% (c) industrial objects only, recognition rate: 58.0% (d) tennis ball vs. soft ball, recognition rate: 93.8%

our approach in experiments with a large range of real-world objects and obtained highly accurate recognition results.

Despite the encouraging results there are several warrants for future research. Instead of using the entire tactile images, we want to explore the possibility of using local features that are invariant to translations and orientations of the object. Additionally, we want to look at estimating the pose of the gripped object which could be beneficial during object manipulation. Also, we want to look at the time profiles of the tactile observations, for example to recognize whether and how objects are deformable.

ACKNOWLEDGMENT

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 as well as by the EC under contract number FP6-IST-045388-INDIGO.

REFERENCES

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[2] P. K. Allen. Integrating Vision and Touch for Object Recognition Tasks. *The International Journal of Robotics Research*, 7:15–33, 1988.

[3] F. Castelli. An integrated tactile-thermal robot sensor with capacitive tactile array. *IEEE Transactions on Industry Applications*, 38, 2002.

[4] C. Chuang and R. Chen. 3D capacitive tactile sensor using DRIE micromachining. In C. Cane, J.-C. Chiao, and F. Vidal Verdu, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5836, pages 719–726, July 2005.

[5] G. Csurka, L. Dance, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proceedings of the 8th European Conference on Computer Vision, Wokshop on Statistical Learning in Computer Vision, ECCV 2004*, pages 59–74, 2004.

[6] Z. Doulgeri and S Arimoto. Force position control for a robot finger with a soft tip and kinematic uncertainties. *Robotics and Autonomous Systems*, 55(4):328 – 336, 2007.

[7] Y. Hasegawa, M. Shikida, T. Shimizu, T. Miyaji, H. Sasaki, K. Sato, and K. Itoigawa. Micromachined active tactile sensor for hardness detection. *Sensors and Actuators A: Physical*, 114(2-3):141 – 146, 2004.

[8] G. Hetzel, B. Leibe, P. Levi, and B. Schiele. 3D object recognition from range images using local feature histograms. In *Proc. of the Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pages 394–399, 2001.

[9] T. Hoshi and H. Shinoda. Robot skin based on touch-area-sensitive tactile element. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3463–3468, 2006.

[10] R. Kageyama, S. Kagami, M. Inaba, and H. Inoue. Development of soft and distributed tactile sensors and the application to a humanoid robot. In *IEEE International Conference on Systems, Man, and Cybernetics 1999 (IEEE SMC 99)*, volume 2, pages 981 – 986, 1999.

[11] O. Kerpa, K. Weiß, and H. Wörn. Development of a flexible tactile sensor system for a humanoid robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, 2003.

[12] M. H. Lee and H. R. Nicholls. Review article tactile sensing for mechatronics–a state of the art survey. *Mechatronics*, 9(1):1 – 31, 1999.

[13] K. Motoo, T. Fukuda, F. Arai, and T. Matsuno. Piezoelectric vibration-type tactile sensor with wide measurement range using elasticity and viscosity change. *Advanced Robotics*, 24(3), 2006.

[14] A.M. Okamura and M. R. Cutkosky. Haptik exploration of fine surface features. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[15] S. Omata, Y. Murayama, and C. E. Constantinou. Real time robotic tactile sensor system for the determination of the physical properties of biomaterials. *Sensors and Actuators*, 112(2-3):278 – 285, 2004.

[16] A. Petrovskaya, O. Khatib, S. Thrun, and A. Y. Ng. Bayesian estimation for autonomous object manipulation based on tactile sensors. In *ICRA 2006*, pages 707–714, 2006.

[17] R.A. Russell. Object recognition by a 'smart' tactile sensor. In *Proceedings of the Australian Conference on Robotics and Automation*, Melbourne, Australia, 2000.

[18] R.A. Russell and J.A. Wijaya. Object location and recognition using whisker sensors. In *Proceedings of the Australian Conference on Robotics and Automation 2003*, 2003.

[19] J. Tegin and J. Wikander. Tactile sensing in intelligent robotic manipulation - a review. *Industrial Robot: An International Journal*, 32, 2005.

[20] K. Weiss and H. Wörn. The Working Principle of Resistive Tactile Sensor Cells. *IEEE International Conference on Mechatronics & Automation*, 2005.

[21] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *Proc. of the Conference on Computer Vision and Pattern Recognition Workshop*, Washington, DC, USA, 2006.

[C6] H. Strasdat, C. Stachniss, and W. Burgard. Which landmark is useful? learning selection policies for navigation in unknown environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

Which Landmark is Useful? Learning Selection Policies for Navigation in Unknown Environments

Hauke Strasdat Cyrill Stachniss Wolfram Burgard

Abstract— **In general, a mobile robot that operates in unknown environments has to maintain a map and has to determine its own location given the map. This introduces significant computational and memory constraints for most autonomous systems, especially for lightweight robots such as humanoids or flying vehicles. In this paper, we present a novel approach for learning a landmark selection policy that allows a robot to discard landmarks that are not valuable for its current navigation task. This enables the robot to reduce the computational burden and to carry out its task more efficiently by maintaining only the important landmarks. Our approach applies an unscented Kalman filter for addressing the simultaneous localization and mapping problems and uses Monte-Carlo reinforcement learning to obtain the selection policy. Based on real world and simulation experiments, we show that the learned policies allow for efficient robot navigation and outperform handcrafted strategies. We furthermore demonstrate that the learned policies are not only usable in a specific scenario but can also be generalized towards environments with varying properties.**

I. Introduction

In recent years, there has been a trend towards embedded systems in robotics. A series of such approaches deal with autonomous cars, helicopters, blimps, underwater vehicles, and wheeled or humanoid robots. As embedded systems typically have much higher limitations with respect to the computational power and memory capacity, it is important in the context of embedded systems to develop efficient algorithms that scale with the computational constraints of the underlying hardware.

In robotics, one of the core capabilities needed for the majority of applications is autonomous navigation. For truly autonomous navigation in initially unknown environments, the robot has to solve the so-called simultaneous localization and mapping (SLAM) problem [2, 12, 19]. Solving the SLAM problem, however, is computationally demanding and the memory requirements increase with the number of landmarks that need to be maintained by the robot. In practice, there are many scenarios in which the number of visible landmarks during a navigation task is significantly larger than the number of landmarks which can be processed efficiently using an embedded device. This leads to the question which landmark should be stored and maintained by the robot to optimally solve the navigation task. A landmark is only useful if it contributes to keep an accurate pose estimate of the robot at the right time and in such a way that it is valuable for the navigation task. In this paper, we present an approach for learning a landmark selection policy that optimizes the navigation task carried out by the robot given its computational or memory constraints. It is obvious that the utility of a landmarks depends on the type of navigation task. We analyze two types of navigation tasks: A single-goal navigation task and a round-trip navigation task where subgoals are visited more then once. One major advantage of our approach is that the policies are not limited to the environment they have been learned in. Rather, they can also be applied successfully in environments with different properties of the underlying landmark distribution.

This paper is organized as follows. After a discussion of related work, Section III briefly introduces the unscented Kalman filter and its application to SLAM as well as reinforcement learning. Section IV then describes the different navigation tasks considered in this paper. After that, we introduce our approach to learn the optimal landmark selection policy. Finally, we present experimental results carried out in simulation as well as on a real wheeled robot.

II. Related Work

The standard method for SLAM relies on the extended Kalman filter (EKF) [11] or its variants such as the unscented Kalman filter (UKF) [7]. Using these approaches, the computational requirement and memory demand increase with the number of landmarks since the full correlation between the position all landmarks is taken into account. There are many approximative filtering techniques for SLAM [12, 19]. These methods do not incorporate the full correlation between the landmarks, so that the computational constraints are less restrictive. However, their memory demand increases at least linearly with the number of landmarks used.

Recently, Sala *et al.* [15] presented a graph-theoretic formulation for the selection problem of visual features to perform navigation in known environments. The optimal set of features is defined as the minimal set with which navigation is possible. Zhang *et al.* [20] proposed an entropy-based landmark selection method for SLAM. This method specifies a measure about which visible landmark is best in the sense of entropy reduction. However, it only provides a vague guideline for how many features should be selected at a given point in time. Furthermore, Lerner *et al.* [9] presented another quality measure for landmark selection in known environments which is based on the comparison of pose uncertainties. Dissanayake *et al.* [6] suggested a map management which ensures a uniform distribution of

This work has partly been supported by the DFG within the Research Training Group 1103 and under SFB/TR-8 as well as by the European Commission under FP6-IST-34120-muFly and FP7-231888-EUROPA.

H. Strasdat, C. Stachniss, and W. Burgard are with Dep. of Computer Science, University of Freiburg, Germany. H. Strasdat is also with the Dep. of Computing, Imperial College London, UK.

landmarks over the traversed area. Apart from landmark selection, other active methods were presented such as maximizing the SLAM estimate by intelligent path planning [10] or increasing the performance of a soccer playing robot by active sensing [5].

In this paper, we present a new and universal approach for landmark selection in unknown environments. The value of a landmark is measured in terms of how well it improves the navigation/localization capabilities of the robot given the targeted navigation task. This is especially important for robots with restricted resources. We learn a landmark selection policy using Monte-Carlo reinforcement learning [3, 17, 18] and k-nearest neighbor regression [16]. We show in real world and simulation experiments that this technique allows for more efficient robot navigation.

III. Preliminaries

A. Unscented Kalman Filter

The unscented Kalman filter (UKF) is a recursive Bayes' filter that estimates the state $\mathbf{x}$ of an dynamical system in discrete time steps given a sequence of actions $\mathbf{u}$ and observations $\mathbf{z}$. The n-dimensional state vector $\mathbf{x}$ is represented by a multivariate Normal distribution with mean μ and covariance matrix $\Sigma^{(n\times n)}$. The dynamics of the system are described by a *state transition function* g plus Gaussian noise $\epsilon_{g,t}$:

$$\mathbf{x}_t = g(\mathbf{u}_t, \mathbf{x}_{t-1}) + \epsilon_{g,t} \tag{1}$$

Measurements are integrated using the *observation function*:

$$\hat{\mathbf{z}}_t = h(\mathbf{x}_t) + \epsilon_{h,t} \tag{2}$$

Again, Gaussian noise $\epsilon_{h,t}$ is added. Since Kalman filtering is an approach for systems governed by a linear difference equation, special efforts have to be taken to take the non-linearities in g and h into account.

The key idea of the unscented Kalman filter, which has been introduced by Julier and Uhlmann [7], is to apply a deterministic sampling technique that is known as the unscented transform to select a small set of so-called sigma points around the mean. Then, the sigma points are propagated through the non-linear functions. Afterwards, mean and covariance estimates are computed based on the transformed points. The advantage of this technique is that the filter can much better deal with non-linearities and thus lead to a more robust technique than the EKF.

B. Simultaneous Localization and Mapping

In the context of the SLAM problem, one seeks to simultaneously determine the map of the environment and the pose of the robot. Probabilistic methods seek to estimate the joint probability distribution

$$p(\mathbf{p}_t, \mathbf{l}_1, ..., \mathbf{l}_M | \mathbf{u}_1, ..., \mathbf{u}_t, \mathbf{z}_1, ..., \mathbf{z}_t) \tag{3}$$

about the pose $\mathbf{p}_t$ of the robot at time t and the position of the landmarks $\mathbf{l}_1, ..., \mathbf{l}_M$ given all previous motions $\mathbf{u}_1, ..., \mathbf{u}_t$ and observations $\mathbf{z}_1, ..., \mathbf{z}_t$. Various approaches to estimate this posterior have been presented in the literature.

In this paper, we address the SLAM problem using the UKF by representing the joint state $(\mathbf{p}_t^T, \mathbf{l}_1^T, ..., \mathbf{l}_M^T)^T$ with $\langle \mu, \Sigma \rangle$. This is a standard approach which has shown to operate successfully in the past. For convenience, we abbreviate the mean of the robot pose $(\mu_1, \mu_2, \mu_3)^T$ as $(x, y, \theta)^T$. The mean of the jth landmark location $(\mu_{2j+2}, \mu_{2j+3})^T$ is denoted by $\left(l_x^{[j]}, l_y^{[j]}\right)^T$. Furthermore, we interpret the state transition function h as the robots motion model. In addition, we assume that range and bearing observations $(\rho, \phi)^T$ are given so that we can define a corresponding observation model g. In our work, we initialize new landmarks in a single step following the approach of Bailey [2].

Note that our approach is not limited to UKF or Kalman filter-based approaches and any other method can be applied for addressing the SLAM problem.

C. Monte-Carlo Reinforcement Learning

The basic idea of reinforcement learning [18] is to learn by the interaction with the environment. We consider a dynamical system consisting of an agent and its environment at discrete time steps τ. At each point in time τ, the world is in state $s_\tau \in \mathcal{S}$ and the agent chooses an action $a \in \mathcal{A}$. Then, the world transform into a new state $s_{\tau+1}$ and the agent receives an reward $r_{\tau+1} \in \mathcal{R}$. The goal is to maximize the return

$$R_\tau = \sum_{k=\tau+1}^{\mathcal{T}} r_k, \tag{4}$$

whereas $\mathcal{T}$ is the total number of time steps of one learning episode. The agent is following a policy

$$\pi(s, a) := p(a|s) \quad \forall s \in \mathcal{S} \tag{5}$$

which represents the probability of choosing action a under the assumption of being in state s. Each policy π has a corresponding Q-function

$$Q^\pi(s, a) := E_\pi\{R_\tau | s_\tau = s, a_\tau = a\} \tag{6}$$

which specifies the expected return R from choosing action a in state s. During the learning process, we would like to approximate the optimal policy $\pi^*(s, a)$ that maximizes the expected return. Therefore, we have to approximate the corresponding Q-function simultaneously.

One way of solving the reinforcement learning problem is based on Monte Carlo methods [3, 17]. Here, we estimate the Q-function as the average return over sample episodes. Initially, the Q-function is initialized with an arbitrary prior. During the training, a soft policy should be used. Thus, it should hold that $\pi(s, a) > 0$ for all possible state-action pairs in order to assure that each state is reachable during the training process. One common soft policy is ϵ-greedy which selects with the high probability of $1 - \epsilon$ the action

$$a^* = \arg\max Q(s, a) \tag{7}$$

that maximizes the expected return and a random action otherwise.

Note that the time index t used in the SLAM setting is not necessarily identical with the discrete time at which the

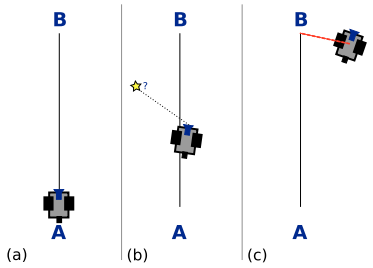


Fig. 1. Illustration of the single-goal navigation task.

reinforcement learning framework has to make decisions. Therefore, we introduced a second time index τ to distinguish both in a sound way.

IV. Navigation Tasks

A. Single-goal Task

Let us consider the following most basic navigation task (see Fig. 1). The robot is located at position A. It is supposed to drive from there to the goal position B. In this example, the robot's motion is affected by a drift. In addition, N landmarks are distributed randomly over the environment. When the robot perceives a new landmark, it has to decide whether it should integrate this landmark in the UKF or not. The UKF has a landmark capacity of M landmarks with $M << N$. The goal is to choose the landmarks in a way such that the distance of the final position of the robot $(x_T, y_T)^T_{\text{true}}$ and the target position B is minimized. Hence, we define the reward

$$r_\tau = \begin{cases} -\left|B - (x_T, y_T)^T_{\text{true}}\right| & \text{if } \tau = \mathcal{T} \\ 0 & \text{else,} \end{cases} \tag{8}$$

as the negative Euclidean distance of the robot's true position to the goal B if the training episode reaches the terminal state $s_\mathcal{T}$; intermediate rewards r_1,...,$r_{\tau-1}$ are set to zero.

B. Round-trip Task

In the round-trip task, the robot is supposed to reach several subgoals. First, it starts at A and it is supposed to drive to B, then back to A. Afterwards, it should target B again and, finally, it should return to A. In this task, a new subgoal is selected as soon as the position estimate of the robot $(x_t, y_t)^T$ is close to it – independent of the robot's true position $(x_t, y_t)^T_{\text{true}}$. In this task, the error in the pose estimate should be minimized over the whole trajectory. For convenience, we specify the return directly as the negative average localization error over the remaining trajectory,

$$R_\tau = -\frac{1}{|T - t(\tau)|} \sum_{t'=t(\tau)}^{T} \left| \begin{pmatrix} x_{t'} \\ y_{t'} \end{pmatrix}_{\text{true}} - \begin{pmatrix} x_{t'} \\ y_{t'} \end{pmatrix} \right|, \tag{9}$$

whereas $t(\tau)$ specifies the time when the τth decision is made and T is the time when the robot reaches its final destination. To simplify things for the second task, landmark selection is only allowed while the robot moves from A to B the first time. The round-trip task is more complex than the previous one. However, it is worth considering since it focuses on the loop-closing problem of SLAM and has a higher a practical relevance than the single-goal task.

V. Navigation and Landmark Selection

A. Motion Control

The robot is steered towards the subgoals using a straightforward controller. An appropriate translational acceleration $\dot{\omega}_t$ and rotational acceleration $\dot{\upsilon}_t$ is selected based on the current estimate of the robot pose $\mathbf{p}_t$, the translational velocity ω_t and rotational velocity υ_t.

B. Learning Landmark Selection Policies

In order to learn landmark selection policies with Monte Carlo reinforcement learning, we need to define the state space $\mathcal{S}$ and the action space $\mathcal{A}$. In addition to that, we need to find an appropriate representation for the continuous Q-function.

1) State Space: The available state information consists of the UKF state $\langle \mu, \Sigma \rangle$ and the current range and bearing observation $(\rho, \phi)^T$. This full information would lead to an high-dimensional state space so that a successful learning is impractical. It is therefore desirable to reduce the space while preserving as much as possible of the relevant information. This can be achieved by defining features that summarize the essential information. One of the features is the position of the potentially new landmark,

$$\begin{pmatrix} l_x^{[\text{new}]} \\ l_y^{[\text{new}]} \end{pmatrix} = \begin{pmatrix} x_t + \rho\cos(\phi + \theta_t) \\ y_t + \rho\sin(\phi + \theta_t) \end{pmatrix}, \tag{10}$$

according to the current range and bearing observation $(\rho, \phi)^T$ and the robot's pose estimate $(x_t, y_t, \theta_t)^T$. Additionally, we define the following five features:

1. Estimated distance to subgoal B,
$$d_{\text{est}} = \left|B - (x_t, y_t)^T\right|, \tag{11}$$
2. Number of landmarks integrated in the UKF,
$$m = \left|\{j \in M : \Sigma_{2j+2} < \infty \wedge \Sigma_{2j+3} < \infty\}\right|, \tag{12}$$
where Σ_{2j+2} and Σ_{2j+3} are the variances of the jth landmark in the x and y direction.
3. Yaw angle to potential new landmarks ϕ,
4. Distance of the potentially new landmark to the closest landmark already integrated,
$$d_l = \min_{\substack{j \in L \\ \text{with } \Sigma_{2j+2} < \infty \\ \wedge \Sigma_{2j+3} < \infty}} \left| \begin{pmatrix} l_x^{[j]} \\ l_y^{[j]} \end{pmatrix} - \begin{pmatrix} l_x^{[\text{new}]} \\ l_y^{[\text{new}]} \end{pmatrix} \right|, \tag{13}$$
5. Uncertainty of the robot pose $\Sigma^{3\times3}$ in terms of its entropy,
$$H = \ln(\sqrt{(2\pi e)^3 |\Sigma^{3\times3}|}). \tag{14}$$

The first of these features summarizes the robot position (x_t, y_t). The landmark positions $\mathbf{l}_1, ..., \mathbf{l}_M$ are summarized by the fourth feature while the new observation $(\rho, \phi)^T$ is represented by the third feature as well as fourth one. The covariance Σ_t is comprised by the second feature and the fifth one.

In the following, we will consider three different variants of the learning approaches. The first approach only relies on a two dimensional state space (first and second feature), the second one uses an four dimensional feature space (first to fourth feature) and the third one uses five dimensions (all five features).

2) Function Approximation: Since the state space of the features is continuous (with the exception of the second dimension), we need to estimate the Q-function with some function approximator. In our current implementation, we use k-nearest neighbor (k-NN) regression [16]. Training points – i.e. state/action values (s, a) which are each labeled with a return R – are efficiently stored in set of kd-trees [4, 1]. The jth kd-tree represents the returns from choosing an action a_j in a given state s. If a query (s', a') is performed, the k nearest data points to the query point s' (w.r.t. Euclidean distance) are selected from the appropriate kd-tree. The return is estimated as unweighted average over the corresponding R-values. If less then k_{min} data points can be found within a fixed radius around the query point, some prior R_{prior} is returned. In our current implementation, we set $k = 50$ to reflect the high amount of noise in our training data; k_{min} is set to 10. The k-NN regression approach has the advantage over the common grid-based discretization methods that it has a high degree of generalization in areas where the density is low and it is precise in regions where the data points are dense. In contrast to non-linear models such as neural networks [14], no over-fitting occurs. As opposed to other regression techniques, in which the model is also expressed directly in terms of their training data such as Gaussian Processes [13], k-NN regression is very fast. Even with hundred thousands of data points, a query can be performed in a few milliseconds. An efficient evaluation is essential for learning in practice, since the regression has to be carried out frequently. In various tests, we could not reveal a significant benefit from using Gaussian process regression over k-NN regression for reinforcement learning in our domain. Due to space restrictions, these experiments are omitted in the experimental section.

3) Action Selection: In our learning problem, the action is a binary decision:

$$\mathcal{A} = \{a_{\text{reject}}, a_{\text{accept}}\} \tag{15}$$

This means that either the potential new landmark is chosen or not. In order to boost the training, a variant of ϵ-greedy is used:

$$\pi(s) = \begin{cases} \arg\max Q(s,a) & \text{if } Q(s, a_{\text{reject}}) \neq Q(s, a_{\text{accept}}) \\ & \text{and } \chi_1 < 1 - \epsilon \\ a_{\text{accept}} & \text{if } [Q(s, a_{\text{reject}}) = Q(s, a_{\text{accept}}) \\ & \text{or } \chi_1 \leq \epsilon] \text{ and } \chi_2 < \frac{M}{N_{\text{visible}}} \\ a_{\text{reject}} & \text{else} \end{cases} \tag{16}$$

Here, χ_1 and χ_2 are uniform random samples between zero and one; N_{visible} is the expected number of visible landmarks in one training episode. Thus, in the beginning of the training – when $Q(s, a_{\text{reject}}) = Q(s, a_{\text{accept}}) = R_{\text{prior}}$ in most cases – it is ensured that landmarks are selected over the whole trajectory. Neither landmarks in the beginning of the episode nor landmarks in the end are preferred. If standard ϵ-greedy is used, a_{accept} and a_{reject} would be chosen with a probability of 0.5 each. Thus, depending on the values for the landmark capacity M and expected number of visible landmarks N_{visible} it could happen that either all landmarks are selected in the beginning of the episode or that considerably fewer landmarks than M are selected. Either would lead to a slow convergence rate.

To sum up, we use a learning approach for landmark selection based on Monte-Carlo reinforcement learning and k-NN regression. The state space is compactly represented by five features and the action is a binary decision.

C. Generalization

Until now, we considered an approach to learn a selection policy in unknown environments, but for a specific scenario. However, it is desirable to train a policy in one scenario and then apply this policy in another setting. Important parameters of a training scenario are the number N of landmarks in the environment and the landmark capacity M of the UKF. To generalize, it is important to have a scenario-independent state space representation. For instance, instead of the number of landmark integrated in the UKF m, we need to speak about the percentage of landmarks $\frac{m}{M}$.

D. Deletion of Landmarks

In the Kalman filter framework, it is possible to delete already integrated landmarks. This can be done without affecting the statistical consistency by removing the appropriate value from the mean vector and the corresponding rows and columns from the covariance matrix [6]. If we want to allow deletion, we must extend our action set $\mathcal{A}$. Since deletion is only useful if we replace the deleted landmark with a new one, we propose the following action set

$$\mathcal{A} = \{a_{\text{reject}}, a_{\text{replace}_1}, ..., a_{\text{replace}_M}\}. \tag{17}$$

The deletion of landmarks might be particular interesting in connection with the round-trip task where those landmarks should not be replaced which facilitate loop closure.

VI. Experiments

A. Single-goal Task in Simulation

We evaluate the performance of our learning procedure for the single-goal task in a simulated environment. We choose an environment where N landmarks are randomly distributed in a $30m \times 60m$ area. The distance between the start position A and the goal B is set to $44m$. We train our policy for 2,000 episodes. In each episode, landmarks are randomly re-distributed. We compare the trained policies with two heuristics. The first one is the *M-first heuristic* which simply integrates the M first landmarks that are observed. An apparently better policy is the *equidistant heuristic*. With this heuristic, the robot only integrates a new landmark after it has driven a certain distance so that the

Test scenario M / N	Policy trained in scenario 5 / 100	5 / 50	10 / 100	10 / 50	15 / 100	15 / 50	equidistant heuristic
5 / 100	**11.5 ± 0.64**	**12.2 ± 0.68**	14.9 ± 1.19	15.6 ± 1.36	16.6 ± 1.05	17.4 ± 0.99	13.2
5 / 50	**10.7 ± 0.42**	**10.5 ± 0.49**	**11.6 ± 0.50**	**12.2 ± 0.52**	**12.7 ± 0.49**	13.5 ± 0.62	13.6
10 / 100	**6.9 ± 0.54**	**6.3 ± 0.47**	**6.8 ± 0.39**	**7.6 ± 0.61**	8.4 ± 0.6	9.7 ± 0.97	8.5
10 / 50	**8.2 ± 1.05**	**7.2 ± 0.48**	**7.1 ± 0.45**	**6.8 ± 0.26**	**7.1 ± 0.47**	**7.2 ± 0.35**	9.6
15 / 100	**5.9 ± 0.87**	**5.1 ± 0.36**	**4.9 ± 0.35**	**5.1 ± 0.24**	**5.1 ± 0.22**	**6.0 ± 0.46**	6.6
15 / 50	8.0 ± 1.24	**6.9 ± 0.71**	**6.5 ± 0.76**	**5.9 ± 0.41**	**6.0 ± 0.22**	**5.6 ± 0.28**	7.5

TABLE I

HIGH DEGREE OF GENERALIZATION IN THE SINGLE-GOAL TASK. THE MEAN ERROR OVER TEN TRAINING RUNS AND THE CORRESPONDING STANDARD DERIVATION IS SHOWN. ALL POLICIES MARKED BOLD ARE SIGNIFICANTLY BETTER THAN THE EQUIDISTANT HEURISTIC ($\alpha = 0.05$).

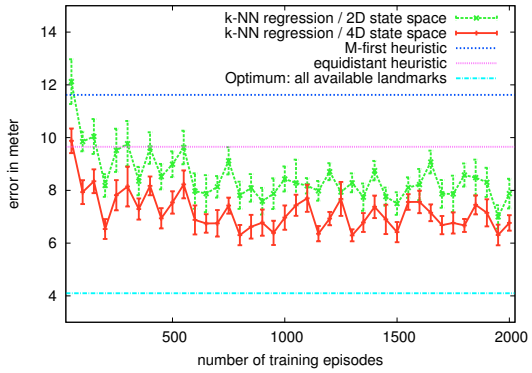


Fig. 2. Evolution of the error during training using k-NN regression. The mean errors over 10 training runs as well as the corresponding 95%-confidence intervals are plotted in the graph. In addition to that, the average performance of two different heuristics over 2000 episodes is shown. Furthermore, the figure shows the average performance of navigation under optimal conditions where all visible landmark are integrated in the UKF.

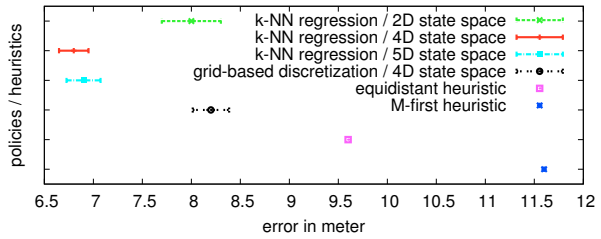


Fig. 3. Average performance of the trained policies and heuristics w.r.t. 1000 test episodes. For the trained policies, the mean over the ten training runs as well as the corresponding 95%-confidence interval is shown.

landmarks are approximately uniformly distributed over the whole trajectory (similar to [6]).

For the learning, we use k-NN regression with a two-, a four-, and a five-dimensional state space (see Section V-B). For comparison, a learning approach using a grid-based discretization is performed. At first, we consider an UKF with a landmark capacity of $M = 10$ and an environment with $N = 50$ landmarks. Fig. 2 shows the evolution of the error of different learning approaches using k-NN regression during the training. The error is defined by the Euclidean distance of the robot to the goal B. For each learning approach, ten training runs are performed. Each trained policy and heuristic is evaluated in 1,000 different environments (see Fig. 3). The one-sample t-test with a significance level of $\alpha = 0.05$ shows that all four learning approaches are significantly better than the equidistant heuristic. Furthermore, it is shown using a two-sample t-test that k-NN regression with a four

Fig. 4. Pioneer 2-DX8 robot with upward-looking camera and SICK laser range scanner (left) and detected visual landmarks on the ceiling (right).

dimensional state space leads to a significant smaller error than grid-based discretization with four dimensions as well as k-NN with two dimensions. Thus, the third feature, which is the distance d_l of a new landmark to landmarks already integrated, and the fourth one, which is the angle ϕ to the new landmark, seem to include relevant information which are not encoded in the first two dimension of the state space. Further experiments revealed that indeed both features are essential. However, we were not able to show that there is any benefit from including the fifth feature, the entropy H of the robot's pose. Even with a significance level of $\alpha = 0.25$, the t-test did not reveal a difference between the learning approach using the four dimensional state space and the one using five dimensions. A qualitative comparison in an example environment between the learned policy and the heuristics is shown in the first half of the accompanying video submission.

In order to evaluate how good the trained policies generalize, we trained and tested a policy in environments with $N = 50$ as well as $N = 100$ landmarks. In addition, we use UKFs with a capacity M of five, ten, and 15 landmarks. Tab. I illustrates the high degree of generalization of our learning approach. For instance, if we perform a training in a setting with $N = 50$ and $M = 5$, we see that the trained policy leads to significantly better results than the equidistant heuristic in all six test scenarios. This indicates that our approach generalized over different landmark densities which is similar to environments of different scale and sensor range.

B. Single-goal Task Performed in a Real World Experiment

Furthermore, we evaluated our learning approach in a laboratory environment. Visual markers [8] have been randomly attached to the ceiling in a $2.5m \times 5m$ area. Our used robot, a Pioneer 2DX-8, is equipped with a upward-looking camera and a SICK laser range scanner (see Fig. 4). The camera is used for the experiments observing landmarks

at the ceiling whereas the laser is used for (near) ground truth evaluation. Since the odometry of the robot was too accurate in the limited space in which we carried out the experiment, we added a rotational bias of 0.1 rad per meter. It is impractical to train the policy in the real-world because this would not only require to perform hundreds of training episodes but also to install different landmark distributions for each training episode. Thus, we trained the policy in simulation and tested it in the real-world setting. We also compared the trained policy to the equidistant heuristic. Both, the trained policy as well as the equidistant heuristic were tested ten times. The trained policy results in an error of 0.50 ± 0.08 whereas the equidistant heuristic leads to an error of 0.66 ± 0.07. Hence, the trained policy is significantly better than the equidistant heuristic (w.r.t. a t-test with $\alpha = 0.05$).

C. Round-trip Task

The performance of our learning procedure for the round-trip task is evaluated in a simulated environment with $N = 50$ landmarks. It was trained using k-NN regression with a four dimensional state space over ten training runs. Here, the error is defined as the average localization error over the whole trajectory. Again, we compare our learning with the equidistant heuristic. Tab. II shows that the learned policy is significantly better than the heuristic. Furthermore, it is shown that we were able to generalize over the UKF capacity M. In the second half of the accompanying video, a qualitative evaluation is given.

D. Round-trip Task with Landmark Deletion

Finally, we analyzed a variant of the round-trip task that, compared to the previous experiments, also allows the deletion of landmarks. It should be noted that this scenario is significantly more complex compared to the previous tasks. For instance, the state space must be extended in order to represent the already integrated landmarks. In initial experiments, we figured out that good strategies keep a set of landmarks fixed for re-localization and perform an incremental pose correction with set of frequently replaced landmarks. Further investigations are currently ongoing.

VII. Conclusion

In this paper, we presented an novel approach for landmark selection in unknown environments using reinforcement learning. The ability of a mobile robot to incorporate a landmark into its belief or to discard it allows for efficient robot navigation under computational constraints. The presented method is able to determine which landmark is valuable for the robot to efficiently solve its current navigation task. This is especially important for robots with restricted resources. We demonstrated by a series of real world and simulation experiments that the learned policies outperform handcrafted heuristics. Furthermore, we showed that a learned policy has a high degree of generalization since it can be applied in different environments with changed underlying parameters.

Despite these encouraging results, there is space for further optimizations. One interesting aspect is the possibility to delete an already incorporated landmark. First experiments indicate that improvements can be made although the problem is substantially more complex.

Test	**Policy trained in scenario**			**equidistant**
M	5	10	15	**heuristic**
5	$\mathbf{3.28 \pm 0.15}$	4.44 ± 0.41	5.06 ± 0.37	3.86
10	$\mathbf{2.38 \pm 0.09}$	$\mathbf{2.37 \pm 0.09}$	$\mathbf{2.43 \pm 0.06}$	2.85
15	$\mathbf{2.30 \pm 0.13}$	$\mathbf{2.24 \pm 0.17}$	$\mathbf{2.23 \pm 0.06}$	2.55

TABLE II

Round-trip task. All policies marked bold are significantly better than the equidistant heuristic ($\alpha = 0.05$).

References

[1] S. Arya and D.M. Mount. Algorithms for fast vector quantization. In *Proc. of the IEEE Data Compression Conference (DDC'93)*, pages 381–390, 1993.

[2] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney, 2002. pages 22–23.

[3] A. Barto and M. Duff. Monte Carlo matrix inversion and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 687–694, 1994.

[4] J.L. Bentley. K-d trees for semidynamic point sets. In *Proc. of the 6th ACM Symposium on Computational Geometry*, pages 187–197, 1990.

[5] Kwok C. and Fox D. Reinforcement learning for sensing strategies. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, Sendai, Japan, 2004.

[6] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, pages 1009–1014, 2000.

[7] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the Int. Symp. on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[8] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. of the Int. Workshop on Augmented Reality (IWAR'99)*, 1999.

[9] R. Lerner, E. Rivlin, and I. Shimshoni. Landmark selection for task-oriented navigation. *IEEE Transaction on Robotics*, 23(3), 2007.

[10] Bryson M. and Sukkarieh S. Active airborne localisation and exploration in unknown environments using inertial SLAM. In *Proc. of the IEEE Aerospace Conference*, 2006.

[11] P.S. Maybeck. The Kalman filter: An introduction to concepts. In *Autonomous Robot Vehicle*. Springer Press, 1990.

[12] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simulaneous localization and mapping problem. In *Proc. of the National Conf. on Artificial Intelligence (AAAI'02)*, pages 593 – 598, 2002.

[13] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.

[14] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer Press, 1996.

[15] P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson. Landmark selection for vision-based navigation. *IEEE Transaction on Robotics*, 22(2), 2006.

[16] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, Cambridge, MA, USA, 2006.

[17] S.P. Singh, R.S. Sutton, and P. Kaelbling. Reinforcement learning with replacing eligibility traces. In *Machine Learning*, volume 22, pages 123–158, 1996.

[18] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, USA, 1998.

[19] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23, 2004.

[20] S. Zhang, L. Xie, and M.D. Adams. Entropy based feature selection scheme for real time simultaneous localization and map building. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, 2005.

[C7] B. Frank, C. Stachniss, R. Schmedding, W. Burgard, and M. Teschner. Real-world robot navigation amongst deformable obstacles. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

Real-world Robot Navigation amongst Deformable Obstacles

Barbara Frank Cyrill Stachniss Rüdiger Schmedding Matthias Teschner Wolfram Burgard

Abstract— **In this paper, we consider the problem of mobile robots navigating in environments with non-rigid objects. Whereas robots can plan their paths more effectively when they utilize the information about the deformability of objects, they also need to consider the influence of the interaction with the deformable objects on their measurements during the execution of their navigation task. In this paper, we present a probabilistic approach to identify the measurements influenced by the deformable objects. Based on a learned statistics about the influence of the deformable objects on the measurements, the robot is able to perform a sensor-based collision avoidance of unforeseen objects. We present experiments carried out with a real robot that illustrate the practicability of our approach.**

I. INTRODUCTION

The ability to safely navigate in their environment is one of the fundamental tasks of mobile robots. Accordingly, the problem of safe navigation has received considerable attention in the past. The majority of approaches for navigation, however, has been developed for environments with rigid obstacles [17, 13] and does not consider the potential deformations imposed on the corresponding objects while the robot navigates through the environment. In the real world, however, not all obstacles are rigid and taking this knowledge into account can enable a robot to accomplish navigation tasks that otherwise cannot be carried out. For example, in our everyday life we often deal with deformable objects such as plants, curtains, or cloth and we are also able to utilize the information about the deformability of the corresponding objects. Consider, for example, the situation in which a curtain blocks a potential path of the robot as depicted in Fig. 1. Without the knowledge that the curtain can be deformed, the robot would always have to take a detour. Precise information about the cost of potential deformations, however, allows the robot to plan cost-optimal paths through the corridor, thereby deforming the curtain at minimal cost.

For robots that operate in environments with deformable objects, two tasks are essential. First, the robot needs to be able to take the cost of deformations resulting from its interaction with deformable objects into account during the path planning process. Furthermore, the robot needs to be able to appropriately interpret its sensory input during the interaction with the deformable objects. For example, during the interaction, the robot necessarily gets close to the deformable object so that its field of view might get obstructed. However, for safe navigation the robot still needs to be able to identify the measurements that do not correspond to the deformable object and come from other, possibly even rigid objects.

All authors are with the Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany.
{bfrank,stachnis,schmedd,burgard,teschner}@informatik.uni-freiburg.de

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8.

Fig. 1. The mobile robot Albert reasoning about its trajectory.

In this paper, we present a probabilistic approach that allows a mobile robot to distinguish measurements caused by deformable objects it is interacting with from ordinary measurements. This allows the robot to utilize standard reactive collision avoidance techniques like potential fields [12] or dynamic window techniques [4, 3, 14] simply by filtering out measurements that are caused by the objects the robot is interacting with. Additionally, the ability to reliably identify measurements not perceiving parts of the deformable object enables the robot to correctly interpret them also for the sake of collision avoidance. Our approach has been implemented on a real robot and evaluated in a collision avoidance task carried out while the robot interacts with a curtain. The results demonstrate that our approach allows the robot to safely avoid obstacles while it is interacting with a deformable object.

This paper is organized as follows. After discussing related work in the following section, we present in Section III an overview of our current navigation system for robots operating in environments with deformable objects. In Section IV, we describe how our robot estimates the cost of deforming objects and how it incorporates this information during the path planning process. Section V then contains our approach to determining which sensors measurements are influenced by the deformable object and how this information can be incorporated into the collision avoidance process. Finally, Section VI contains experimental results.

II. RELATED WORK

Most approaches to mobile robot path planning assume that the environment is static and that all objects are rigid [13, 11, 2]. In the last years, however, path planning techniques for deformable robots in static environments have been presented [7, 10].

In case objects in the environment are deformable, the underlying model for deformations and the model of the environment have a substantial influence on the accuracy of the estimated deformations as well as on the performance of the planner. There exist geometric approaches such as the free-form deformation (FFD) that can be computed efficiently, for example, the FFD method of Sederberg and Parry [19]. Physically motivated approaches use either mass-spring systems [16] or Finite element methods (FEMs) which reflect physical properties of the objects in a better way, see [8, 15].

Kavraki *et al.* [10] developed the f-PRM-Framework that is able to plan paths for flexible robots of simple geometric shapes such as surface patches [9] or simple volumetric elements [1]. They apply a mass-spring model and the planner selects the deformation of the robot that minimizes its deformation energy. Similar to this technique, Gayle *et al.* [7] presented an approach to path planning for a deformable robot that is based on PRMs. To achieve a more realistic simulation of deformations they add constraints for volume preservation to the mass-spring model of the robot.

In the context of collision avoidance, several successful methods have been presented. They are typically executed with a higher frequency compared to path planning, operate mainly on the sensor data itself with the task to ensure collision free motion of the robot. Such methods can roughly be divided into map-based approaches such as road-map or cell-decomposition techniques (see [13] for an extensive overview), and reactive, sensor-based approaches [4, 14, 20]. Such methods are designed to react to unforeseen obstacles but assume all objects to be rigid.

The techniques described by Fox *et al.* [5] as well as Schmidt and Azarm [18] combine the sensory information with a given map of the environment to deal with objects that cannot be detected with the robot's sensors. Brock and Kathib [3] presented an integration of path planning and reactive collision avoidance. There exist also methods that incorporate speed into the planning process in combination with collision avoidance [22].

The techniques mentioned above that are able to deal with deformable objects have been mainly used in simulations and *not on real robots*. When applying those techniques in the real world, a series of problems arise such as how to interpret the sensor data perceived by the robot while it is deforming an object as well as adaptation to the collision avoidance system.

Our planning system applies FEMs to compute object deformations. In order to perform the path planning task efficiently, we precompute potential deformations for a set of robot movements through the objects and estimate the costs by means of regression. This is based on our previous approach [6]. In contrast to [6], we realize in this paper a planning system on a real physical mobile robot and not only in simulation which requires a series of adaptations and new techniques for successfully planning paths in environments with deformable objects. This includes a sound way on how to interpret the sensor data a mobile robot perceives while deforming an object. Our approach allows for filtering the range data obtained with the robot's sensor to label beams that are reflected from a deformable objects. This, in turn, makes our technique orthogonal to other collision avoidance techniques and enables the robot to combine existing techniques with our method. Thus, we explicitly address these open issues and are able to deploy a real robot with the capability of safely moving though environments with deformable objects, leaving the world of simulation behind.

III. SYSTEM OVERVIEW

Our approach to mobile robot motion planning in real environments with deformable objects uses a typical layered architecture for realizing the navigation functionalities. Besides drivers for sensors and the robot, the hardware abstraction layer, etc., three key components are the

- path planning module, the
- collision avoidance module, and the
- localization module.

The path planning system computes trajectories that guide the robot to its desired goal location and is executed with rather low frequency. In contrast to that, a collision avoidance module operates with high frequency in order to avoid collisions with unforeseen and/or dynamic obstacles. Finally, the localization module runs Monte-Carlo localization keeping track of the robot's pose.

In the context of navigation in real environments with deformable objects the key questions are: (i) how to plan trajectories in the presence of such obstacles and (ii) how to interpret the sensor data so that the robot can distinguish between unforeseen obstacles to avoid and deformable objects, which is needed for collision avoidance as well as for localization.

A prerequisite to address these issues is an appropriate model of the environment. First, a traditional map (here grid map) is needed to represent static obstacles. Second, deformable objects need to be modeled. It is, however, significantly more complex to represent deformable objects since one needs to store the three-dimensional structure of the object as well as its elasticity parameters to allow for adequate simulation of deformations.

IV. ROBOT TRAJECTORY PLANNING CONSIDERING OBJECT DEFORMATIONS

A. Learning Deformation Cost Functions

To allow for efficient generation of trajectories for a mobile robot in environments with deformable objects, we build upon our recent work [6]. The key idea is to learn cost functions for the individual deformable objects parameterized

by different trajectories leading to deformations. In order to carry out this task in an efficient manner, a physical simulation engine is used in a preprocessing step to calculate the corresponding cost functions. For making adequate predictions of the object deformations, we apply finite element methods to model the deformations.

Once a set of trajectories deforming an object is simulated in order to obtain the corresponding costs, these values can be used to approximate the deformation cost function. Our path planner then evaluates trajectories using $A^\star$ according to the cost function

$$C(path) = \alpha\, C_{def}(path) + (1-\alpha)\, C_{travel}(path), \quad (1)$$

where $\alpha \in [0,1]$ is a user-defined weighting coefficient that determines the trade-off between deformation and path costs.

Given our current implementation, the robot is able to answer path queries in typical indoor environments in less than 1 second – in contrast to several hours that would be needed if the deformation simulations were carried out at runtime. For further details, we refer the reader to our previous work [6]. It should be noted that our approach makes the assumption that there are no interactions between the different deformable objects and that they are fixed in the environment, such as curtains or (rather heavy) plants.

B. Object Reconstruction

Our previous work dealt with the path planning issues on an abstract level carried out only in simulation. We furthermore assumed that accurate 3D models incorporating the deformation parameters are known. In this work, we go a step further and also learn the 3D model of the objects. This is done by using a real mobile robot equipped with a laser range finder mounted on a pan-tilt unit.

The robot perceives 3D range scans of the object from different perspectives and generates a consistent 3D model by means of the iterative closest point (ICP) algorithm. For the simulation of deformable objects, a tetrahedral mesh is needed, which is reconstructed from the 3D model as shown in [21]. This method can handle un-orientable, non-manifold or damaged surfaces, and is therefore particularly suitable for the reconstruction from 3D scans. Based on the 3D scan, a signed distance field is computed where the set of voxels having negative sign represents the volume of the object. Next, a uniform axis-aligned grid is laid over the distance field. All cells outside the volume are discarded and the remaining cubical cells are split into tetrahedrons. Finally, a smoothing filter is applied to optimize the tetrahedral mesh (see Fig. 2 for example models). Deformations of objects are then computed using a linear relation between the forces and displacements q of the single elements (i.e. the tetrahedrons):

$$f = K(E,\nu)q \quad (2)$$

with stiffness matrix $K(E,\nu)$ depending on the elasticity parameters Poisson ratio ν and Young modulus E.

One open issue is the question of how to determine the elasticity parameters of the individual objects after acquiring the 3D model. In our current system, these parameters are set manually. However, in a future step, we plan to acquire this information autonomously by the robot itself from force-displacement relations obtained with a 7-DoF manipulator. By applying a force to unknown objects and by measuring the displacement, we hope to learn the elasticity parameters. Such a procedure, however, is not yet implemented in our current system.

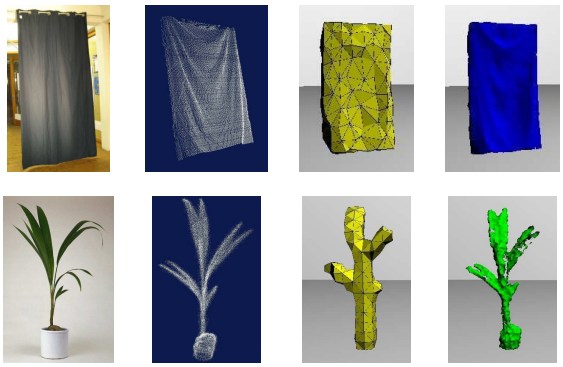


Fig. 2. Generating a model of a curtain (top) and a plant (bottom) for predicting the deformation cost: Left: photo. Second from left: point cloud. Second from right: tetrahedral mesh. Right: 3D model .

V. COLLISION AVOIDANCE

In this section, we describe the collision avoidance system developed for our robot that navigates in environments with deformable objects. Our robot is equipped with a SICK laser scanner with 180 degree opening angle. We use the range measurements for a basic collision avoidance behavior.

When navigating autonomously, the robot constantly has to observe its environment in order to react to unforeseen obstacles. At the same time, it might get close to deformable objects when deforming them. Therefore, the main problem in our setting is to figure out which measurements correspond to a deformable object, which means that these measurements can be ignored by the collision avoidance system. Note that we do not claim that our approach can distinguish deformable from rigid objects only based on laser data in general. However, by combining the knowledge about objects in the environment and their geometry with estimates of range scans during deformations, we can estimate the deformability of an observed object.

We model this problem in a probabilistic fashion: Let c_i denote the binary random variable which describes the event that beam i hits a deformable object. Then, $p(c_i \mid x, z_i)$ describes the probability that beam i hits a deformable object given the robot position x and the range measurement z_i. Applying Bayes' formula, we obtain

$$p(c_i|x,z_i) = \frac{p(z_i|x,c_i)p(c_i|x)}{p(z_i|x,c_i)p(c_i|x) + p(z_i|x,\neg c_i)p(\neg c_i|x)}. \quad (3)$$

Here, $p(z_i \mid x, c_i)$ is the sensor model and $p(c_i \mid x)$ is the prior denoting the probability of observing a deformable object from position x. We will shortly go into detail of how to learn these models. The sensor model $p(z_i \mid x, \neg c_i)$ corresponds to the common sensor model $p(z_i \mid x)$ when no deformable objects are present.

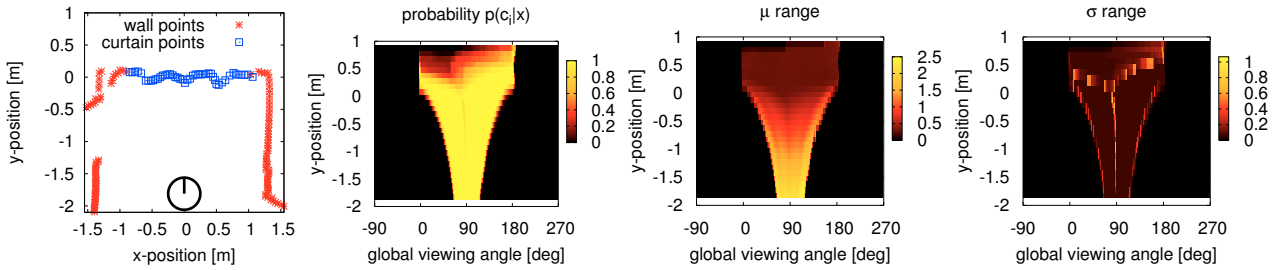


Fig. 3. Sensor model for the trajectory depicted in the left figure: shown are the probabilities $p(c_i \mid x)$ (second from left), the average beam length when observing the deformable object (second from right) as well as the standard deviation σ (right) for the robot position and the viewing angle.

A. Learning sensor models for deformable objects

The sensor model $p(z_i \mid x, c_i)$ not only depends on the robot position but also on the trajectory relative to an object. For instance, the robot will measure a different distance to the curtain when it is situated in front of it than it would while passing through and deforming the curtain. Therefore, we determine sensor models corresponding to different trajectories of the robot relative to an object.

For each trajectory, we record different datasets consisting of the robot positions x (provided by the localization module) and the ranges z_i and then manually label the beams reflected by the deformable object. From the labeled measurements obtained along these trajectories, we compute the statistics

$$p(c_i \mid x) = \frac{hits_{def}}{hits_{def} + misses_{def}}, \tag{4}$$

where $hits_{def}$ is the number of beams reflected by a deformable object and $misses_{def}$ states, how often no deformable object was observed for position x and viewing angle i. The sensor model $p(z_i \mid x, c_i)$ is described by a Gaussian with average range μ and variance σ^2. An example of the deformable sensor model for a typical robot trajectory through the curtain is shown in Fig. 3.

B. Avoiding collisions

During path execution, the robot constantly monitors its position and also its sensor measurements for utilization in the collision avoidance system. In our case, the robot has to distinguish between allowed collisions with deformable objects and impending collisions with unforeseen or dynamic obstacles which have to be avoided. This is done by filtering out the range measurements that observe a deformable object with high probability. Therefore, we evaluate Eq. (3) for each beam and identify those beams that can be neglected for the collision avoidance.

Note that this labeling or filtering of the range measurement offers a great potential since it is done orthogonal to traditional collision avoidance methods. As a result, this technique can be combined with any other collision avoidance technique as, for example, with the dynamic window approach [4] or the nearness diagram technique [14].

The detected measurements which are identified as belonging to dynamic obstacles can be incorporated into the navigation system to update the path of the robot or into any existing sensor based collision avoidance routine. Our current implementation performs replanning if a path is blocked by a dynamic object or simply stops the robot if the distance to an obstacle is too close. An example of the collision detection is given in Fig. 4.

VI. EXPERIMENTAL RESULTS

We performed several experiments to evaluate the performance of our developed planning system on a real robot. We used an iRobot B21r platform equipped with a SICK laser range finder. Our implementation is based on CARMEN which is a navigation software allowing independent modules to communicate via a middle-ware. To integrate our approach into CARMEN, we replaced the collision detection method inside the module "robot" as well as the planning module termed "navigator" with our software. In addition to that, we extended the localization module which is based on MCL so that the laser beams hitting a deformable object during deformation are not considered in the sensor model.

We mounted a set of curtains in the corridor of our lab as deformable objects. First, we evaluate our sensor model for deformable objects. Next, we analyze the performance of our collision avoidance system during path execution in the presence of unforeseen and dynamic obstacles. Finally, we give some examples of how the incorporation of the deformation cost function influences the path search.

A. Sensor model prediction

In the first experiment, we evaluated how well our sensor model for deformable objects is able to predict the presence of deformable objects. We learned a sensor model for two different trajectories through the curtain that were chosen preferably by our path planner. To compute the sensor model statistics, we recorded the laser data and the robot position and manually labeled the laser beams that were reflected by the curtain. For each trajectory, we performed a leave-one-out cross-validation using 11 trajectories for learning the model and one for evaluation. The results of this experiment are summarized in Table I and demonstrate, that the system is able to distinguish between deformable and static obstacles with high accuracy. While the number of false positives is at around 3%, the number of false negatives is below 1%.

B. Recognition of dynamic obstacles

While it is intuitive that the sensor model is able to distinguish between deformable and static obstacles, it is not clear how well the classification works in the presence of dynamic obstacles in the vicinity of the deformable

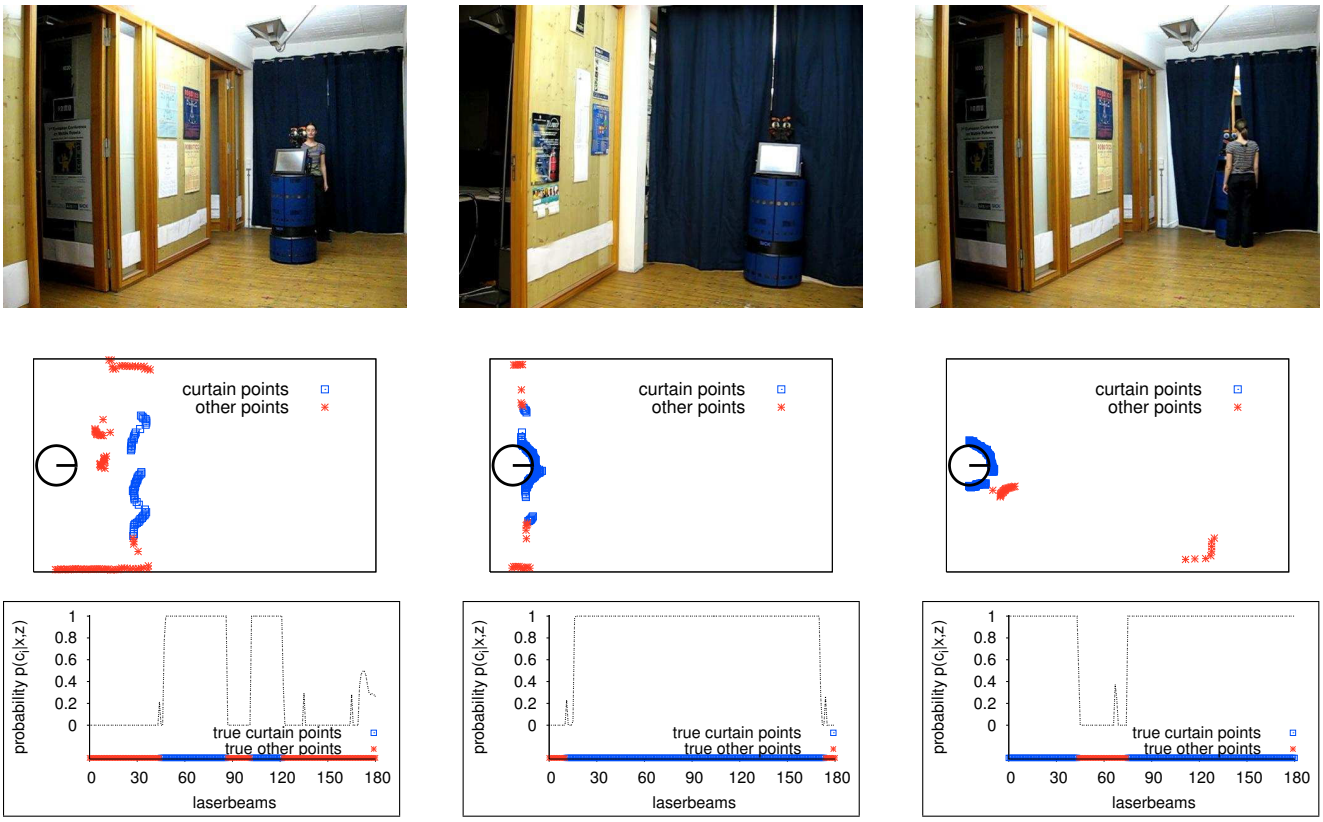


Fig. 4. Different collision avoidance scenarios (top row): Laser beams are evaluated with respect to their likelihood of observing a deformable object. In the second row, the classification of the individual laser beams is illustrated while in the bottom row, the probability of each beam together with the ground truth is shown.

	True label	
Detected label	Deformable Object	No deformable Object
Deformable Object	43857 (97.1%)	621 (0.9%)
No Deformable Object	1292 (2.9%)	65907 (99.1%)
Total	45149	66528

TABLE I

CONFUSION MATRIX FOR PREDICTING WHETHER A BEAM HITS A DEFORMABLE OBJECT IN A STATIC ENVIRONMENT.

	True label	
Detected label	Deformable Object	Dynamic Object
Deformable Object	8563 (96.5%)	98 (2.1%)
Dynamic Object	314 (3.5%)	4600 (97.9%)
Total	8877	4698

TABLE II

CONFUSION MATRIX FOR AN ENVIRONMENT CONTAINING BOTH DEFORMABLE AND DYNAMIC OBJECTS.

objects. The key question is whether the system is able to distinguish well between these obstacle classes and therefore is able to navigate safely. An important precondition for this is of course that the sensor can perceive a dynamic obstacle and that it is not completely occluded by the deformable object. To answer this question we performed several experiments where our robot moved on a trajectory deforming the curtain while dynamic obstacles were blocking its path. The recorded laser scans were labeled accordingly and evaluated with respect to the prediction performance. The results are listed in Table II. In this experiment, the number of false negatives is comparable to the situation in static environments while the number of false positives is around 1% higher than in the previous experiment. Our experiments, however, showed that this still leads to a safe behavior. In the worst case, the false negatives forced the robot to stop when it was not necessary while the false positives usually where outliers in a region of correctly classified dynamic obstacle beams. Therefore, the robot was still able to recognize dynamic obstacles and thus avoided collisions with these obstacles.

C. Example Trajectories through Curtains

For our experimental setup, we varied the trade-off between the deformation cost and the travel cost. The results for an example trajectory can be seen in Fig. 5. When the weighting coefficient α, which determines the trade-off between deformation and travel cost, is set to moderate values, then the planner prefers trajectories going through easily deformable objects. Note that in our scene, the curtain consists of two individual, neighboring curtains. The minimal-cost path, therefore, guides the robot through the contact point of both curtains. This fact can be observed in Fig. 6, where the curtains are moved compared to the previous example. Here, the planner chooses a slightly longer trajectory in order to minimize the deformation costs. Finally, a sequence of snapshots of our real robot navigating through the curtains is shown in Fig. 7. The execution of this

Fig. 7. The mobile robot Albert moving through a curtain.

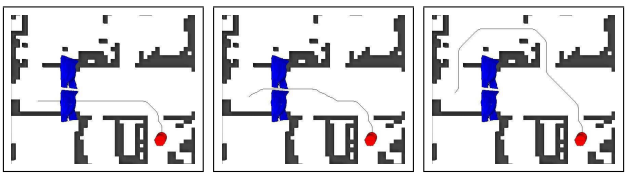


Fig. 5. Planning a trajectory for different weightings of the deformation cost ($\alpha = 0$ (left), $\alpha = 0.2$ (middle), $\alpha = 0.8$ (right)).

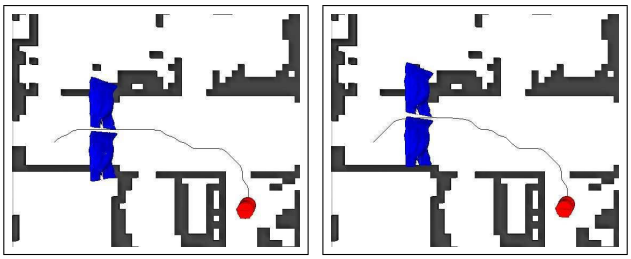


Fig. 6. The planner prefers trajectories that minimize object deformations. The curtains in the left picture are moved 40 cm along the positive y-axis compared to the picture on the right. The weighting coefficient α is set to 0.2 in both examples.

path together with demonstrations of the collision avoidance system can be found in the accompanying video.

VII. CONCLUSIONS

In this paper, we presented an approach for navigation in environments with deformable objects that explicitly takes into account the influence of the interaction between the robot and the deformable objects onto the measurements. Our approach is purely probabilistic and estimates for each measurement as to whether or not it might be caused by the deformable object in the environment. This allows the robot to get close to deformable objects and still avoid collisions with non-deformable objects. In our planning system, the costs of object deformations are determined using finite element methods to appropriately model the physical properties. Additionally we perform pre-computations to allow for an efficient online-calculation of path queries.

Our approach has been implemented and tested on a real robot and in a practical experiment, in which the robot is able to deform objects and at the same time avoid collisions with people. Future work will include the learning of the parameters of the deformable object based on the interaction between the robot and the objects so that better statistics about the influence on the sensory input can be calculated.

REFERENCES

[1] E. Anshelevich, S. Owens, F. Lamiraux, and L.E. Kavraki. Deformable volumes in path planning applications. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 2290–2295, 2000.

[2] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[3] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 1997.

[5] D. Fox, W. Burgard, and S. Thrun. A hybrid collision avoidance method for mobile robots. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[6] B. Frank, M. Becker, C. Stachniss, M. Teschner, and W. Burgard. Efficient path planning for mobile robots in environments with deformable objects. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[7] R. Gayle, P. Segars, M.C. Lin, and D. Manocha. Path planning for deformable robots in complex environments. In *Proc. of Robotics: Science and Systems (RSS)*, pages 225–232, 2005.

[8] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. In *WSCG*, pages 137–145, 2004.

[9] C. Holleman, L.E. Kavraki, and J. Warren. Planning paths for a flexible surface patch. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 21–26, 1998.

[10] L.E. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Robotics: The Algorithmic Perspective*, pages 313–325. A.K. Peters, 1998. Proc. of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR).

[11] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[12] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. Int. Conf. on Intelligent Autonomous Systems (IAS'4)*, 1995.

[13] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Pub., 1991.

[14] J. Minguez and L. Montano. Nearness diagram navigation (nd): A new real time collision avoidance approach. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2094–2100, 2000.

[15] M. Mueller and M. Gross. Interactive Virtual Materials. In *Graphics Interface*, pages 239–246, 2004.

[16] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[17] N.J. Nilsson. A mobile automation: An application of artificial intelligence techniques. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1969.

[18] K. Schmidt and K. Azarm. Mobile robot navigation in a dynamic world using an unsteady diffusion equation strategy. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 1992.

[19] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. In *Proc. of the Conf. on Computer graphics and interactive techniques*, pages 151–160, 1986.

[20] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 1996.

[21] J. Spillmann, M. Wagner, and M. Teschner. Robust tetrahedral meshing of triangle soups. In *Proc. Vision, Modeling, Visualization (VMV)*, pages 9–16, 2006.

[22] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 508–513, 2002.

[C8] M. Bennewitz, C. Stachniss, S. Behnke, and W. Burgard. Utilizing reflection properties of surfaces to improve mobile robot localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

Utilizing Reflection Properties of Surfaces to Improve Mobile Robot Localization

Maren Bennewitz Cyrill Stachniss Sven Behnke Wolfram Burgard

Abstract— **A main difficulty that arises in the context of probabilistic localization is the design of an appropriate observation model, i.e., determining the likelihood of a sensor measurement given the pose of the robot and a map of the environment. Many successful approaches to localization rely on data provided by range sensors, e.g., laser range scanners. When using such data one normally has to deal with erroneous maximum-range readings that occur due to poor-reflecting surfaces. In general, these readings cannot be distinguished from readings obtained when no obstacle is within the measurement range of the sensor. Therefore, existing localization techniques treat these readings alike in the observation model. In this paper, we present a novel approach that explicitly considers the reflection properties of surfaces and thus the expectation of valid range measurements. In addition to the expected range measurement, we compute the probability of reflectance for a beam given the relative pose of the robot to the obstacle taking into account the angle of incidence of the beam. We estimate the reflection properties of surfaces using data collected with a mobile robot equipped with a laser range scanner. As we demonstrate in experiments carried out with a real robot, our technique leads to significantly improved localization results compared to a state-of-the-art observation model.**

I. INTRODUCTION

Robust localization is a prerequisite for mobile service robots operating in the real world. Several tasks, such as deliveries [1], giving tours [2] as well as assisting people [3], can only be carried out if the robot knows its pose.

Since sensor data is noisy, probabilistic approaches that explicitly take the uncertainty into account are typically applied to estimate the pose of the robot. One of the key problems in probabilistic localization is the design of the observation model. For a given pose of the robot and the map of the environment, the observation model specifies the likelihood of a sensor measurement.

Laser range sensors have been widely used for successful localization [4], [5], [6], [7], [8]. They provide distance and bearing information to objects in the environment. In practice, one has to deal with erroneous readings also called "maximum-range" readings that result from poor-reflecting surfaces or readings obtained in situations in which no obstacle is within the measurement range of the sensor. Especially, low-cost laser range sensors suffer from maximum-range readings caused by objects with low reflection properties. One popular approach that explicitly models failures corresponding to low or non reflectance is the ray-cast model proposed by Fox *et al.* [5]. This model, however, does not take into account that the likelihood of erroneous readings depends on the reflection properties of the corresponding surfaces.

In this paper, we propose to estimate the reflection properties of surfaces and to use this information in the observation model. We collect data with a mobile robot equipped with a low-cost, miniature laser range scanner to estimate the distances and angles from which the objects are detected by the scanner. We compute histograms for the number of detections and non-detections for regions in the environment given the viewing angle and the viewing distance. We then use this information to calculate the probability of reflectance for a beam given the pose of the robot in the map.

We apply the well-known Monte-Carlo localization (MCL) technique [9] to estimate the robot's pose and use a variant of the ray-cast model [5] in which expected measurements are compared to measured distances. The novelty of our approach is that we additionally take into account the viewing angle and distance to the objects contained in the map to calculate the probability of reflectance for a beam.

As we demonstrate in the experiments carried out with a real robot, we can significantly improve the localization compared to the standard ray-cast model. Our approach is especially valuable when large parts of a range scan are erroneous maximum-range readings due to low reflectance of objects or due to a comparably short sensor range. Such effects may occur rarely when using a highly accurate SICK laser range finder but can be observed frequently when using low-cost, light-weight scanners such as the Hokuyo URG-04LX. This sensor is often used for humanoid robots [10] or flying vehicles [11] since these types of robots have only a very limited payload of a few hundred gram.

This paper is organized as follows. After reviewing related work, we explain in Section III how to learn reflection properties of surfaces. In Section IV, we present our novel observation model for MCL. Finally, our experimental results illustrate that the accuracy of our localization approach and demonstrates the significantly improved performance compared to the standard ray-cast model.

II. RELATED WORK

Various observation models for probabilistic localization based on laser range data have been proposed [12], [13]. These approaches either approximate the characteristics of the sensor or aim to increase the robustness of the localization process by smooth likelihood models. Thrun *et al.* [9] as well as Lenser and Veloso [14] observed that the likelihood

M. Bennewitz, C. Stachniss, and W. Burgard are with the Institute for Computer Science, University of Freiburg, Germany and S. Behnke is with the Institute for Computer Science, University of Bonn, Germany

This work has been supported by the DFG under the contract number BE 2556/2-2 and SFB/TR-8 as well as by the EC under FP7-231888-EUROPA.

function can have a serious influence on the performance of the localization technique.

In the standard ray-cast model proposed by Fox *et al.* [5], it is assumed that beams are reflected by the first obstacle in the map along the ray with the robot's pose as origin. Expected distances, which can be computed easily for a pose of the robot given the map, are then compared to the actually measured distances. In this approach, failures due to low or non reflectance are considered in the observation model and it has been successfully applied in practice. However, it is not taken into account that different surfaces can have distinct reflection properties. As we show in our experiments, this variance in reflection properties may substantially influence the localization performance especially when they lead to larger numbers of erroneous maximum-range readings.

In correlation-based methods presented by Konolige and Chou [6] and by Schiele and Crowley [15] as well as in the endpoint model proposed by Thrun [7], the likelihood of a single range measurement depends on the distance of the corresponding beam endpoint to the closest obstacle represented in the map. Whereas these models haven been shown to be robust in highly cluttered environments, they suffer from two drawbacks. First, they do not take into account visibility constraints, second, they provide no direct mechanism to deal with maximum-range readings, which is why these readings are typically ignored.

Gutmann *et al.* [4] and Arras *et al.* [8] presented feature-based localization approaches. Here, a set of features is extracted out the range scan and matched to features contained in the representation of the environment. One drawback of these methods are the assumptions the feature extractor makes about the structure of the environment.

Moravec and Elfes [16] presented an approach to mapping with sonar sensors. When using sonar sensors, one encounters the inherent problem of specular reflection which means that sonar beams are reflected between different objects resulting in false range measurements. Moravec and Elfes simply do not consider range readings above a certain distance since they assume that specular reflection results in readings near the maximum range. Lim and Cho [17] proposed to use specular reflection probabilities to compute a measure of reliability for range readings. This quantity is computed given the measured distance and the angle of incidence. The difference to our approach is that we explicitly model that objects yield either valid measurements or erroneous maximum-range readings depending on the viewing angle and distance. Our goal is not to discount measurements with a low reliability or to ignore readings. Instead, we seek to utilize all measurements in an appropriate fashion during localization.

III. ESTIMATING REFLECTION PROPERTIES

A. Standard Reflection Probability Maps

In our approach, the environment is represented using a grid map that consists of equally spaced cells. Reflection probability grids are typically computed using so-called hits and misses which are counted for each cell [13]. The number

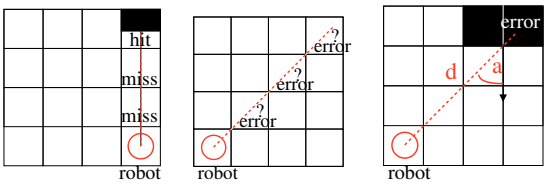


Fig. 1. (Left) The cell containing the beam endpoint gets assigned a hit, the cells the beam passes through get assigned a miss. (Middle) In the first stage of our mapping approach, we do not know which object along the ray of the beam caused the erroneous maximum-range reading. Thus, all cells along the ray get assigned the error. (Right) In the second stage, we assign the error reading to the first occupied cell along the ray (which has been observed from a different pose). We update the histogram given the distance d and the angle a.

of hits represents the number of cases a beam ended in the corresponding cell. The endpoint of a beam can easily be computed given the robot's pose and the measured range. The number of misses corresponds to the number of cases a beam passed through the cell. The cells a beam passes through can be determined via ray-casting. Consider the example depicted in Fig. 1 (left) for an illustration. The reflection probability of a cell (x, y) is then computed as

$$p_{ref}(x, y) = \frac{hits_{x,y}}{hits_{x,y} + misses_{x,y}}. \quad (1)$$

A classical reflection probability map does not model the case that erroneous maximum-range readings can occur when the beam hits an object with a poor-reflecting surface and such beams are ignored during mapping. In our approach, we explicitly take into account the case that the beam is reflected by the object or that a too small fraction of the light has been reflected by the surface and thus an error reading is obtained.

When using highly accurate laser range sensors such as the SICK laser scanner, these considerations can in general be neglected since the mentioned effects occur rarely. In contrast to that, range measurements obtained by low-cost and light-weight range scanners, such as the Hokuyo URG-04LX, are highly sensitive to the surface material of the measured object. In our experience, the probability of a valid measurement depends on the viewing angle, i.e., the angle of incidence of the beam as well as on the distance to the object. In the following, we describe how to estimate such reflection properties of surfaces.

B. Estimating Reflection Properties from Laser Data

Learning reflection probabilities requires an accurate estimate about the robot's trajectory. To acquire such an estimate, we use a robot that is equipped with two laser range finders: an accurate SICK LMS and a comparably noisy Hokuyo URG-04LX. To compute the trajectory of the robot given *accurate* laser range data, we apply an approach to grid-based SLAM (Simultaneous Localization and Mapping) with Rao-Blackwellized particle filters. A detailed description of this approach can be found in [18]. We then use the obtained pose estimates to learn a reflection probability map of the environment given data of the *noisy* laser range scanner.

We count the number of hits and misses for each cell. Additionally, the number of error readings potentially caused by each cell is determined (see Fig. 1, center image). We use histograms that store those values for a discrete set of viewing distances and angles of incidence. To compute the angle of incidence of a beam, we estimate the normal of the surface in a neighborhood in the scan around the beam under consideration by fitting a line through neighboring endpoints. Here, we assume that the normal can be uniquely determined for each grid cell (which typically have a size of $5 \times 5\,\text{cm}$).

When learning these histograms, two problems arise. First, it is not clear which cell along the beam caused the error reading given an invalid measurement. Second, the histograms are typically not completely filled since the cells are not observed from all distances and angles. In the following, we describe how to deal with these two problems.

1) Two-stage Mapping: To deal with the first problem, we apply a two-stage mapping approach. In the first stage, we process the sensor data to count the number of hits, misses, and errors for each cell. Here, misses are counted for all cells along the ray within a distance below the maximum measurement range of the sensor. To obtain a decision at which cell a ray-casting operation is aborted, we calculate for each cell:

$$bin(x,y) = \begin{cases} 1 & \text{if } \frac{hits_{x,y}}{hits_{x,y}+misses_{x,y}} > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Afterwards, we initialize the histograms for all occupied cells that have error readings assigned and process the sensor data again. This time, we can assume that error readings are caused by the first occupied cell along the beam if one exists within the maximum measurement range of the sensor. Fig. 1 (middle) and (right) visualize the error assignment in the two stages of mapping. In case an error reading occurs, we update the histogram of the occupied cell (x, y), which is assumed to have caused the error reading. This is done given the distance d and the angle a computed based on the map. This means that $err(d, a)$ is increased for (x, y). Accordingly, $hits(d, a)$ is updated for (x, y) when a beam with length d and angle a ends in the cell (x, y) .

Since we only need the histograms at occupied cells, our representation is only slightly more complex than that of standard reflection probability maps.

2) Dealing with Incomplete Data: We assume a monotonic increase of the probability of error readings with distance. In practice, one typically observes a cell from many different angles but only from very few distances. Therefore, if an error reading occurs at distance d_e, we also count an error for all greater distances $d_i > d_e$ in the histogram given the angle of incidence a:

$$\forall d_i > d_e : err(d_i, a) \leftarrow err(d_i, a) + err(d_e, a) \quad (3)$$

For reasons of readability, we omit the cell (x, y) in the formulas. The motivation behind this approach is the mode of operation of a laser scanner. The sensor emits light and measures the time needed until a certain amount of the reflected light is observed by the detector in the scanner. If not enough light is reflected, we obtain an error reading. In this case, either no object was within the range of the sensor or too little light was reflected in direction of the sensor. The amount of the reflected light given a surface depends non-trivially on the angle of incidence and the distance between the sensor and the object. The closer the object to the sensor and the closer the angle of incidence to the normal angle of the measured object, the higher the amount of reflected light.

Similarly to Eq. (3), for a hit at distance d_h, we count this hit also for all shorter distances $d_i < d_h$ in the histogram given the angle of incidence:

$$\forall d_i < d_h : hits(d_i, a) \leftarrow hits(d_i, a) + hits(d_h, a) \quad (4)$$

To estimate the maximum angle of incidence for which we expect to obtain valid range measurements, we proceed as follows. Given the histogram for a certain distance, we search for a separator line that best explains the corresponding hit and error readings. The optimal separator *given the data* can be determined by means of a score function we have to maximize. The score function is defined as

$$\begin{aligned} score(d,a) &= \sum_{i=0}^{a} [hits(d,i) - err(d,i)] + \\ &\quad \sum_{i=a+1}^{B} [err(d,i) - hits(d,i)], \end{aligned} \quad (5)$$

where B refers to the number of bins in the angular histogram. According to this function, we compute for each distance d the angle a^* that gets the highest score and thus best separates the hit observations from the error observations

$$a^* = \operatorname*{argmax}_a score(d,a). \quad (6)$$

Fig. 2 illustrates the separator line (dashed line) for an example histogram and Fig. 3 depicts the corresponding score function.

For probabilistic localization, we need to compute the probability that an occupied cell reflects a beam or leads to an error reading. From physics, we know that the closer the angle of incidence of a beam is to the normal angle of the observed object, the higher the amount of reflected light. It is therefore reasonable to assume that there exists one angular difference where there will be a valid measurement for smaller values and an error reading for bigger ones. As a result, we use a step function to model this probability. Given the separator a^*, we can do this in a straight-forward manner. We then can compute the probability of reflectance given d averaged over the all angles a smaller than a^* as

$$p_{ref}(d, a < a^*) = \frac{\sum_{a'=0}^{a^*} hits(d,a')}{\sum_{a'=0}^{a^*} hits(d,a') + \sum_{a'=0}^{a^*} err(d,a')}, \quad (7)$$

and for $d, a < a^*$ accordingly. Since a^* cannot always be determined perfectly due to missing data (see also Fig. 2), we do not use Eq. (7) directly but apply linear interpolation between $p_{ref}(d, a < a^*)$ and $p_{ref}(d, a > a^*)$ for all angles around a^* where no data is available. See Fig. 4 for an illustration.

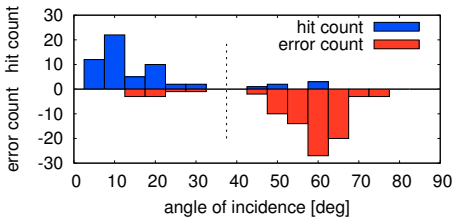


Fig. 2. Illustration on how to compute the optimal separator a^* (here illustrated by the dashed line).

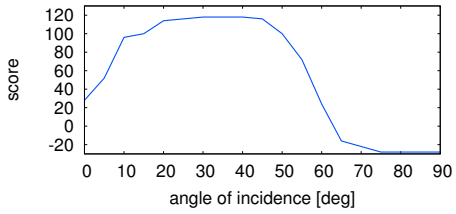


Fig. 3. Score function corresponding to the histogram show in Fig. 2.

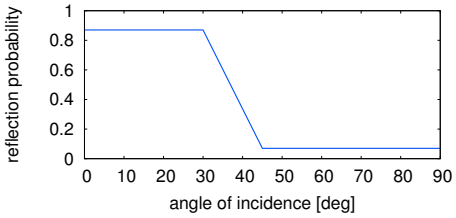


Fig. 4. Resulting reflection probability function.

Note that if enough statistical data was collected with the robot, these approximations would not be necessary. In this case, one could compute the reflectance probability and analogously the probability of obtaining an error reading for each discrete angle a separately. In practice, however, it is typically impossible to observe each cell with each angle of incidence and from each distance. In the following section, we describe how to use the information about the reflectance properties of objects during probabilistic localization with a particle filter.

IV. CONSIDERING REFLECTION PROPERTIES DURING LOCALIZATION

A. Monte-Carlo localization

We apply Monte-Carlo localization (MCL) [9] to estimate the pose x_t of the robot at time t. MCL recursively estimates the posterior about the robot's pose based on the following equation

$$p(x_t \mid z_{1:t}, u_{0:t-1}) = \eta \cdot \underbrace{p(z_t \mid x_t)}_{\text{observation model}} \cdot \int_{x_{t-1}} \underbrace{p(x_t \mid x_{t-1}, u_{t-1})}_{\text{motion model}} \cdot \underbrace{p(x_{t-1} \mid z_{1:t-1}, u_{0:t-2})}_{\text{recursive term}} dx_{t-1}. \quad (8)$$

Here, η is a normalization constant, $u_{0:t-1}$ denotes the sequence of motion commands up to time $t-1$, and $z_{1:t}$ is the sequence of observations. For reasons of readability, the map m is neglected in Eq. (8).

MCL uses a set of particles to represent the above posterior. This set is updated using the sampling-importance-resampling particle filter.

B. Improved Observation Model

We propose an improved observation model $p(z_t \mid x_t)$ that takes into account the estimated reflection properties explained in the previous section to realize a more robust and efficient localization for robots equipped with low-cost laser range finders such as the Hokuyo URG-04LX.

Typical observation models for MCL, that use ray-casting operations in the map to compute the expected distance of a sensor measurement, are defined as a weighted sum of functions. In Section 6.3 of [13], Thrun *et al.* propose to apply a uniform distribution to model a random measurement, an exponential function to better cope with people in the vicinity of the robot, a Gaussian to model the measurement uncertainty of the sensor, and a constant for maximum range readings.

In general, such ray-cast models can be described by

$$\begin{aligned} p(z_t \mid x_t, m) \quad = \quad & \alpha \cdot p_{max\_range}(z_t \mid x_t, m) + \\ & \beta \cdot p_{exp\_dist}(z_t \mid x_t, m) + \\ & \gamma \cdot p_1(z_t \mid x_t, m) + \ldots, \end{aligned} \quad (9)$$

where p_{max_range} describes the function that determines the likelihood to obtain maximum-range readings, p_{exp_dist} handles the measurement noise in the sensor and is typically modeled by a Gaussian. Other properties can be described by the functions labeled $p_1, \ldots$. The terms α, β, γ, $\ldots$ represent weights that can be set manually or learned by techniques such as expectation maximization.

Our novel technique, that considers the angle of incidence of the beam as well as the distance of the robot for each cell individually to estimate the reflection probability, can be seen as orthogonal to such models and can easily be combined with them. We can directly integrate our knowledge about the reflection properties of cells into Eq. (9) without the need to change the individual functions:

$$\begin{aligned} p(z_t \mid x_t, m) \quad = \quad & \alpha \cdot p_{err}(d, a) \cdot p_{max\_range}(z_t \mid x_t, m) + \\ & \beta \cdot p_{ref}(d, a) \cdot p_{exp\_dist}(z_t \mid x_t, m) + \\ & \gamma \cdot p_1(z_t \mid x_t, m) + \ldots, \end{aligned} \quad (10)$$

where $p_{ref}(d, a)$ denotes the probability of reflection of the first occupied cell along the beam given distance d and angle of incidence a, and $p_{err}(d, a) = 1 - p_{ref}(d, a)$ stands for the corresponding probability for an error reading. These probabilities are computed for a cell given the histogram as described in the previous section.

As a result, the reflection properties of surfaces of the individual cells are directly incorporated into the existing observation model. Perfectly reflecting surfaces will lead to the original observation model. However, if the surface in the corresponding cell has different properties, our model explicitly takes these properties into account which results in more appropriate distributions as our experiments demonstrate.

V. EXPERIMENTAL RESULTS

To evaluate our approach, we carried out experiments with a wheeled robot equipped with a noisy and short-range (theo-

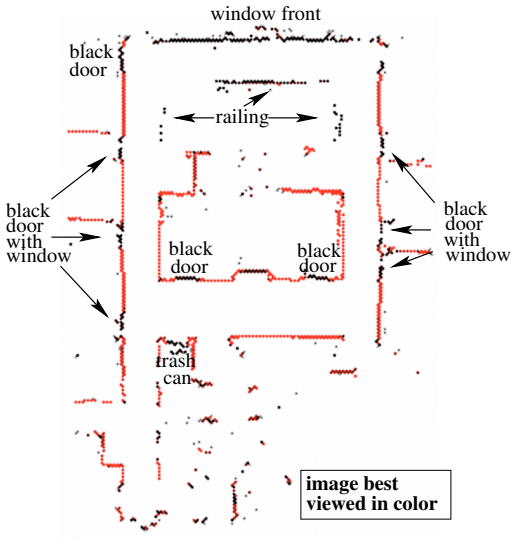


Fig. 5. Enhanced reflection probability map. Different colors of cells indicate the maximum angle of incidence until which valid measurements are expected (the brighter the color red, the bigger the angle). The shown angles correspond to a measurement distance of $0.5 - 1\,m$.

retical maximum measurement range: $5.6\,m$) Hokuyo URG-04LX laser range scanner. The scanner provides a $240°$ field of view. To estimate the trajectory of the robot and to obtain ground truth data, we used the data of the SICK LMS laser scanner (maximum measurement range: $80\,m$, $180°$ field of view) which was also mounted on the robot. In contrast to the Hokuyo, the SICK sensor provides highly accurate distance data.

A. Learning an Enhanced Reflection Probability Map

In the dataset used to learn the enhanced reflection probability map of the environment, the robot traveled approximately $300\,m$. We manually steered the robot through the environment which has a size of approximately $20 \times 20\,m$. Afterwards, we estimated the trajectory of the robot from the data of the SICK scanner using a SLAM approach with Rao-Blackwellized particle filters [18]. We then used the resulting trajectory to learn a representative map of the environment for the data of the Hokuyo scanner.

The enhanced reflection probability map learned by our approach is visualized in Fig. 5. The map has a resolution of $5\,cm$. The different colors of the cells indicate the maximum angle of incidence until which valid measurements are expected (the darker the color the smaller the angle). The shown angles correspond to the measurement distance of $0.5\,m$ to $1\,m$. For this dataset, we used an angular discretization of $5°$ and a distance discretization of $50\,cm$.

As can be seen, dark doors which frequently occur in the environment, have only a small range of angles for which valid measurements are expected. Similarly, the window front leads only to valid measurements when the angle of incidence is almost orthogonal to the surface.

B. Improving Global Localization

We recorded a second dataset which we used for the evaluation of our localization approach. A close to ground truth estimate of the robot poses is obtained by using the SICK laser scanner (and by registering the second dataset in the map build from the first dataset). The following localization experiments were carried out using only the data of the Hokuyo scanner.

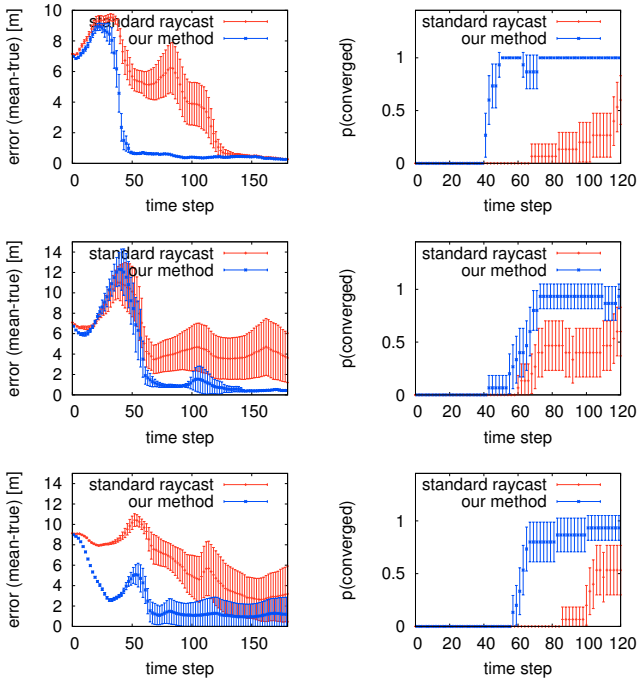


Fig. 6. Statistical evaluation of the global localization performance of our method vs. the standard ray-cast model for three different dataset (15 runs with different seeds). The left plots shows the error, computed as weighted mean error, vs. time and the right plots show the corresponding convergence probability vs. time given the 15 runs. As can be seen, our method significantly outperform the ray-cast model (the error bars illustrate the 95% confidence intervals).

We partitioned the dataset into three parts and evaluated the performance of our method in comparison to the standard ray-cast model. We performed 15 runs with different seeds for each dataset and evaluated the distance of the weighted mean of the particles to the true pose over time for each technique. One time step corresponds to the integration of an observation which is done after the robot traveled for at least $10\,cm$ or rotated by at least $10°$ according to odometry. The results are shown in the left column of Fig. 6. Furthermore, we computed the convergence probability of the filter over time by counting when more that 95% of the probability mass is inside a 1.5 m radius. The results are depicted in the right column of Fig. 6 and the error bars indicate the 95% confidence intervals. As these results demonstrate, our method that explicitly considers reflection properties of surfaces significantly outperforms the standard ray-cast model since *our filter converges significantly faster* (the task in global localization is to determine the robot's pose as fast as possible).

We performed a further experiment in which the robot was moving in an environment in which the walls are highly reflecting. As expected, no significant difference between the standard ray-cast model and our approach can be determined in this case (see Fig. 7).

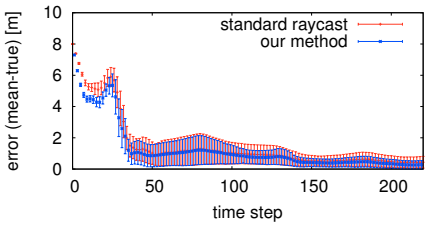


Fig. 7. Comparison of the pose error during global localization in an environment that contains only few surfaces with different reflection properties (multiple bright white posters have been added to the walls to ensure high reflectance). As expected, in this setting no significant difference between the ray-cast model and our new approach can be observed.

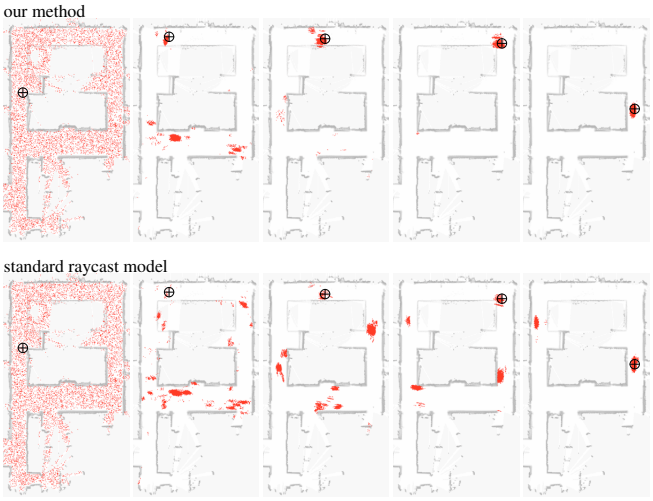


Fig. 8. Typical results obtained in a global localization experiment for our method (top row) and the standard ray-cast model (bottom row). The images depict the particle distribution at different time steps and the cross indicates the ground truth. The images correspond to one experiment of the statistical evaluation shown in the first row of Fig. 6. As can be seen, our method converges faster towards the true location of the robot. For illustration reasons, the maps depicted in the background of the images show the same reflection probability map of the environment.

The images in Fig. 8 visualize a typical evolution of the particles for one experiment. The first row depicts the resulting particle sets when applying our method, whereas the second row shows the particle distribution obtained with the standard ray-cast model. Again, it can be seen that our method converges faster towards the true pose of the robot. Since the standard ray-cast model does not consider different reflection properties of surfaces, particles at poses far away from the true location of the robot are more likely to survive.

VI. CONCLUSIONS

In this paper, we presented a novel technique to learn reflection properties of surfaces and to utilize this knowledge in probabilistic localization. Especially for low-cost and light-weight laser range scanners, which are frequently used with humanoids or small flying vehicles, the probability of the reflectance of a beam depends highly on the angle of incidence and on the distance of the scanner to the object. We therefore proposed to explicitly consider the reflection properties of surfaces during localization. Our approach extends the standard ray-cast observation model by incorporating the learned knowledge about reflection properties of objects.

As we demonstrate in experiments carried out with a wheeled robot, we can significantly speed-up global localization in comparison to the standard ray-cast model. Our method converges faster to the true pose of the robot and substantially reduces the error in the estimated pose.

Given these encouraging results, it can be presumed that utilizing reflection properties of surfaces can also improve solutions to the simultaneous localization and mapping problem using data of such low-cost sensors. One possibility, for example, is to incorporate the reflection properties in the scan-matching routine underlying existing SLAM methods such as [18] to obtain better proposal distributions which in turn leads to more efficient algorithms.

REFERENCES

[1] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan, "A layered architecture for office delivery robots," in *Proc. of the First Int. Conf. on Autonomous Agents*, 1997.

[2] R. Siegwart, K. Arras, B. Jensen, R. Philippsen, and N. Tomatis, "Design, implementation and exploitation of a new fully autonomous tour guide robot," in *Proc. of the 1st Int. Workshop on Advances in Service Robotics (ASER)*, 2003.

[3] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma, "Experiences with a mobile robotic guide for the elderly," in *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2002.

[4] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[5] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[6] K. Konolige and K. Chou, "Markov localization using correlation," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1999.

[7] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *Int. Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[8] K. Arras, R. Philippsen, N. Tomatis, M. de Battista, M. Schilt, and R. Siegwart, "A navigation framework for multiple mobile robots and its application at the Expo.02 exhibition," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[9] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, 2001.

[10] S. Thompson, S. Kagami, and K. Nishiwaki, "Localisation for autonomous humanoid navigation," in *Proc. of IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids)*, 2006.

[11] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[12] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 2005.

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[14] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2000.

[15] B. Schiele and J. L. Crowley, "A comparison of position estimation techniques using occupancy grids," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1994.

[16] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1985.

[17] J. Lim and D. Cho, "Physically based sensor modeling for a sonar map in a specularenvironment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1992.

[18] C. Stachniss, G. Grisetti, W. Burgard, and N. Roy, "Evaluation of Gaussian proposal distributions for mapping with Rao-Blackwellized particle filters," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[C9] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

Imitation Learning with Generalized Task Descriptions

Clemens Eppner Jürgen Sturm Maren Bennewitz Cyrill Stachniss Wolfram Burgard

***Abstract*— In this paper, we present an approach that allows a robot to observe, generalize, and reproduce tasks observed from multiple demonstrations. Motion capture data is recorded in which a human instructor manipulates a set of objects. In our approach, we learn relations between body parts of the demonstrator and objects in the scene. These relations result in a generalized task description. The problem of learning and reproducing human actions is formulated using a dynamic Bayesian network (DBN). The posteriors corresponding to the nodes of the DBN are estimated by observing objects in the scene and body parts of the demonstrator. To reproduce a task, we seek for the maximum-likelihood action sequence according to the DBN. We additionally show how further constraints can be incorporated online, for example, to robustly deal with unforeseen obstacles. Experiments carried out with a real 6-DoF robotic manipulator as well as in simulation show that our approach enables a robot to reproduce a task carried out by a human demonstrator. Our approach yields a high degree of generalization illustrated by performing a pick-and-place and a whiteboard cleaning task.**

I. INTRODUCTION

Several techniques exist for transferring new skills to robots. A very promising technique is called "imitation learning": Here, a robotic system observes an instructor while performing a task [4], [2]. From multiple demonstrations, the robot then needs to infer a generalized task description and reproduce it accordingly even under slightly modified conditions. As an example, consider Fig. 1. The task of cleaning a whiteboard is demonstrated by a human and transferred to a simulated as well as to a real robotic manipulator.

Teaching skills by direct demonstration is a very natural way of skill transfer in humans and animals. In the aim to create more versatile, adaptable, and sociable robotic platforms, research on the mechanisms of learning new behaviors by observation has a very high potential. Furthermore, such social learning can speed up learning complex behaviors enormously, as it provides strong prior information for the learning process, thereby scaling back the learning task for the robot. This can reduce the search space for traditional learning algorithms significantly, such that previously intractable tasks can be learned.

In this paper, we show that imitation learning is well suited as a user-friendly instruction method for manipulation tasks. Our approach uses motion capture data generated by a vision system to track body parts of the instructor and the 3D positions of relevant objects in the scene. The body configuration as well as the relations between objects and body parts of the demonstrator are in turn modeled as normally distributed observation nodes of a dynamic Bayesian network (DBN). For reproducing an observed skill, the network is evaluated at each time step in order to infer the most-likely action. Within our framework, new constraints can be dynamically added to the network, e.g., to incorporate collision avoidance during reproduction in order to deal with unforeseen obstacles.

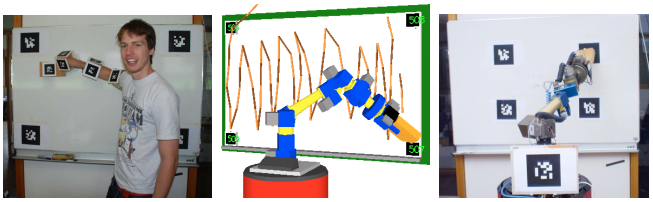


Fig. 1. Learning the whiteboard cleaning task. Left: The human instructor demonstrates how to clean the whiteboard using visual motion capturing. Middle: After the first demonstration, the robot can replay the recorded trajectory. Right: After several demonstrations, the robot can generalize the task and reproduce it under changed conditions, for example, on a whiteboard with different size and position.

Our relation-based approach extends the recent work of Calinon and Billard [6] by formalizing the problem by means of a DBN. We furthermore allow for incorporating additional constraints for modeling unexpected obstacles that should be considered during imitation.

The remainder of the paper is organized as follows: We briefly review related work on imitation learning in Sec. II. Our framework on imitation learning via inference in a DBN is introduced in Sec. III, followed by Sec. IV on learning the parameters of the DBN from motion capture data and Sec. V on reproducing the generalized skill. Experimental results obtained with a real 6-DoF robotic manipulator as well as in simulation are presented in Sec. VI.

II. RELATED WORK

In the past, various techniques have been used for transferring task knowledge to a robotic system. Initially, the required motion trajectories were hand-coded by an engineer [10], [19]. However, the more complex the task description becomes, the more difficult it is to create and maintain large controllers. Alternatively, the required joint angle trajectories of the robot can be shown by a single demonstration, for example, by a human teacher using a joystick, a motion capturing system, or kinesthetic training. The resulting sequence is recorded and can then be replayed by the robot. However, if the observations are noisy or

This work was partly supported by the DFG under contract number SFB/TR-8 as well as by the EC under contract number FP6-IST-045388-INDIGO

All authors are with the Computer Science Department of the University of Freiburg, Germany
{eppner, sturm, maren, stachnis, burgard}@informatik.uni-freiburg.de

unpredicted disturbances in the task environment occur, simple playback of the recorded motion is in general not sufficient to reliably reproduce a given task. To deal with noise in the observations and to generalize over multiple demonstrations of the same tasks, several authors suggested hidden Markov models (HMM) to encode and reproduce a demonstrated action, e.g., [1], [7], [20].

Reinforcement learning techniques have been successfully applied to learn controllers for an individual skill or for motor primitives [12], [3], [17]. As the size of the search space grows exponentially with the dimensionality of the learning problem, Ijspeert and Schaal [13], [18] proposed to learn parameterized controllers instead that are based on differential equations.

Pardowitz and Dillmann presented a system that generalizes over household tasks in a hierarchical manner [16]. Actions performed by the human demonstrator are recognized as a sequence of "elementary operators", of which a graph-based task representation is learned. In this approach, the incrementally updated network topology reflects the learned temporal ordering of the individual actions.

Although symbolic representations are well suited for planning and reasoning, their limitation to higher-level skills renders them inapplicable in domains where a continuous motor control is required. By contrast, trajectory learning directly starts by encoding each demonstration by a sequence of continuous observations. Due to the massive amounts of captured data, dimensionality reduction techniques are often applied. Chalodhorn *et al.* [8] used principal component analysis (PCA) to reduce the high-dimensional motion capture data of a recorded human walk. While a direct playback of the human data on a humanoid robot would make the robot fall, the authors showed that after a few trials the robot was able to modify the imitated gait incrementally. There, a sensory-motor predictor was learned and used to produce dynamically stable actions. Similarly, Grimes *et al.* [11] also used PCA to reduce the high-dimensional configuration space and applied a DBN to infer dynamically stable imitative actions using constraint variables and a learned forward model of the robot dynamics.

In our approach, we also use a DBN to learn and reproduce tasks. By means of the DBN, relations between objects in the environments and body parts are learned and considered during reproduction. We show that by using this framework, it becomes also possible to incorporate additional constraints (such as collision avoidance) during skill reproduction.

We use the idea presented by Calinon and Billard [6] to describe actions using relations between objects. In contrast to their approach which is based on Gaussian mixture models, we use a kernel estimator to model the relations. This way, we can deal with a small number of demonstrations and avoid fitting Gaussian mixtures.

III. IMITATION LEARNING FRAMEWORK

As most techniques for generating imitative actions, our approach uses two steps. First, a set of repeatedly carried-out actions of a human demonstrator is observed by the robot. The robot has to infer the relevant parts of the demonstrated task and to build an internal representation. This is done in the so-called learning step. Second, the robot must be able to reproduce an action to actually imitate the demonstrator. This step is called imitation or reproduction step.

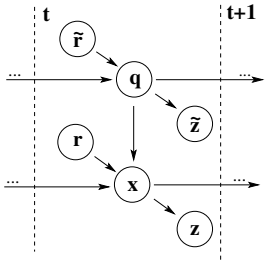


Fig. 2. Dynamic Bayesian network (DBN) illustrating the conditional independence assumptions used for learning and reproducing tasks. The arrows indicate conditional dependencies between variables. Here, $\tilde{z}$ denotes the observation of the demonstrators body configuration which is represented by q (joint angles). z is the observation of the world state x that encodes the position of relevant object in the scene. The object-manipulator relations r as well as the constraints on joint angles $\tilde{r}$ are used to model the action that should be learned and carried out by the robot. During learning, the distributions over the relations $p(\tilde{r})$ and $p(r)$ is determined. During reproduction, these relations are known and the most likely body configuration of the robot is to be estimated and executed by the robot.

From a more formal point of view, we treat the problems as a stochastic process that can be described by means of the dynamic Bayesian network depicted in Fig. 2. The DBN is an intuitive graphical description of the conditional independence assumptions made in the model. While the teacher demonstrates the task several times during the first phase of our imitation learning approach, we learn relevant relations, also called constraints, between n objects in the world and the manipulator as well as the joint angle constraints for each point in time. Thus, the constraints encode the actions necessary to carry out the task which is being demonstrated. The corresponding latent variables are denoted $r(t) \in \mathbb{R}^{3n}$ – the constraints between objects in world space, encoded as the 3D displacement from end-effector to each object – and $\tilde{r}(t) \in \mathbb{R}^m$ the constraints in configuration space – i.e., the joint angles of the m-DoF manipulator – respectively.

The relations $\tilde{r}(t)$ regarding the joint angles are only meaningful if the demonstrator and the imitator have a similar body structure. They are important in case actions should be imitated as precise as possible.

In the second phase, we use the learned relations to infer actions of the robot. At this time, the relation variables are observable variables in the DBN and generate the actions of the robot given the estimated world state.

For reasons of simplicity, let us consider first the DBN for one time step t (neglecting the index t). Let $q \in \mathbb{R}^m$ refer to the configuration of the m-DoF arm of the demonstrator (during learning) or the robot (during reproduction). Let $x \in \mathbb{R}^{3n+3}$ be the vector of the 3D positions of relevant objects in the scene

$$x \;=\; \{x_E, x_1, \ldots, x_n\}\,, \tag{1}$$

where $x_1, \ldots, x_n$ are the positions of the n objects in

the scene. In the remainder of the paper, we use a robot manipulator for imitating humanoid arm movements and x_E is the position of the end effector. Note that any set of body parts can be used in case full body actions should be imitated without changing the math except for adding additional variables and indices (x_E would become $x_{E_1}, \ldots, x_{E_M}$).

The observation of 3D poses x is called $z \in \mathbb{R}^{3n+3}$, and the observations of joint configurations $q \in \mathbb{R}^m$ will be referred to as $\tilde{z} \in \mathbb{R}^m$.

Our DBN depicted in Fig. 2 implies the following independence assumptions:

$$\begin{aligned} p(\tilde{r}, q, \tilde{z}, r, x, z) &= p(z \mid x) \cdot p(\tilde{z} \mid q) \cdot p(\tilde{r}) \cdot p(r) \cdot \\ &\quad p(x \mid q, r) \cdot p(q \mid \tilde{r}) \end{aligned} \tag{2}$$

In our model, we assume the following distributions in the nodes of the DBN: The observation models $p(z \mid x)$ and $p(\tilde{z} \mid q)$ are assumed to be Gaussian distributions and so are the distributions over the relations $p(r)$ and $p(\tilde{r})$. Since we have no information about the distributions over relations in the beginning, we set their variance to infinity.

The posterior about the objects in the scene can be written as follows:

$$p(x \mid q, r) = p(x_E \mid q) \cdot p(x_1, \ldots x_n \mid x_E, r) \tag{3}$$

Eq. (3) is obtained by applying the product rule and by assuming that the poses of objects are independent from the joint configuration given the position of the end effector. By further applying the product rule, we obtain:

$$\begin{aligned} p(x \mid q, r) &= p(x_E \mid q) \cdot p(x_1 \mid x_E, r) \cdot \\ &\quad p(x_2, \ldots x_n \mid x_E, r) && (4) \\ &= p(x_E \mid q) \cdot \Pi_{i=1}^{n} p(x_i \mid x_E, r_i) && (5) \\ &\approx p(x_E \mid q) \cdot \Pi_{i=1}^{n} \mathcal{N}_{r_i}(x_i - x_E; \Sigma_i) && (6) \end{aligned}$$

Eq. (4) is obtained by assuming that given the relations r as well as x_E, the positions of two objects in the scene are independent. By applying this assumption n times, we obtain Eq. (5).

We additionally assume that also the posterior about the pose of the end effector $p(x_E \mid q)$ is Gaussian, which is a reasonable assumption for the robots equipped with accurate joints (such as Schunk modules in our case). $p(x_E \mid q)$ then corresponds to the kinematic function.

Finally, we make the assumption that the differences between actions carried out during the individual demonstrations can be described by Gaussian distributions. Thus, the individual $r_i \in r$ are represented by a mean and a variance for the three dimensions. This leads to Eq. (6). Similarly, the individual joint constraints $r_j \in \tilde{r}$ are represented by Gaussians. Thus, $p(q \mid \tilde{r})$ can also be computed by a product as in Eq. (6).

IV. LEARNING PHASE

In the first phase, the robot observes a person that repeatedly carries out the task the robot has to perform. Given the DBN structure explained above, the key challenge of this learning phase is to learn the object-manipulator relations (r) and – if needed – the joint constraints ($\tilde{r}$).

A. Motion Capturing and Object Pose Estimation

To estimate the motion trajectories of the human demonstrator while executing a task and the 3D position of relevant objects in the scene, we use passive markers and images of a monocular camera. In particular, we use the ARToolKit [14], which is a software library providing the means to extract the 6D pose (orientations and position) of fiducial markers (black squares on a white background) from single camera images.

We attach compounds of 4 markers around the teacher's arm (see left image of Fig. 1) to bypass the problem of occlusions. In most cases, not more than one marker of the same compound is visible. To deal with the case that two markers are visible simultaneously (sensing more markers at the same time is impossible due to their mutual orthogonality within one compound), we perform a linear interpolation between their poses depending on the degree of visibility of the markers. Finally, we apply a Kalman filter to track the 3D marker position estimates over time. To derive the demonstrator's joint angles from marker poses, we use an anthropomorphic arm model and apply straightforward geometric operations. As a result, we are able to reliably estimate the probability distributions over q and x during the demonstrations.

B. Multiple Demonstrations

Our approach relies on demonstrating the same task multiple times in order to achieve a good generalization. One problem when generalizing task descriptions from multiple demonstrations is the fact that the repeatedly observed actions are not time-synchronized, even though the different demonstrations typically do not vary largely. To deal with varying movement velocity profiles, we apply derivative dynamic time warping (DDTW) [15] which is able to account for local distortions in the time domain by computing a nonlinear transformation of the time axis of the individual demonstrations. Based on the aligned demonstrations, we can derive the relations r and $\tilde{r}$ which encode the action to imitate.

C. Deriving Relations for Generalized Task Descriptions

By assuming that all relations in r and $\tilde{r}$ are (individual) Gaussians, we can directly infer a mean and a variance estimate for the individual relations for each point in time given the estimates for x and q. Especially the variance is a key element for the tasks descriptions since it describes how accurately the demonstrator enforced this relation during the demonstrations.

Formally, a relation r_i is fully described by a 3D mean μ_i and 3D variance σ_i^2 vector (assuming the relations in the three dimensions to be independent).

In theory, we could compute μ_i and σ_i^2 directly from the estimates for x and q during learning. In practice, however, we typically have to deal with a rather small number of demonstrations and therefore rather rough and non-smooth estimates are obtained if the values are computed directly. To overcome this problem, we apply a Parzen window kernel estimator for computing smooth function approximations.

This is a non-parametric technique that allows for estimating a function μ based on a set of sample points. To compute estimates of the relations between objects in the scene r_i and the end effector, we compute relation samples consisting of time and position $\{t, l(t,d,i)\}$ with $l(t,d,i) = x_i^d(t) - x_E^d(t)$, $t = 1, \ldots, T$ and $d = 1, \ldots, D$ where D are the number of demonstrations. Then, we obtain

$$\mu_i(t') = \frac{\sum_{d=1}^{D}\sum_{i=1}^{T} K(\frac{t'-t}{h})\; l(t,d,i)}{\sum_{d=1}^{D}\sum_{i=1}^{T} K(\frac{t'-t}{h})}, \tag{7}$$

where h is the Parzen window size (empirically determined, $h = 0.2\,s$) and K is a kernel function. We use the standard choice for the K, namely the squared exponential kernel

$$K(x) = \exp\left(-\frac{1}{2}||x||^2\right). \tag{8}$$

The variance given the sample points can be estimated similarly as

$$\sigma_i^2(t') = \frac{\sum_{d=1}^{D}\sum_{i=1}^{T} K(\frac{t'-t}{h})\; (l(t,d,i) - \mu(t))^2}{\sum_{d=1}^{D}\sum_{i=1}^{T} K(\frac{t'-t}{h})}. \tag{9}$$

This procedure is carried for each object in the scene and accordingly done for the joint relations $\tilde{r}$.

V. REPRODUCTION PHASE

The goal of the reproduction or imitation phase is to carry out the demonstrated task to achieve the same result. Given the DBN, we can seek for the configuration of joints q^* that maximizes the likelihood given the demonstrations.

A. Incremental Optimization

If we consider only one time step during the optimization, we seek for the configuration q^* that maximizes the joint probability. During reproduction, r and $\tilde{r}$ are known. Furthermore, the robot can control its body/manipulator by specifying a joint configuration and does not have to rely on noisy marker observations of its body. Therefore, the maximization turns into

$$q^* = \operatorname*{argmax}_q p(q \mid \tilde{r}) p(x \mid r, q) p(z \mid x). \tag{10}$$

As discussed before in Sec. III, the posteriors $p(q \mid \tilde{r})$, $p(x \mid r, q)$, and $p(z \mid x)$ are basically products of Gaussians and lead to a Gaussian distribution again. Thus, to maximize the joint probability, we need to determine the mean of that Gaussian. To do so, we proceed as follows. Consider that we are currently at time step t and want to seek for the joint configuration that maximizes Eq. (10) at $t+1$. Each relation between x_i and x_E generates a relative displacement vector Δ_i:

$$\Delta_i(t+1) = x_i(t) - x_E(t) - \mu_i(t+1) \tag{11}$$

Since we want to compute a new joint configuration for the robot, we need to convert the constraints expressed in world coordinates in joint space. We achieve this by applying a variant of the damped-least squares method described by Buss [5]. This approximative technique performs a linearization of the kinematic function. According to this method, a desired movement in world coordinates (Δ) is transformed to an executable movement in joint space ($\tilde{\Delta}$) by

$$\tilde{\Delta}_i(t+1) = J\left(JJ^T + \lambda^2 I\right)^{-1} \Delta_i(t+1) \tag{12}$$

$$\tilde{\Sigma}_i(t+1) = \left(J\left(JJ^T + \lambda^2 I\right)^{-1}\right) \cdot \Sigma_i(t+1) \left(J\left(JJ^T + \lambda^2 I\right)^{-1}\right)^T \tag{13}$$

where λ is the so-called damping factor and J refers to the Jacobian. $\Sigma_i(t+1)$ is a 3×3 matrix encoding the variances from the relation r_i with the corresponding variances for dimension x, y, and z on the diagonal. Due to the linear mapping, we obtain also a Gaussian in configuration space.

The constraints defined by $\tilde{r}(t+1)$ can be easily used to compute a desired movement

$$\tilde{\Delta}_{\tilde{r}}(t+1) = \tilde{r}(t+1) - q_t \tag{14}$$

while the variances $\tilde{\Sigma}(t+1)$ do not need to be transformed.

All constraints resulting from the observation of the demonstrator's joint angle configurations or from the arrangement of objects in the scene are expressed in terms of updates in joint space. This allows us to combine them by multiplying the normal distributions as it has also been done by Calinon and Billard [6]. The resulting distribution is the product over the $N+1$ Gaussians resulting from the N task space relations plus the joint space relations. We can use it to directly obtain the next configuration $q^*(t+1)$ according to Eq. (10) by selecting the mean from this combined Gaussian, as given by

$$\hat{q}(t+1) = q_t + \tilde{\Sigma}(t+1)\Bigg((\tilde{\Sigma}_{\tilde{r}}(t+1))^{-1}\tilde{\Delta}_{\tilde{r}}(t+1) + \sum_{i=1}^{n}(\tilde{\Sigma}_i(t+1))^{-1}\tilde{\Delta}_i(t+1)\Bigg), \tag{15}$$

with

$$\hat{\Sigma}(t+1) = \left(\sum_{i=1}^{n}(\tilde{\Sigma}_i(t+1))^{-1} + (\tilde{\Sigma}_{\tilde{r}}(t+1))^{-1}\right)^{-1} \tag{16}$$

The mean of that distribution defines the configuration of the robot at the next time step that maximizes the probability distribution specified in Eq. (10).

B. Local Optimization with Obstacles

The technique described in the previous section can directly be applied to deal with unforeseen obstacles in the scene during reproduction. Consider that the robot observes an obstacle during the imitation that was not there during the demonstrations. To avoid this obstacle during the reproduction task, we can add additional constraints between the observed obstacle and the closest point on the robot's body, as used in approaches based on potential fields for collision avoidance.

Without changing the framework described above, the robot can reactively introduce constraints for avoiding obstacles while carrying out its task as close a possible to the human demonstrator. Let x_O be the position of the obstacle. Instead of adding a repellent force, we add an attractor at the opposite side of the end-effector,

$$\Delta_O = -\alpha \frac{x_E - x_O}{\|x_E - x_O\|} \quad (17)$$

$$\Sigma_O = \beta \cdot \exp\left(\|x_E - x_O\|^2\right) \cdot I, \quad (18)$$

where α determines the desired distance to the obstacle and β gives the relative importance with respect to the other constraints.

It should be noted that this technique works well for small or rather simple structured obstacles added to the scene. In case complex or, for example, U-shaped obstacles are found in the environment, this approach is likely to suffer from local minima caused by contradictory constraints.

C. Global Optimization

The problem of local minima, however, can be avoided by not incrementally optimizing the joint probability distribution of the DBN for the upcoming time step but optimizing it over all time steps $1 \dots T$ of the task sequence at once:

$$q^*_{1:T} = \underset{q_{1:T}}{\operatorname{argmax}}\, p(q_{1:T}, x_{1:T}, \tilde{r}_{1:T}, r_{1:T}, z_{1:t}) \quad (19)$$

Note that at a particular time step t, only the first $1 \dots t$ observations $z_{1:t}$ are already available and can be included for planning. Doing this optimization on a global level, however, comes with significantly increased computational cost due to the high dimensionality of $q^*_{1:T}$.

One way of rather efficiently estimating $q^*_{1:T}$ is to make use of probabilistic roadmaps or rapidly-exploring random trees (RRTs) [9], and find the shortest path using A^* on the sampled set of nodes since we are only interested in the most-likely imitation sequence. Given that we properly encode the likelihoods of all constraints in the cost function later used by A^*, the solution of the planner will approximate Eq. (19) well.

We propose to base the cost function on the Mahalanobis distance to the combined Gaussian $\mathcal{N}(\hat{q}(t); \hat{\Sigma}(t+1))$ computed in (15) and (16), as this Gaussian already incorporates all constraints r, $\tilde{r}$ and the obstacle constraints in a time-dependent way. For a configuration q at time t, we define the cost as the likelihood with respect to the previously computed combined Gaussian, i.e.,

$$cost(q,t) = (q - \hat{q}(t))^T \hat{\Sigma}(t+1)^{-1} (q - \hat{q}(t)) \,. \quad (20)$$

Then, finding the cost-optimal sequence of configurations $q^*_{1:T}$ is equivalent to maximizing the likelihood of the trajectory $q^*_{1:T}$ in (19).

VI. EXPERIMENTS

We carried out a set of experiments to analyze our approach. We always observed a human demonstrator equipped with markers of the ARToolKit, see Fig. 3. We used this data to learn the relations and thus constraints for the task reproduction online. The motions were sampled at a rate of 5 Hz. We segmented the training trajectories manually.

Fig. 3. Photos of the demonstrated pick-and-place task. The trajectory is recorded both, in task and joint space.

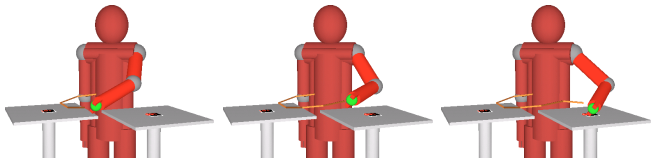


Fig. 4. Reproduction of the pick-and-place task by a human-like manipulator using both, task and joint space constraints.

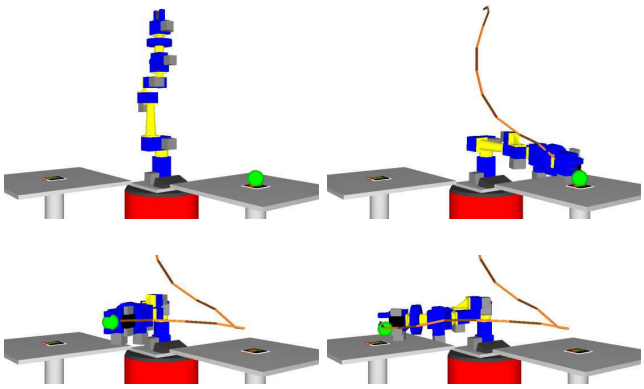


Fig. 5. Reproduction of the same pick-and-place task by our 6-DoF manipulator. Note that the robot successfully generalized the task, as the target and source location have been swapped.

A. Imitating Human Actions

To imitate the observed behavior, we reproduced the tasks using a real robot equipped with a manipulator and two simulated robots, one with a manipulator and one with a human-like arm.

Fig. 4 shows the reproduction of the pick-and-place task after being demonstrated four times (see Fig. 3). The human-like simulated robot considers both, the joint and the task constraints, which leads to the fluent, human-like movement.

In Fig. 5, the same task is reproduced by our robotic manipulator. As the demonstrator and the imitator have a significantly different body scheme, the joint constraints $\tilde{r}$ have been disabled. As can be seen, the robot is able to reproduce the task even though the setup between demonstration and imitation has been changed by setting a different target location.

We furthermore analyzed the number of demonstrations needed until a task could be reproduced reliable. For this analysis, a teacher picked up a cup and placed it at a distance of 1 m. As can be seen in Fig. 6, our approach converged after 4 iterations.

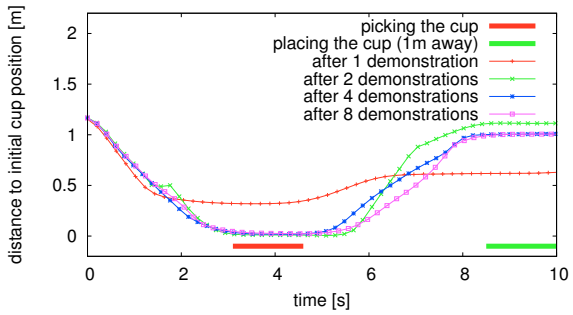


Fig. 6. Illustration how the approach converges to an action after different number of demonstrations. The demonstrated task was to pick up a cup and place it 1 m away. As can be seen, after 4 demonstrations, our approach converged and the robot was able to pick and place the cup.

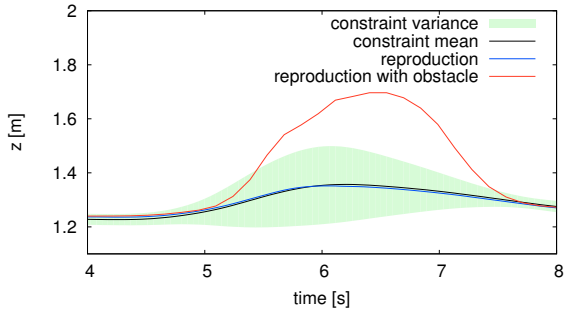


Fig. 7. Illustration of a constraint including the variance as well as two trajectories of reproductions – one for the obstacle free case and one in case an obstacle blocks the trajectory.

B. Dealing with Obstacles during Imitation

The additional obstacle constraints described in Sec. V-B allow the robot to deal with unforeseen obstacles during the task execution. The obstacle constraints act similar to a potential field pushing the robot away from obstacles, which are labeled by predefined markers.

Fig. 7 illustrates an example for a constraint in a pick and place task. The figure shows a reproduced trajectory for the obstacle free case and a trajectory that was selected in the presence of an obstacle. As can be seen, the robot moved its arm over the obstacle in order to avoid a collision.

We carried out a whiteboard cleaning task that nicely illustrates the properties of the presented methods. First, a human repeatedly cleaned a whiteboard in an area bounded by 4 markers with the same number of ups and downs (see left image of Fig. 1). Then, we attached a sponge to the robot and let it perform the demonstrated task. In the first experiment, we modified the size of the area to clean for illustrating the capabilities of generalization. Photos from this experiment can be seen in Fig. 8. Note that in case the area to clean is much larger than during learning, the whiteboard may not be cleaned well. The reason for that is that our approach then scales the trajectory to reproduce and thus there might be parts that will not be covered by the sponge.

In the second experiment, we showed the robot during reproduction phase an obstacle marker that was not there

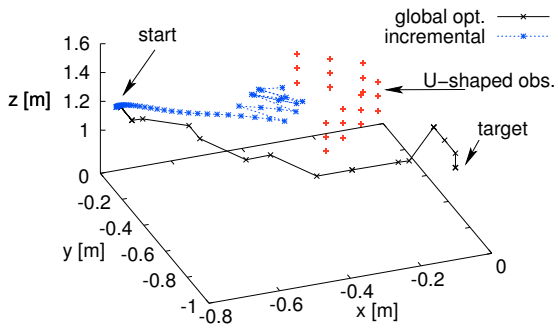


Fig. 10. The plot shows the end effector position of the robot over time during two experiments. When applying the incremental method, the end effector gets stuck in a U-shaped obstacle while the global method solves the task and the end effector reaches the target location.

during the learning phase, see first image of Fig. 9. Then, the robot had to clean the whiteboard while avoiding obstacles (and thus not cleaning the area of the marker). For reasons of illustration, we removed the marker during the experiment but kept it in the internal memory of the robot. In this way, the reader can see that the robot did not clean the corresponding area. Eight photos were taken during the reproduction and are depicted in Fig. 9. To avoid the area marked as an obstacle area, the robot lifts the sponge away from the whiteboard (in the direction of the observing camera).

C. Imitation by Planning

The experiments presented above used the reproduction by means of incrementally computing the maximum-likelihood configuration of the DBN. This is can be done efficiently online, but the approach can suffer from local minima, for example, in the presence of U-shaped obstacles. Such an example is presented in Fig. 10 where the robot gets stuck.

If one applies the global optimization technique described in Sec. V-C, one can overcome this problem since the optimal solution over all time steps is computed. Thus, the robot is able to reproduce the task including the avoidance of the U-shaped obstacle. This global method, however, comes with a significantly increased computational load.

VII. CONCLUSION

In this paper, we presented an approach to imitation learning that extends a recently published method of Calinon and Billard. It enables a robot to observe, generalize, and reproduce tasks from observing a human demonstrator. We formalized the problem using a dynamic Bayesian network that is used for learning relations between the observed positions of the objects and body parts of the instructor. Additional constraints, for example, to avoid unforeseen obstacles can be added online. To imitate the action of a human, we estimate the actions that maximize the joint probability distribution represented by the DBN. We evaluated the approach and showed that a real robot equipped with a manipulator can learn and reproduce demonstrated actions. Based on a pick-and-place and a whiteboard cleaning task,

Fig. 8. The reproduction of the board cleaning task by our robot. It imitates the zig-zag movement for cleaning the board with the sponge. Note that the learned task representation allows for cleaning differently sized surfaces based on the markers.

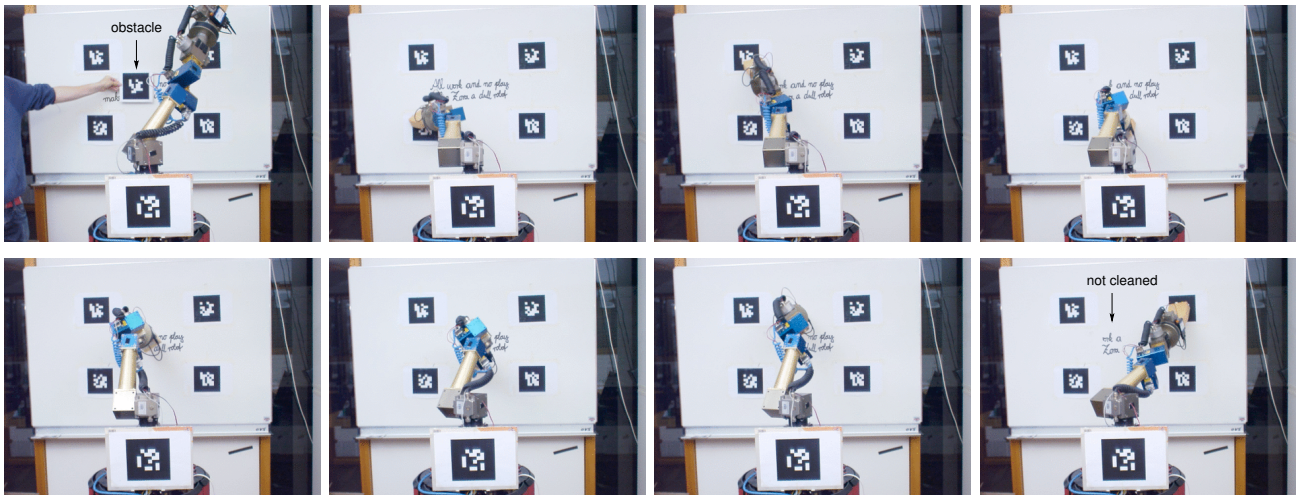


Fig. 9. The board cleaning task with an obstacle reproduced by our 6-DoF manipulator. In the first frame, the position of the obstacle is shown to the system via a marker. Next, the manipulator cleans the whiteboard similar to the previous experiment shown in Fig. 8 but additionally avoids the obstacle. In our current implementation, the obstacle is supposed to have the size and position of the corresponding marker that can be perceived using the ARToolkit system. As can be seen in the last frame, part of the text in the area of the obstacle marker was not wiped.

we illustrated the flexibility of the method to generalize over different spatial setups.

References

[1] T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2006.

[2] P. Bakker and Y. Kuniyoshi, "Robot see, robot do: An overview of robot imitation," in *In AISB96 Workshop on Learning in Robots and Animals*, 1996, pp. 3–11.

[3] D. Bentivegna, C. G. Atkeson, and G. Cheng, "Learning tasks from observation and practice," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 163–169, 2004.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, in press.

[5] S. R. Buss and J. su Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics Tools*, vol. 10, pp. 37–49, 2004.

[6] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling skill constraints in joint space and task space," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, September 2008.

[7] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[8] R. Chalodhorn, D. B. Grimes, and R. P. N. Rao, "Learning to walk through imitation," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. San Mateo, CA: Morgan Kaufmann, 2007.

[9] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[10] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[11] D. Grimes, R. Chalodhorn, and R. Rao, "Dynamic imitation in a humanoid robot through nonparametric probabilistic inference," in *Proc. of Robotics: Science and Systems (RSS)*, 2006.

[12] R. Hafner and M. Riedmiller, "Neural reinforcement learning controllers for a real robot application," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007, pp. 2098–2103.

[13] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *NIPS*, 2002, pp. 1523–1530.

[14] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. of the 2nd Int. Workshop on Augmented Reality (IWAR)*, 1999.

[15] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *In Proc. of the 1st SIAM Int. Conf. on Data Mining (SDM)*, 2001, pp. 5–7.

[16] M. Pardowitz and R. Dillmann, "Towards life-long learning in household robots: The piagetian approach," in *Proc. 6th IEEE International Conference on Development and Learning, London, UK*, 2007.

[17] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2003.

[18] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning Movement Primitives," in *International Symposium on Robotics Research (ISRR2003)*, 2003.

[19] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. Springer, January 2000.

[20] S. Tso and K. Liu, "Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories," in *Proceedings of The IEEE International Conference on Industrial Technology (ICIT)*, 1996.

[C10] H. Kretzschmar, C. Stachniss, C. Plagemann, and W. Burgard. Estimating landmark locations from geo-referenced photographs. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

Estimating Landmark Locations from Geo-Referenced Photographs

Henrik Kretzschmar Cyrill Stachniss Christian Plagemann Wolfram Burgard

***Abstract*— The problem of estimating the positions of landmarks using a mobile robot equipped with a camera has intensively been studied in the past. In this paper, we consider a variant of this problem in which the robot should estimate the locations of observed landmarks based on a sparse set of geo-referenced images for which no heading information is available. Sources for such kind of data are image portals such as Flickr or Google Image Search. We formulate the problem of estimating the landmark locations as an optimization problem and show that it is possible to accurately localize the landmarks in real world settings.**

I. Introduction

Popular Internet resources such as Flickr or Google Image Search offer a large amount of real world imagery. Many of these images contain geo-references, i.e., the locations where the photographs have been taken in longitude and latitude coordinates as well as manual annotations such as marked image regions and a tag word like "cathedral". Currently, Flickr offers millions of tagged images, a trend which is likely to continue given the growing popularity of mobile devices, GPS receivers and specialized integrated systems. The question of how this large amount of freely available data can be used to infer quantitative knowledge about the world was our main motivation for this work. On a comparably small spatial scale, systems such as Microsoft's Photo Tourism [13] process sets of images of the same location to yield a dense 3D model of the local environment, which is capable of producing artificial views and virtual fly-throughs. In this paper, we deal with the problem of localizing a discrete set of distinct landmarks on a larger spatial scale, like a town or a campus environment. Concretely, our task can be formulated as follows. Given a set of geo-referenced photographs of an environment annotated with labels for distinct landmarks, how can we recover the locations of the landmarks in the world? Aside from the noisy geo-reference coordinates and inaccurate label placements, the main difficulty lies in the missing information about the camera headings.

A robot that is able to utilize a so far unused source of information offers new ways for building models of places it has not observed directly. It furthermore allows a robot to also refine or annotate exiting models. Consider, for example, a mobile tour guide robot deployed to a city center or to an archaeological site. Given the localized landmarks and the corresponding imagery, the system could offer a large range of location-dependent information without requiring a human expert to collect and formalize this knowledge.

The authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany. {kretzsch, stachnis, plagem, burgard}@informatik.uni-freiburg.de

Fig. 1. The goal of the work presented in this paper is to estimate the locations of the set of distinct buildings (here enclosed by rectangles) using geo-referenced photographs taken while walking through the city.

In this paper, we consider the problem of estimating the positions of landmarks given a set of geo-referenced photographs. The longitude and latitude information of the locations from which the photos have been taken are assumed to be known approximatively by means of a standard consumer GPS device. By combining this data with labeled regions in the photos referring to objects, such as buildings, our approach is able to localize these buildings and to determine the direction in which the photo has been taken. In contrast to bearing-only SLAM, our approach does not require an image stream from a camera. We furthermore assume to have no knowledge about the orientation of the camera at any point in time. We address this problem by formulating it as an optimization problem. As we show in the experiments, we are able to accurately localize the labeled buildings based on photos taken in an urban environment.

Figure 1 depicts the downtown area of Freiburg including several distinct landmarks enclosed by rectangles. The goal of this work is to estimate the positions of such landmarks based on photographs taken while walking or moving through the city center.

The remainder of this paper is organized as follows. After discussing related work, we present a mathematical formulation of our problem and derive an objective function that needs to be minimized to solve the estimation problem. In Section IV, we then explain two optimization procedures that are used throughout this work. Finally, we present real world as well as simulation experiments that illustrate the performance of our method.

II. Related Work

The problem of estimating the poses of landmarks based on observations has been intensively studied in the past. Several researchers studied the landmark-based simultaneous localization and mapping (SLAM) problem [1], [9], [8], [12], [4], [7]. In the literature, one distinguishes between SLAM approaches that are designed to operate on proximity sensors and those operating on bearing-only (typically vision) sensors. For example, Davison *et al.* [2] presented a vision-based 6 DoF SLAM system that extracts features from a monocular camera and creates a sparse map of high-quality stable features. The locations of the features are tracked by applying an EKF. Lemaire *et al.* [6] focused on the problem of how to initialize landmarks in the context of EKFs and bearing-only observations.

Approaches to bearing-only SLAM, however, consider that the robot constantly perceives the environment with its camera. As a result, a continuous stream of images and thus feature observations is provided so that the landmarks can be tracked over time. Furthermore, the robot is often assumed to roughly know the relative change in its orientation while moving from odometry. These two assumptions are not made in the work presented in this paper.

Recently, related approaches to this paper have been presented. The most prominent one is probably Microsoft's Photo Tourism [13]. This approach considers a set of images of the same place and generates a dense 3D model of the local environment. It is capable of producing artificial views and virtual fly-throughs. In contrast to this, we deal with the problem of localizing a discrete set of distinct landmarks on a larger spatial scale, for example in a town or a campus environment.

The MIT City Scanning Project [14] addresses the problem of building models from city environments with a mobile robot. This approach focuses on the textured 3D reconstruction of buildings in the environment. In the so-called "4D-Cities" project, Dellaert and colleagues [3], [11] address the problem of building spatial-temporal models of cities. They use current and historical photographs to reconstruct city scenes at different points in time. The temporal ordering can be inferred from the images by formulating it as a constraint satisfaction problem. This allows for time travels in cities.

III. Landmark and Camera Pose Estimation

We consider the problem of estimating the camera poses and the locations of observed landmarks given a labeled set of camera images. In the remainder of this paper, we use the following notation.

A. Problem Formulation

Let $P_i = (X_{P_i}, Y_{P_i}, \theta_{P_i})^T$ be the position and orientation of the camera when recording image i. Let $L_j = (X_{L_j}, Y_{L_j})^T$ be the position of landmark j in the Cartesian space. Each observation of landmark j seen in image i recorded at position P_i is a horizontal angle α_{ij} that describes the location of the landmark relative to the optical axis of the camera when recording the image.

In this paper, we assume that the 2D locations (X_{P_i}, Y_{P_i}) at which the camera images have been recorded are approximately known since images are supposed to be geo-reference. This information can be obtained from a low cost consumer GPS device. However, the orientation information θ_{P_i} is unknown. Given the observations α_{ij}, the goal is to estimate the landmark locations L_j as well as the orientations of the cameras θ_i. In addition to that, we improve the estimate of the locations of the cameras as delivered by the GPS device.

We consider the landmarks as uniquely identifiable. The problem of extracting appropriate features to identify them is not the focus of this work. We furthermore assume that the roll angle of the camera during image recording is zero. By means of low cost attitude sensors used together with the cameras, images can easily be corrected by a simple rotation. Other information such as the focal length of the lens while taking the image can be obtained from the EXIF tags stored in the images.

Note that we assume to have no direct information about the orientations of the cameras. As a result, images that only contain a single landmark have no influence on the estimate because the landmark can be located anywhere. This makes our approach different from typical bearing-only SLAM techniques which, in general, assume to have an estimate about the orientation of the camera that is typically computed from an image stream recorded by the camera or by using odometry information.

B. Objective Function

In our approach, we solve the described location estimation problem by means of optimization. To apply an optimization procedure, one needs to define an objective function. In our scenario, the objective function can be defined as the error between the obtained observations and the estimated positions of the landmarks.

Let the variables indicated by $\hat{\cdot}$ refer to the estimated quantities. In case the estimated camera and landmark positions are consistent with the observations, we obtain

$$\tan(\hat{\theta}_{P_i} + \alpha_{ij}) \quad = \quad \frac{\hat{Y}_{L_j} - \hat{Y}_{P_i}}{\hat{X}_{L_j} - \hat{X}_{P_i}}. \tag{1}$$

In practice, we want to minimize the position error of the landmarks. Our observations, however, only provide bearing information. Thus, the error to be minimized can be specified by the difference between the estimated landmark location $(\hat{L}_j)$ and the line of sight starting from the camera position $(\hat{P}_i)$ in the direction of the observed landmark $(\hat{\theta}_{P_i} + \alpha_{ij})$ (see Figure 2 for an illustration).

One can easily compute the arc length E_{ij} between the estimated landmark location $(\hat{L}_j)$ and the line of sight resulting from the observation. It can be computed as

$$E_{ij} \quad = \quad e_{ij} r_{ij}, \tag{2}$$

where

$$e_{ij} \quad = \quad \operatorname{atan2}(\hat{Y}_{L_j} - \hat{Y}_{P_i}, \hat{X}_{L_j} - \hat{Y}_{P_i}) - \hat{\theta}_{P_i} - \alpha_{ij} \tag{3}$$

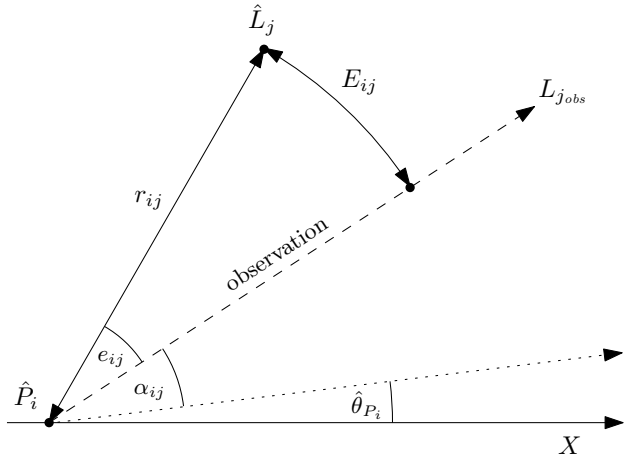


Fig. 2. Illustration on how to compute the estimated error E_{ij}. $\hat{P}_i$ refers to the pose of the camera looking in the direction indicated by the dotted line. The dashed line is the line of sight on which the landmark is located given the observation. $\hat{L}_j$ is the estimated landmark location and e_{ij} is the angular error between the estimated and the observed landmark. r_{ij} is the Euclidean distance between the camera and the estimated landmark.

is the angular error in radians between the estimated and the observed landmark computed from Eq. (1) and

$$r_{ij} = \sqrt{(\hat{X}_{P_i} - \hat{X}_{L_j})^2 + (\hat{Y}_{P_i} - \hat{Y}_{L_j})^2} \tag{4}$$

is the Euclidean distance between camera i and estimated landmark j. The function $\mathrm{atan2}(\Delta Y, \Delta X)$ refers to $\arctan(\Delta Y / \Delta X)$ but explicitly considers the four quadrants.

In our scenario, we assume that the locations of the cameras are measured with a consumer GPS device. Thus, these positions can be regarded as globally correct with a bounded noise term. As a result, the optimization approach should be allowed to *locally* modify them if this reduces the overall error. Therefore, we add a penalty term $f(x)$ that allows for local corrections only. This function is a differentiable barrier function that goes to infinity as x approaches the maximum assumed GPS error, but is near zero close to the measured location. Such an approach is frequently applied to cope with GPS errors [15]. As a result, the objective function E turns into

$$E \quad = \quad \sum_{i,j} {E_{ij}}^2 + \sum_i f(||\hat{P}_i - \tilde{P}_i||), \tag{5}$$

where $\tilde{P}_i$ refers to the locations of the cameras as provided by the GPS observation.

C. Gradient for Optimization

Most optimization techniques either directly rely on gradient information or can be sped up significantly by incorporating knowledge about the gradient of the objective function. In our model, the gradient of the error function E as stated in Eq. (5) is given by

$$\nabla E \quad = \quad \sum_{i,j} \nabla {E_{ij}}^2 + \sum_i \nabla f(||\hat{P}_i - \tilde{P}_i||) \ . \tag{6}$$

The gradient consists of the partial derivatives with respect to the individual variables we want to optimize, namely $\hat{\theta}_{P_i}$, $\hat{X}_{L_j}$, $\hat{Y}_{L_j}$, $\hat{X}_{P_i}$, and $\hat{Y}_{P_i}$. By applying a series of mathematical derivations, we obtain

$$\frac{\partial E}{\partial \hat{\theta}_{P_i}} \quad = \quad -2 \sum_j e_{ij} {r_{ij}}^2 \tag{7}$$

$$\frac{\partial E}{\partial \hat{X}_{L_j}} \quad = \quad 2 \sum_i {e_{ij}}^2 (\hat{X}_{L_j} - \hat{X}_{P_i}) - e_{ij} (\hat{Y}_{L_j} - \hat{Y}_{P_i}) \tag{8}$$

$$\frac{\partial E}{\partial \hat{Y}_{L_j}} \quad = \quad 2 \sum_i e_{ij} (\hat{X}_{L_j} - \hat{X}_{P_i}) + {e_{ij}}^2 (\hat{Y}_{L_j} - \hat{Y}_{P_i}) \tag{9}$$

$$\frac{\partial E}{\partial \hat{X}_{P_i}} \quad = \quad 2 \sum_j e_{ij} (\hat{Y}_{L_j} - \hat{Y}_{P_i}) + {e_{ij}}^2 (\hat{X}_{P_i} - \hat{X}_{L_j}) + \frac{\partial}{\partial \hat{X}_{P_i}} f(||\hat{P}_i - \tilde{P}_i||) \tag{10}$$

$$\frac{\partial E}{\partial \hat{Y}_{P_i}} \quad = \quad 2 \sum_j {e_{ij}}^2 (\hat{Y}_{P_i} - \hat{Y}_{L_j}) - e_{ij} (\hat{X}_{L_j} - \hat{X}_{P_i}) + \frac{\partial}{\partial \hat{Y}_{P_i}} f(||\hat{P}_i - \tilde{P}_i||). \tag{11}$$

Note that the penalty term $f(||\hat{P}_i - \tilde{P}_i||)$ only affects the partial derivatives with respect to the estimated camera locations $\hat{X}_{P_i}$ and $\hat{Y}_{P_i}$.

We have specified the objective function for our problem as well as its partial derivatives. After randomly sampling an initial guess, we can now apply gradient-based optimization techniques to compute a solution.

IV. Optimization

Optimization refers to the task of systematically choosing the values of variables to minimize or maximize an objective function E. For our problem, the objective function is given in Eq. (5). This section briefly introduces gradient descent and RPROP from a general point of view. Both methods are applied in this work.

A. Gradient Descent

Gradient descent is a frequently used iterative optimization technique. Starting from an initial parameter setting x_0, it alternates between (a) computing the gradient ∇E of the objective function E w.r.t. its parameters and (b) changing the parameter vector in the direction opposing the gradient. More formally, we set

$$x_{n+1} \quad = \quad x_n - \varepsilon \cdot \nabla E \ , \tag{12}$$

where ε is a scale factor that specifies the change in the variables according to the gradient. This update-rule is iterated until convergence or until a maximum number of iterations has been carried out.

Standard gradient descent is easy to implement, provided that the gradient of the objective function E is known. The scale factor ε, however, is hard to choose in practice and there is no general rule on how to determine it. If ε is chosen too small, the resulting small steps cause convergence to be slow. Too big values for ε, however, can lead to oscillation or even divergence. In short, standard gradient descent converges rather slowly and has no convergence guarantee in the general case.

B. RPROP

Resilient backpropagation (RPROP) [10] was originally proposed as a learning algorithm for artificial neural networks. The goal was to overcome the weaknesses of standard gradient descent outlined above. In contrast to standard gradient descent, RPROP neglects the absolute value of the derivative. Instead, it considers the changes of signs of the individual partial derivatives. The update rule of RPROP consists of two steps. First, the so-called update value Δ_n^k for each dimension k is computed in each iteration n as

$$\Delta_n^k = \begin{cases} \eta^+ \Delta_{n-1}^k & , \text{ if } \frac{\partial E}{\partial x^k}^{n-1} \frac{\partial E}{\partial x^k}^{n} > 0 \\ \eta^- \Delta_{n-1}^k & , \text{ if } \frac{\partial E}{\partial x^k}^{n-1} \frac{\partial E}{\partial x^k}^{n} < 0 \\ \Delta_{n-1}^k & , \text{ else,} \end{cases} \quad (13)$$

where $0 < \eta^- < 1 < \eta^+$ and $\frac{\partial E}{\partial x^k}^n$ is an abbreviation for $\frac{\partial E}{\partial x^k}(x_n^k)$. Based on this update value Δ_n^k, the update rule can be specified as

$$x_{n+1} = x_n + \begin{cases} -\Delta_n^k & , \text{ if } \frac{\partial E}{\partial x^k}^{n} > 0 \\ +\Delta_n^k & , \text{ if } \frac{\partial E}{\partial x^k}^{n} < 0 \\ 0 & , \text{ else.} \end{cases} \quad (14)$$

According to Eqs. (13) and (14), if the sign of a partial derivative changes with respect to the previous iteration, the step size is decreased by a constant factor η^-. If the sign does not change, the step size is increased by the factor η^+. The former is done in order to prevent the algorithm from jumping over local minima, whereas the latter is done to accelerate convergence in shallow regions.

Despite its comparably fast convergence, RPROP is rather easy to implement. Compared to gradient descent, it does not depend on a fixed scale factor ε which is hard to determine. RPROP adapts its scale factors automatically and thus leads to more robust and flexible optimization, which does not require manual parameter tuning. As we will show in the experimental section, RPROP is a suitable technique for solving our estimation problem and clearly outperforms gradient descent.

Note that other optimization approaches such as Levenberg-Marquardt or scaled conjugate gradient can be used as alternatives to gradient descent or RPROP. However, as we illustrate in the experimental evaluation, the conceptually simpler and easy to implement RPROP already leads to highly satisfactory results.

V. Experiments

We performed a series of simulation and real world experiments to test our method. Simulated experiments allow us to compare the solution of our algorithm versus the ground truth, whereas the real world experiments show that our technique is able to solve the addressed problem in realistic settings.

A. Real World Experiments

For data acquisition, we used a standard digital photo camera and a consumer GPS logger (XAiOX iTrackU SiRF III). We walked through the city of Freiburg and took a series of photos from different locations. We then manually labeled a set of buildings in the images to obtain the correspondences between the images. Note that the data association problem is not addressed in the paper. To get an estimate of the quality of our approach in real applications, we compared the estimated landmark locations to the ones obtained from satellite images. We furthermore mounted a compass to the camera in order to compare the estimated orientation of the camera to the angle indicated by the compass (analog, accurate up to $\sim 3\,\mathrm{deg}$).

To determine the horizontal angle in which objects are observed, the camera needs to be calibrated. Such a calibration is a mapping between pixel coordinates and bearing angles and accounts for lens distortion and other camera specific parameters. To achieve the necessary calibration, one could either use appropriate databases for consumer cameras or accurately calibrate it using chessboard patterns [5].

We collected two datasets, one in Freiburg downtown and one on the campus of the computer science department of the University of Freiburg. In the city center, the landmarks were located in an area of approximately 1.5 km by 1 km (distance camera-landmarks: 180 m to 2.5 km) and on the campus in an area of approximately 320 m by 300 m (distance camera-landmarks: 10 m to 300 m). Whereas the left image in Figure 3 shows the Freiburg downtown area, the right image depicts the campus area including the estimated landmark locations as well as the true positions. Additionally, Figure 4 illustrates the absolute error for the individual landmarks based on the campus experiment. The true landmark locations were measured manually using high-resolution satellite images. To further analyze the robustness of our methods, we carried out 300 optimization runs in both real world experiments and randomly initialized the landmark locations and camera heading angles. In all runs, our approach converged towards the same solution which illustrates its robustness.

We also compared the estimated camera orientations to the ones measured with a compass. It turned out that all estimated orientations differ from the ones measured with the compass by less than 3 deg, which is approximately the measurement accuracy of our analog compass. In the future, a digital compass could be used to automate this task.

B. Simulated Experiments

Simulated experiments allow us to analyze our method in a controlled environment. We examine the evolution of the real error during the process of optimization. We also provide a comparison of the performance of our approach using RPROP and the same approach applying standard gradient descent.

As explained above, we assume that the locations where the images were taken are roughly known from a GPS device. Like all measurements, GPS observations are distorted by noise. Hence, in our simulation experiments, we simulate the noise by sampling from a Gaussian with a standard deviation $\sigma = 10\,\mathrm{m}$. Furthermore, we assume a horizontal opening angle of the cameras of 65 deg.

Fig. 3. True and estimated landmark locations in Freiburg downtown overlayed on a street map (left) as well as for the Freiburg campus experiment overlayed on a building plan (right). Note that due to copyright reasons, we do not visualize the results using the original satellite images.

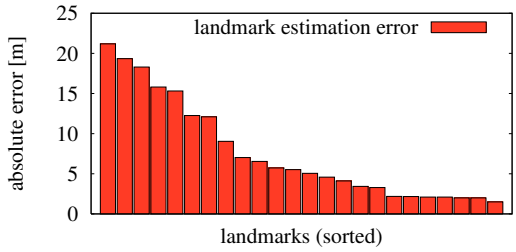


Fig. 4. Individual estimation errors of the different landmarks in the campus experiment. The landmarks are sorted by their errors for better visibility.

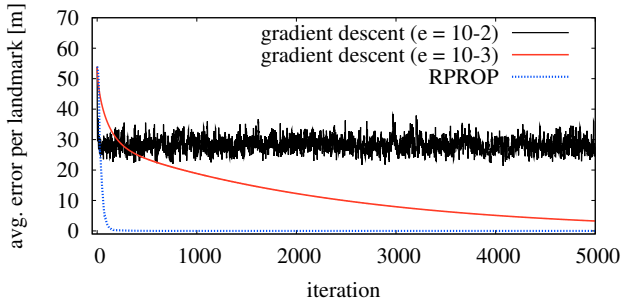


Fig. 5. Convergence behavior of a single optimization run, during which six partially visible landmarks have been observed from six geo-referenced camera locations using gradient descent with two different scale factors (ε) and RPROP. The figure illustrates that an improperly chosen ε parameter leads to oscillation.

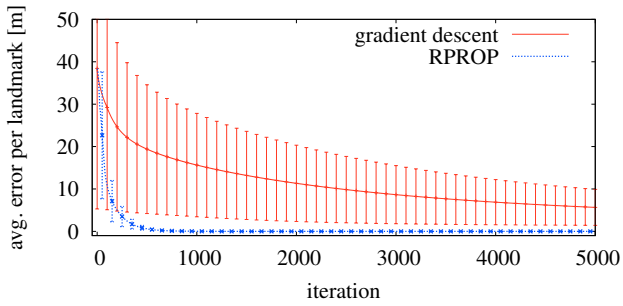


Fig. 6. Empirical evaluation of the convergence behavior of gradient descent versus RPROP. The error-bars in this plot give the 95% confidence interval for 10 runs. On a 2 GHz laptop computer, the average computation times per iteration of RPROP and gradient descent were 35 μs and 8 μs, respectively.

In the first simulation run, we used six images to estimate the positions of six partially visible landmarks. We applied our algorithm using RPROP and gradient descent. We applied gradient descent in two settings, using $\varepsilon = 0.01$ and $\varepsilon = 0.001$. Figure 5 shows the evolution of the real error versus the number of iterations in a typical run of the experiment. As can be seen, RPROP shows the best performance: the algorithm converges quickly to the correct configuration (zero error). In contrast to this, gradient descent with $\varepsilon = 0.001$ converges significantly slower. We repeated the experiment with a value of $\varepsilon = 0.01$. In this setting, the optimization oscillates and does not converge to the correct solution. This illustrates the sensitivity of the factor ε in gradient descent. In contrast to this, RPROP yielded a substantially better performance without the need to manually choose parameters. RPROP adapted these parameters automatically and was able to converge to the correct configuration quickly. To provide a statistical evaluation, Figure 6 depicts the results of the experiment with different configurations averaged over 10 runs. The error bars show the 95% confidence intervals. As can be seen in Figure 7, a similar convergence behavior can be observed when increasing the size of the scene as well as the number of cameras and landmarks.

While RPROP is guaranteed to always converge to a solution, it is still a local optimization approach that might yield a *local* minimum. However, we found that the starting point of the optimization is not a critical choice. With randomly chosen starting locations, our approach converged in all our real world experiments to the same solution.

A more crucial precondition for success is the amount of images and landmarks and their spatial arrangement. Obviously, if only a single landmark is detected by each camera or less than two landmarks are common between different

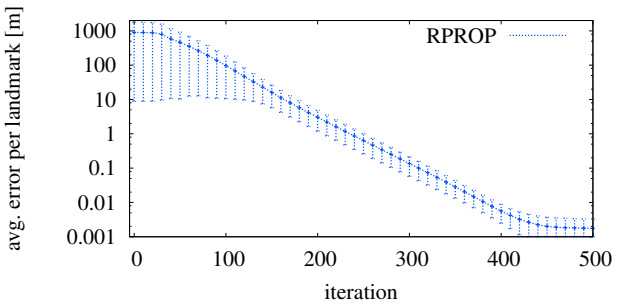


Fig. 7. Convergence behavior on an extremely large dataset. The locations of 10,000 partially visible landmarks and 1000 camera images were initialized randomly. The error-bars give the 95% confidence intervals for 10 runs. On a 2 GHz laptop computer, the average computation time per iteration was 4.5 s.

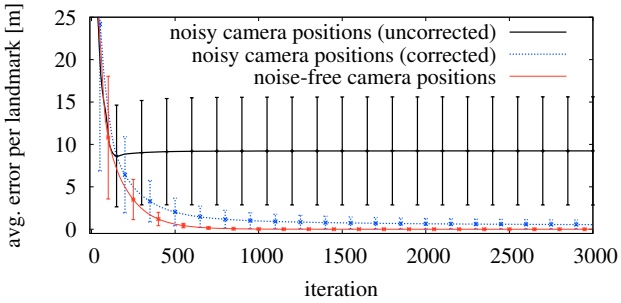


Fig. 8. Influence of the x/y-position error (resulting from the GPS) on the estimate of the landmark locations evaluated over 10 runs. The black line shows the performance of a run with a Gaussian noise of $\sigma = 10\,\mathrm{m}$ without optimizing the camera positions, while the blue line indicates the error when also optimizing the camera locations. The red line shows the result if the camera poses are error-free. As can be seen, our approach is able to compensate for this position error. The performance of the full optimization (blue) approaches the error-free model (red).

images, the system is under-constrained geometrically and cannot be solved. In our real world experiments and also in a large variety of simulated scenarios, enough geometric constraints are present to find a close to optimal solution.

Figure 8 depicts a statistical experiment which illustrates that our approach can easily compensate for the GPS inaccuracies of the geo-referenced photographs. As in all simulated experiments, we added a Gaussian noise with $\sigma = 10\,\mathrm{m}$ in the x and y-positions of the cameras. As can be seen from the diagram, our approach compensates for this noise and converges to the same estimate as if no noise was present.

VI. Conclusions

In this paper, we presented an approach to estimate the positions of landmarks based on a set of labeled, geo-referenced photographs, under absence of any camera heading information. We believe that such an approach is a first step towards allowing a mobile robot to use additional, publicly available sources of information like the image portal Flickr or Google Image Search. Our technique formulates the problem of estimating the positions of photographed objects, such as buildings in a city, as an optimization problem and uses the resilient backpropagation (RPROP) algorithm to solve it. We implemented our method and used it to estimate the locations of different buildings based on photographs taken in the city of Freiburg, Germany. Our experiments show that RPROP significantly outperforms gradient descent in this task.

Despite this encouraging results, there is further space for optimizations. To actually use image databases in a completely autonomous way, our system needs means to robustly eliminate outliers and to build representations of objects for determining correspondences. So far, we assumed the labels to be known.

To summarize, we presented an approach that is able to localize labeled objects based on geo-referenced photographs and to simultaneously estimate the unknown camera headings. This is a first step towards making such information available to robots allowing them to improve their service for applications in which spatial knowledge about the environment is required.

Acknowledgments

This work has partly been supported by the EC under contract numbers FP6-IST-034120-muFly and FP6-IST-045144-RAWSEEDS, by the DFG under contract number SFB/TR-8, and by the German Ministry for Education and Research (BMBF) through the DESIRE project.

References

[1] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ATLAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[2] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[3] F. Dellaert. 4d-cities. Invited Talk at the Int. Symposium on 3D Data Processing, Visualization and Transmission, 2006.

[4] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, San Francisco, CA, USA, 2000.

[5] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. of the CVPR '97*, Washington, DC, USA, 1997.

[6] T. Lemaire, S. Lacroix, and J. Sola. A practical 3d bearing-only slam algorithm. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[8] P.M. Newman. *On the structure and solution of the simultaneous localization and mapping problem*. PhD thesis, University of Sydney, Australia, 1999.

[9] P.M. Newman and J.J. Leonard. Consistent convergent constant time slam. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[10] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, April 1993.

[11] G. Schindler, S.B. Kang, and F. Dellaert. Inferring temporal order of images from 3d structure. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[12] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[13] N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3), 2006.

[14] S. Teller, M. Bosse, M. Jethwa, and A. Khripin. Automated, scalable model capture of urban environments. LCS Research Abstracts, MIT Laboratory for Computer Science, 2003.

[15] S. Thrun and colleagues. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006.

[C11] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

Efficiently Learning High-dimensional Observation Models for Monte-Carlo Localization using Gaussian Mixtures

Patrick Pfaff Cyrill Stachniss Christian Plagemann Wolfram Burgard

***Abstract*— Whereas probabilistic approaches are a powerful tool for mobile robot localization, they heavily rely on the proper definition of the so-called observation model which defines the likelihood of an observation given the position and orientation of the robot and the map of the environment. Most of the sensor models for range sensors proposed in the past either consider the individual beam measurements independently or apply uni-modal models to represent the likelihood function. In this paper, we present an approach that learns place-dependent sensor models for entire range scans using Gaussian mixture models. To deal with the high dimensionality of the measurement space, we utilize principle component analysis for dimensionality reduction. In practical experiments carried out with data obtained from a real robot, we demonstrate that our model substantially outperforms existing and popular sensor models.**

I. Introduction

In the past, probabilistic localization techniques have been demonstrated to be a robust approach to mobile robot localization as they allow the vehicles to globally localize themselves, to efficiently keep track of their position, and to even recover from localization failures. One of the key challenges in context of probabilistic localization, however, lies in the design of the so-called observation model $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{m})$ which is a likelihood function that specifies how to compute the likelihood of an observation $\boldsymbol{z}$ given the robot is at pose $\boldsymbol{x}$ in a given map $\boldsymbol{m}$. For probabilistic approaches the proper design of the likelihood function is essential. Too optimistically specified sensor models might make the vehicle overly confident in its position. In the context of Monte-Carlo-Localization [4], this can lead to a depletion of particles and finally might cause a divergence of the filter. Too conservative models, in contrast, might result in a high pose uncertainty or even prevent the robot from localizing itself as the sensor information cannot compensate for the uncertainty introduced by the motion of the vehicle.

In the past, sophisticated sensor models have been developed for probabilistic approaches to robot localization. Some of them take into account various aspects such as objects not contained in the map or sensor cross-talk. Whereas such approaches have been proven to be robust even in real-world situations, they do not appropriately take into account potential effects not stemming from the measurement process itself. This, for example, regards the fact that the map typically is only an approximation of the real world. Additionally, such sensor models are sensitive to discontinuities in the map. For example, when the environment is cluttered, slight changes in the pose of the robot in the map might lead to huge differences in the length of the expected measurement at that particular location. This fact is illustrated in Figure 1. The figure shows to different scans obtained with a laser range scanner with a robot that passes a doorway. Whereas the two positions, at which the scans were taken, are close to each other, the obtained scans differ largely. Accordingly, even small errors in the pose estimate can lead to an extremely small likelihood of the measurement. This in turn can lead to a divergence of the filter.

Fig. 1. In mobile robot localization, small variations in the robot pose can cause large variations of the range measurements. This leads to multi-modal distributions of beam-lengths even in small areas around a potential pose.

One sensor model that has been especially designed to cope with such situations are the so-called likelihood fields [17]. This "beam-end-point-model" provides smooth and multi-modal likelihood functions to better deal with clutter in the environment but ignores the information along the individual beam of a range measurement. Therefore, likelihood fields are less effective in situations in which the robot has to perform global localization. Most observation models furthermore assume the independency of the individual beams of a laser range scan. However, the more beams a scan has, the more this assumption is violated, which then might lead to overly confident estimates. Recently, some techniques have been developed that explicitely consider the dependencies of individual beams [14], [16]. However, these techniques have the drawback that they assume an uni-modal distribution.

In this paper, we propose a novel probabilistic observation model for proximity sensors such as laser range finders. Our model has two advantages over most previous approaches. First, it explicitely considers the dependencies between the individual beams of a range scan, and second, is accounts for

The authors are with the University of Freiburg, Department of Computer Science, Georges-Koehler-Allee 79, 79110 Freiburg, Germany. {pfaff, stachnis, plagem, burgard} @ informatik.uni-freiburg.de

the multi-modal nature of the observation function. It does so while still considering that the observation is obtained from a time-of-flight proximity sensor (such as laser range finders or sonars). This is achieved by considering place-dependent measurement models and utilizing a Gaussian mixture model together with a dimensionality reduction technique. In practical experiments carried out with data obtained with a real robot we demonstrate that our new model substantially outperforms existing sensor models.

This paper is organized as follows. After discussing related work in the next section, we briefly describe in Section III Monte Carlo localization and the principle of likelihood models. In Section IV, we introduce our novel likelihood model based on high-dimensional mixtures of Gaussians and finally, in Section V, we present experimental results illustrating that our sensor model outperforms other popular likelihood models.

II. Related Work

In the literature, various techniques for computing the likelihood function for probabilistic localization methods with proximity sensors have been introduced [3], [8], [17], [18]. These approaches either directly approximate the physical characteristics of the sensor or try to provide smooth likelihood models to increase the robustness of the localization process. In the past, is has been observed that the likelihood function can have a major influence on the performance of Monte Carlo Localization. Thrun *et al.* [19], for example, observed that more particles are required if the likelihood function is peaked. In the limit, i.e., for a perfect sensor, the number of required particles becomes infinite. To deal with this problem, Lenser and Veloso [10] and Thrun *et al.* [19] introduced techniques to directly sample from the observation model and in this way ensure that there is a critical mass of samples at the important parts of the state space. Unfortunately, this approach depends on the ability to sample from observations, which can often only be done in an approximate, inaccurate way. An alternative strategy to deal with this problem is to artificially inflate the measurement uncertainty to achieve a regularization of the likelihood function, e.g., see the *Scaling Series* approach presented by Petrovskaya *et al.* [12].

The classical Kalman filter has limitations since it requires Gaussian observation likelihoods and thus cannot handle multi-modalities or ambiguous situations. To overcome this problem, several researchers used Gaussian mixture models. Duckett and Nehmzow [7], for example, introduced a multi-modal generalization of the Kalman filter based on mixtures of Gaussians. Recently, Upcroft *et al.* [20] introduced a fast re-parameterization of Gaussian mixture models to represent the probability distribution. Takamasa *et al.* [9] use Gaussian mixture models to fuse odometry and sonar and to reduce the localization error in the case of noisy sensors.

Recently, Limketkai *et al.* [11] proposed an approach for learning the motion and sensor model for MCL using conditional random fields. This allows for considering the dependencies between the individual beams of a laser range scan. The approach is furthermore reported to provide better results than generatively learned models but it requires ground truth location information of a robot to carry out discriminative learning.

The focus of this paper is to model possible multi-modalities in likelihood functions for laser range measurements using Gaussian mixture models. Our approach improves the robustness of probabilistic localization approaches like MCL especially in situations in which small changes of the robot's pose can have drastic effects on the range measurements.

III. Monte Carlo Localization using Range Sensors

A. Pose Estimation using a Particle Filter

Throughout this paper, we consider the problem of estimating the pose $\boldsymbol{x} = (x, y, \theta)$ of a robot relative to a given map $\boldsymbol{m}$ using a particle filter. The key idea of this approach is to maintain a probability density $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{0:t-1})$ of the location $\boldsymbol{x}_t$ of the robot at time t given all observations $\boldsymbol{z}_{1:t}$ up to time t and all control inputs $\boldsymbol{u}_{0:t-1}$ up to time $t-1$. This probability is calculated recursively as

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{0:t-1}) = \alpha \cdot p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_{t-1}, \boldsymbol{x}_{t-1}) \cdot p(\boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1} \quad (1)$$

Here, α is a normalizing constant ensuring that $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{0:t-1})$ sums up to one over all $\boldsymbol{x}_t$. The terms to be described in Eqn. (1) are the *prediction model* $p(\boldsymbol{x}_t \mid \boldsymbol{u}_{t-1}, \boldsymbol{x}_{t-1})$ and the *sensor model* $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ respectively.

For the implementation of the described filtering scheme, we use a sample-based approach which is commonly known as *Monte Carlo localization (MCL)* [4]. Monte Carlo localization is a variant of particle filtering [6] where each particle corresponds to a possible robot pose and has an assigned weight w_i. The *belief update* from Eqn. (1) is performed by the following two alternating steps:

1) In the **prediction step**, we draw for each particle with weight w_i a new particle according to w_i and to the prediction model $p(\boldsymbol{x}_t \mid \boldsymbol{u}_{t-1}, \boldsymbol{x}_{t-1})$.
2) In the **correction step**, a new observation $\boldsymbol{z}_t$ is integrated. This is done by assigning a new weight w_i to each particle according to the sensor model $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$.

B. Likelihood Models for Range Sensors

The likelihood model $p(\boldsymbol{z} \mid \boldsymbol{x})$ plays a crucial role in the correction step of the particle filter and its proper design is essential for the robustness of the filter. Due to that, in this paragraph we will describe typical likelihood models for range sensors and we shortly will introduce the likelihood models of our previous work [14], [15]. Afterwards, we will present our new high dimensional Gaussian mixture model that is able to represent multi-modalities in the likelihood function as well as dependencies between the individual laser beams.

A laser scan $\boldsymbol{z}_t$ is a vector of beams $\boldsymbol{z}_t = (z_t^1, \ldots, z_t^N)^T$ which have fixed orientations relative to the sensor. *Beam-based* sensor models (see Fox *et al.* [8] for a typical example) consider each value z_t^i of the measurement vector $\boldsymbol{z}$ as a separate range measurement and represent its one-dimensional distribution by a parametric function depending on the expected distance in the respective beam direction. Such models are closely linked to the geometry and the physics involved in the measurement process. They are sometimes also called *ray cast* models because they rely on ray casting operations within the map of the environment, e.g., an occupancy grid map, to calculate the expected beam lengths. Another popular measurement model for range finder sensors are the so-called *likelihood fields* (aka *end point model*) [17], which are a correlation-based method that measures the correlation between the measurement and the map. Here, the likelihood of a range measurement is a function of the distance of the respective endpoint of the beam to the closest obstacle in the environment. This model lacks physical explanation as it can basically "see through walls", but in the case of position tracking it is efficient and works well in practice. The reader may notice that likelihood fields are less effective in situations in which the robot has to perform global localization. In all above-mentioned approaches, the individual beams are treated independently and the likelihood $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, m)$ of the entire scan $\boldsymbol{z}_t$ is calculated as the product of the individual beam likelihoods $p(z_t^i \mid \boldsymbol{x}_t, \boldsymbol{m})$.

Given the high resolution of typical laser range finders (.25 to 1 degrees), the independence assumption leads to highly peaked likelihoods. In practice, this problem is dealt with by sub-sampling the measurements [18], by introducing minimal likelihoods for beams, or by other means of regularization of the resulting likelihoods (see Arulampalam *et al.* [1]). One way to overcome the peakedness is to consider that the likelihood models are location-dependent as well as that the location of the robot is modeled by set finite set of pose hypotheses (particles). Therefore, we estimate $p(\boldsymbol{z} \mid \boldsymbol{x})$ based on the local environment $\mathcal{U}(\boldsymbol{x})$ around a pose hypothesis $\boldsymbol{x}$ by

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \int_{\tilde{\boldsymbol{x}} \in \mathcal{U}(\boldsymbol{x})} p(\boldsymbol{z} \mid \tilde{\boldsymbol{x}})\; p(\tilde{\boldsymbol{x}})\; d\tilde{\boldsymbol{x}}\;. \tag{2}$$

Here, $\mathcal{U}(\boldsymbol{x})$ is a particle-dependent, circular area. The diameter of that area is given by the distance to the closest neighboring particle which can be efficiently determined using a kd-tree. This model is able to represent the fact that pose hypotheses given by the samples are typically less accurate than the measurements obtained by a (SICK) laser range finder.

Thus, if one learns $p(\boldsymbol{z} \mid \boldsymbol{x})$ directly for exact sensor poses $\boldsymbol{x}$, e.g., with a mobile robot that is not moved during training, one gets an extremely peaked model with $p(\boldsymbol{z} \mid \boldsymbol{x} + \boldsymbol{\delta}) \ll p(\boldsymbol{z} \mid \boldsymbol{x})$ already for small pose perturbations $\boldsymbol{\delta}$. This peakedness, in turn, leads to problems when only a finite number of pose hypotheses can be evaluated, as it is the case, for example, with particle filters where the number of particles is limited. The model described in Eqn. 2, however, is able to explicitly consider the sampling density by adjusting the spatial extent of the local neighborhoods $\mathcal{U}(\boldsymbol{x})$ accordingly. The hard task is indeed to estimate and represent the distributions of range scans $p(\boldsymbol{z} \mid \boldsymbol{x})$ from a given number of training scans from $\mathcal{U}(\boldsymbol{x})$, which are typically simulated using the map of the environment m.

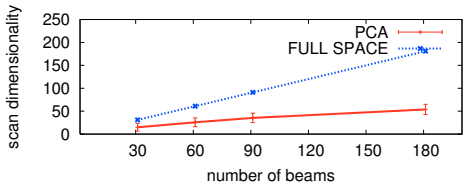


Fig. 2. Obtained dimensionality reduction over a full robot trajectory in a real world experiment using Principal component analysis (PCA).

In our previous approaches, we modeled the observation likelihood as either unimodal distributions for single beams [8], [13] or for entire scans [14], [16]. Alternatively, our recently proposed method [15] is able to consider the multi-modality of the observation model but is unable to represent the dependency between the individual scans. In contrast to this, the novel method presented in this paper combines the advantages of these previous models: It considers the dependency between the beams of a range scan and it correctly handles the multi-modal nature of the likelihood function and at the same time can be computed efficiently. As we will demonstrate in the experiments, this more sophisticated model significantly improves the ability of a mobile robot to localize itself.

In the following section, we describe how to efficiently learn a high-dimensional Gaussian mixture model for the distribution obtained by Eqn. 2 to improve the robustness of the localization process.

IV. Learning High Dimensional Gaussian Mixture Observation Models

Figure 1 illustrates the drastic effects that small changes of the robot's pose can have on the measured range scans. The distribution of measured distances that arises when the robot pose is varied locally as described in the previous section is only unimodal in a perfectly convex world. In general, however, there can be large jumps in perceived range measurements when the sensor pose is changed only slightly. Typically, such multi-modalities arise in the proximity of doorways, corners, and cluttered areas of the environment.

One way to model the different modes in the distribution of the expected range observation for each laser beam is to explicitly consider the multi-model nature [15]. Whereas this technique yields appropriate, multi-model distributions for individual beams, it is unable to model the dependency between in individual beams.

The straightforward extension that considers also the dependency between the individual beams is to learn a Gaussian mixture model (GMM) based on full laser scans and not individual beams. Most clustering techniques based on the Gaussian mixture model, however, show a suboptimal performance if the size of the training dataset is too small

compared to the number its dimensionality (the parameters to estimate). Typically, this leads to serious over-fitting. It is therefore necessary to find a good balance between the number of parameters to estimate and the generality of the model.

One way to overcome this problem is to apply k-means clustering since it does not estimate the covariance matrix and thus less parameters need to be estimated. However, the dependencies between beams are then neglected when estimating the clusters. Alternatively, the high dimensional data clustering (HDDC) approach recently proposed by Bouveyron *et al.* [2] can be applied. This technique combines dimensionality reduction with the expectation-maximization (EM) algorithm [5] to learn a Gaussian mixture model. By assuming certain dependencies in the covariance matrix, the learned clusters can be easily re-projected onto the original space yielding robust clusters with a significantly reduced risk of over-fitting.

Our approach can be seen as a reduced version of the method of Bouveyron *et al.* [2]. We perform an EM-based Gaussian mixture clustering in a reduced space and then use the obtained class coefficients to compute the mixture model in the high dimensional space.

A. Dimensionality Reduction

In dimensionality reduction techniques, one is interested in finding a mapping from the original, n-dimensional inputs space to a new space with $k < n$-dimensions with a minimal loss of information. Principal component analysis (PCA) is an un-supervised technique that maximizes the variance in the data in the new space.

Let Σ be the covariance matrix of the input data D. PCA computes the eigenvalues λ_i and eigenvectors of Σ. Let Q be the matrix of the eigenvectors sorted according to the eigenvalues. We can then compute a matrix $\Delta = Q^T \Sigma Q$ so that Δ is a diagonal matrix with the eigenvalues on the diagonal in descending order. Let $\lambda_i \geq \lambda_j$ for all $i < j$. We consider only the first k dimensions for clustering that cover 95% of the variance which is given by $\sum_{i=1}^{k} \lambda_i \left[\sum_{i=1}^{n} \lambda_i\right]^{-1} \geq 0.95$.

By considering only the first k dimensions, we obtain an approximative but compact representation for laser range scans. Figure 2 depicts the obtained dimensionality reduction in real world settings.

Concretely, for each pose hypothesis $\boldsymbol{x}_t$, we simulate L complete range scans $\mathcal{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_L\}$ at locations drawn uniformly from $\mathcal{U}(\boldsymbol{x}_t)$ using the given map $\boldsymbol{m}$ of the environment. The simulation of the laser range scans $\mathcal{D}$ takes into account the geometry and the physics involved in the measurement process. It relies on ray casting operations within an occupancy grid map to calculate the expected beam lengths. The elements of the set $\mathcal{D}$ of laser range scans are used to compute the PCA and thus lead to a projection into a reduced, k-dimensional space.

B. Clustering in the Reduced Space

Let $\tilde{\cdot}$ refer to quantities computed in the reduced, k-dimensional space. Thus, $\tilde{\mathcal{D}} = \{\tilde{\boldsymbol{d}}_1, \ldots, \tilde{\boldsymbol{d}}_L\}$ are the elements of the set $\mathcal{D}$ projected to the low-dimensional space given the transformation matrices described in the previous section. In the reduced space, we are now able to efficiently cluster the range scans while reducing the risk of over-fitting (compare [2]).

To estimate the clusters in the low-dimensional space, we apply the EM algorithm to efficiently learn the mixture distribution. The EM algorithm iteratively assigns the reduced data scans in $\tilde{\mathcal{D}}$ to the mixture components and optimizes their parameters in the following manner. Consider that $\boldsymbol{\theta}'$ denotes the current estimate of parameters $\tilde{\boldsymbol{\mu}}_j$, $\tilde{\Sigma}_j$, and α_j. In the E-Step, we calculate the expected value of the complete log-likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = E\left[\log\{p(\tilde{\mathcal{D}}, Y \mid \boldsymbol{\theta})\} \mid \tilde{\mathcal{D}}, \boldsymbol{\theta}'\right] \tag{3}$$

$$= \int_y \log\{p(\tilde{\mathcal{D}}, y \mid \boldsymbol{\theta})\} p(y \mid \tilde{\mathcal{D}}, \boldsymbol{\theta}')\, dy, \tag{4}$$

where Y denotes data associations of the projected simulated data points $\tilde{\mathcal{D}}$ to one of the Gaussian mixture components. Visually speaking, we estimate the assignment likelihoods of the individual samples to the clusters while keeping the other model parameters fixed. Then, in the M-Step, we fix the data associations and optimize the expected value of the complete log-likelihood

$$\boldsymbol{\theta}'' = \text{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}') \tag{5}$$

by updating the cluster parameters according to

$$\alpha_j = \frac{1}{L} \sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}'), \tag{6}$$

$$\tilde{\boldsymbol{\mu}}_j = \frac{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')\, \tilde{\boldsymbol{d}}_l}{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')}, \tag{7}$$

$$\tilde{\Sigma}_j = \frac{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')(\tilde{\boldsymbol{d}}_l - \tilde{\boldsymbol{\mu}}_j)(\tilde{\boldsymbol{d}}_l - \tilde{\boldsymbol{\mu}}_j)^T}{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')}. \tag{8}$$

We now set $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}''$ and iterate this procedure until the amount of improvement per iteration falls below a specified threshold. To determine the actual number of clusters in the resulting model, we apply the Bayesian information criterion and choose the model with the best score.

C. Transferring the Mixture Components to the Measurement Space

After identifying the individual clusters and the corresponding probabilities $P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')$, we can compute our mixture model in the high dimensional space. This can be easily achieved if we assume that the corresponding probabilities are identical in the reduced space as well as in the measurements space. Thus, the mixture in the high-dimensional space is given by

$$p(z_t \mid \boldsymbol{x}_t, \boldsymbol{m}) = \sum_{j=1}^{J} \alpha_j \mathcal{N}(\boldsymbol{x}_t, \mu_j, \Sigma_j), \tag{9}$$

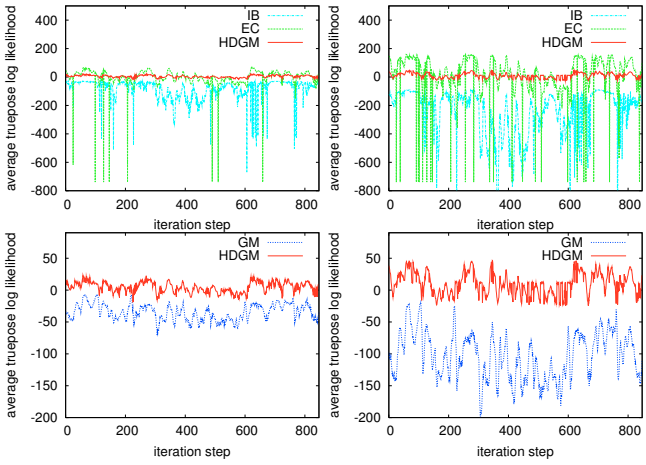


Fig. 3. Evaluated likelihood for 61, and 181 laser beams (from left to right) for different sensor models (upper diagrams) and the two sensor models(*HDGM*,*GM*) which take the multi-modalities in the laser measurements into account (lower diagrams) at 847 robot poses in our office environment depicted in Figure 5.

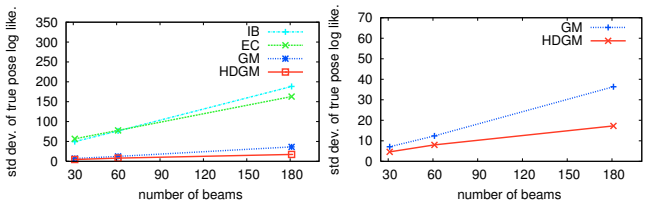


Fig. 4. Standard deviations of the different sensor models for 31, 61, and 181 laser beams (left). Comparison of the standard deviation of the two sensor models(*HDGM*,*GM*) which take the multi-modalities in the expected laser measurements into account.

where J is the number of clusters and $\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}, \Sigma)$ refers to the n-dimensional Gaussian evaluated at $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance Σ according to

$$\boldsymbol{\mu}_j = \frac{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')\, \boldsymbol{d}_l}{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')} \tag{10}$$

$$\Sigma_j = \frac{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')(\boldsymbol{d}_l - \boldsymbol{\mu}_j)(\boldsymbol{d}_l - \boldsymbol{\mu}_j)^T}{\sum_{l=1}^{L} P(j \mid \tilde{\boldsymbol{d}}_l, \boldsymbol{\theta}')}. \tag{11}$$

In contrast to former approaches which modeled the likelihood functions as unimodal distributions for single beams [8], [13] or entire scans [14], [16], or as a multi-modal distributions for single beams [15], we now consider high-dimensional, multi-modal mixture models. This allows us to take the dependency between the individual beams as well as the multi-modal nature of the distribution into account. As we will demonstrate in the experiments, this more sophisticated model significantly improves the ability of a mobile robot to localize itself.

V. Experiments

The approach described above has been implemented and tested on data obtained with a mobile robot. To evaluate our approach, we performed several experiments. We first show that the pose uncertainty of the robot can result in serious problems during a localization process, especially when the multi-modality of the beams is not considered. Additionally, we show the improvements achieved by also considering the dependencies between the individual laser beams. Then in the second set of experiments, we analyze our high-dimensional Gaussian mixture model in a global localization task in which multi-modal situations frequently occur. We therefore compare it to alternative models, which do not simultaneously take into account the multi-modality and the dependencies between the individual laser beams. In particular, we compared the performance of the following sensor models:

HDGM: Our high-dimensional Gaussian mixture model as detailed in Section IV.

GM: The place-dependent beam-based Gaussian mixture sensor model as detailed in our previous work [15].

IB: The standard beam-based sensor model that assumes independent beams with an additive white noise component.

EC: The scan-based, place-dependent model with learned covariance matrix as detailed in our earlier previous work [14].

A. Likelihood Evaluation

In the first set of experiments, we evaluated the likelihood of the true position of the robot in a data set acquired using a real robot. We therefore compared our high-dimensional Gaussian mixture model (*HDGM*) to other likelihood models which are also based on ray casting operations (*GM*, *IB*, and *EC*). This set of experiments is designed to investigate the case that the robot is not able to localize itself at different locations with the same robustness. In our previous work [15], we demonstrated that whenever the robot traverses regions close to obstacles, doorways, or clutter, the likelihood of the true position decreases. In the case of global localization using a particle filter this leads to serious problems because the particles at these positions have a high risk of being depleted. Then, we calculated for different sensor models (*GM*, *IB*, *EC*, and *DC*) the likelihood of the simulated range scan given the true position of the robot. The two upper diagrams of Figure 3 show the evaluated likelihood for 61, and 181 laser beams (from left to right) for different sensor models at 847 robot poses in our office environment depicted in Figure 5. The lower diagrams show the same for the two sensor models (*HDGM*, *GM*) which take the multi-modalities in the expected laser measurements into account. As can be seen from Figure 4, our high-dimensional Gaussian mixture model (*HDGM*) yields a smaller variance in the estimated likelihood of the true pose compared to the other sensor models. Additionally, the right diagram of this Figure illustrates that our novel high-dimensional model (*HDGM*) yields even a smaller variance in the estimated likelihood compared to the beam-based Gaussian mixture model (*GM*) especially when the number of integrated laser beams is increased. This higher variance in the estimated likelihoods, which is caused by the independence assumption of the beam-based sensor model might lead to a divergence of the probabilistic localization even in the case of position tracking.

B. Localization

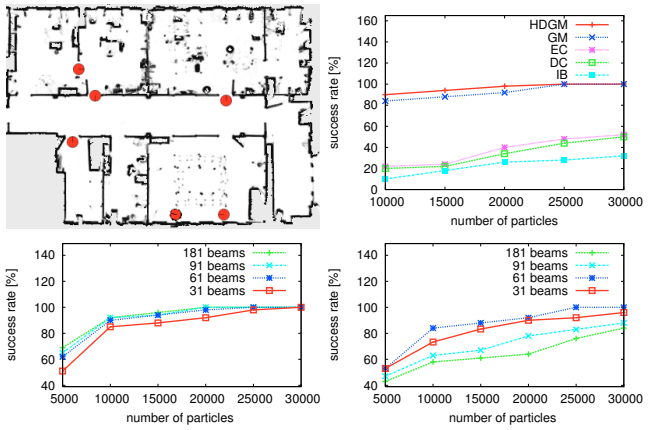


Fig. 5. The six positions with the highest probability that the global localization in the office environment fails (upper left). The upper right diagram shows the number of successful localizations after ten integrations of 61 measurements at these locations. The lower diagrams show the same experiment for the two multi-modal sensor models (left: *HDGM*, right: *GM*) for different beam numbers.

The second set of experiments is designed to illustrate that our new high-dimensional sensor model (*HDGM*), which takes the multi-modality as well as the dependencies of measurements into account, achieves a more robust and accurate localization than the other sensor models. The upper left image in Figure 5 shows the six positions in a real environment where we obtained the highest probability that the global localization fails. These probabilities have been determined by random restarts of the localization procedure during 50 complete runs on the data set. At these positions typically the likelihoods of the true poses are extremely low due to the multi-modality of the measurements. To evaluate the properties of the different sensor models, we performed 20 global localization runs at each position and compared the average success rates. In these experiments, we assumed that the localization was achieved when the mean of the particles differed by at most 50 cm from the true location of the robot. The upper diagram of Figure 5 shows the number of successful localizations after ten integrations of 61 measurements at these locations. The lower diagrams show the same experiment for different beam numbers for the two sensor models (*HDGM*, *GM*) which take the multi-modalities in the expected laser measurements into account. The experiments show that our high-dimensional Gaussian mixture model (*GM*) allows us to more robustly localize the robot in situations in which the other models frequently fail. Additionally, it demonstrates that we are able to integrate more measurements to achieve a higher accuracy of the filtering process without losing robustness.

VI. Conclusions

In this paper, we presented a novel place-dependent sensor model for range scans that considers entire scans instead of individual beams and in this way overcomes the independence assumption underlying popular alternative models. At the same time, it utilizes Gaussian mixture models to represent potential multi-modalities of the likelihood function. To reduce the dimensionality of the measurement space it applies the principle component analysis.

Our approach has been implemented and extensively tested on data obtained with mobile robots equipped with laser range finders. In our experiments, our new model showed superior performance over other popular models proposed in the past.

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. In *IEEE Transactions on Signal Processing*, volume 50, pages 174–188, 2002.

[2] C. Bouveyron, S. Girard, and C. Schmid. High-dimensional data clustering. *Computational Statistics and Data Analysis*, 52(1):502–519, 2007.

[3] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, Kavraki L.E., and S. Thrun. *Principles of Robot Motion Planning*. MIT-Press, 2005.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 99–141, 1998.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.

[6] A. Doucet, N. de Freitas, and N. Gordan, editors. *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[7] T. Duckett and U. Nehmzow. Mobile robot self-localization using occupancy histograms and a mixture of Gaussians location hypotheses. *Robotics and Autonomous Systems*, 34(2-3):119–130, 2001.

[8] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[9] T. Koshizen, P. Bartlett, and A. Zelinsky. Sensor fusion of odometry and sonar sensors by the Gaussian mixture Bayes' technique in mobile robot position estimation. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1999.

[10] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2000.

[11] B. Limketkai, D. Fox, and L. Liao. Crf-filters: Discriminative particle filters for sequential state estimation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[12] A. Petrovskaya, O. Khatib, S. Thrun, and A.Y. Ng. Bayesian estimation for autonomous object manipulation based on tactile sensors. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[13] P. Pfaff, W. Burgard, and D. Fox. Robust monte-carlo localization using adaptive likelihood models. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 181–194. Springer Verlag, 2006.

[14] P. Pfaff, C. Plagemann, and W. Burgard. Improved likelihood models for probabilistic localization based on range scans. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[15] P. Pfaff, C. Plagemann, and W. Burgard. Gaussian mixture models for probabilistic localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[16] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A nonparametric bayesian measurement model for range finders. In *Robotics: Science and Systems (RSS)*, June 2007.

[17] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[18] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT-Press, 2005.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2), 2001.

[20] B. Upcroft, S. Kumar, M.F. Ridley, L. Ong, and H.F. Durrant-Whyte. Fast re-parameterisation of Gaussian mixture models for robotics applications. In *Australian Conference on Robotics and Automation*, 2004.

[C3] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

Coordinated Multi-Robot Exploration using a Segmentation of the Environment

Kai M. Wurm Cyrill Stachniss Wolfram Burgard

***Abstract*— This paper addresses the problem of exploring an unknown environment with a team of mobile robots. The key issue in coordinated multi-robot exploration is how to assign target locations to the individual robots such that the overall mission time is minimized. In this paper, we propose a novel approach to distribute the robots over the environment that takes into account the structure of the environment. To achieve this, it partitions the space into segments, for example, corresponding to individual rooms. Instead of only considering frontiers between unknown and explored areas as target locations, we send the robots to the individual segments with the task to explore the corresponding area. Our approach has been implemented and tested in simulation as well as in real world experiments. The experiments demonstrate that the overall exploration time can be significantly reduced by considering our segmentation method.**

I. Introduction

Autonomous robots that are designed to create a map of their environment require the capability to effectively cover the space. There are several applications in which robots have been designed to autonomously explore their environment such as planetary exploration or in disaster missions. Using a coordinated team of robots instead of a single robot has often been suggested to be advantageous [4], [7] and cooperating robots have the potential to accomplish a task faster than a single robot [11]. By using several robots, redundancy can be explicitely introduced so that such a team can be expected to be more fault-tolerant than a single robot. Another advantage of robot teams arises from merging overlapping sensor information, which can help to compensate for sensor uncertainty. However, when robots operate in teams there is the risk of interference between them [10], [20]. For example, if the robots have the same type of active sensors such as ultrasound sensors, the overall performance can be reduced due to cross-talk. The more robots are used, the more time each robot may spend on detours in order to avoid collisions with other members of the team.

In this paper, we consider the problem of efficient exploration with teams of mobile robots that seek to minimize the overall time required to complete the mission. The entire task of coordinating a team of robots during exploration can roughly be separated into two subsequent tasks. First, one needs to identify potential exploration targets for the robots. Second, one needs to assign the individual robots to the target locations calculated in the previous step.

All authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

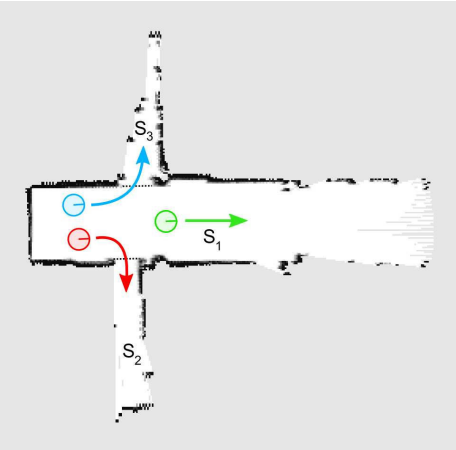


Fig. 1. Typical coordination of robots obtained by assigning them to different segments of the partial map.

A popular method for generating potential exploration targets has been proposed by Yamauchi *et al.* [26]. In this approach, robots are sent to so-called frontiers, which are given as the borders between the explored and the unexplored space. In a multi-robot context, it is important to carefully assign robots to targets so that redundant work and interference between robots is minimized. Therefore, it is the exploration strategy which affects the efficiency of the robot team the most. In many approaches, the robots are assigned directly to frontier targets based on a cost function that takes into account the expected path costs or travel time as well as a utility function that covers aspects such as the expected gain in information [3], [9], [21], [24], [28].

The coordination strategies described there consider individual locations rather than segments of the environment. Segmentation approaches which have recently received an increased amount of attention [2], [8], [23], [27] have originally been designed to facilitate topological localization and loop closing or have been used to reduce planning costs. In this paper, we introduce a new online coordination strategy for multi-robot exploration. It uses a segmentation of the already explored area to assign robots to segments instead of directly assigning them to frontier targets. Based on this segmentation, the robots are distributed over the environment more effectively which leads to a reduction of redundant work and the avoidance of interference between robots. As a result, the exploration time is significantly reduced.

This paper is organized as follows. After discussing related work, we describe the Hungarian Method for target assignment in Section III. In Section IV, we introduce a graph-based method for map segmentation while in Section V we present our coordination approach. Finally, we describe simulated and real world experiments conducted to evaluate our approach.

II. Related Work

The problem of exploring unknown terrains with teams of mobile robots has received considerable attention in the past. Yamauchi [26] presented a technique to learn maps with a team of mobile robots. He introduced the concept of frontiers between known and unknown areas in a grid map, which are widely used to select potential target locations during exploration. In this paper, we also consider frontiers but additionally utilize the structure of the environment for defining potential target locations. Koenig *et al.* [14] analyze different terrain coverage methods for small robots with limited sensing and computational capabilities. Furthermore, there has been research on how to deal with limited communication in the context of multi-robot exploration [3], [19].

An approach towards cooperation in heterogeneous robot systems has been presented by Singh and Fujimura [21]. If a robot is too big to pass through a narrow passage, it informs other robots about this task. Howard *et al.* [12] presented an incremental deployment approach that explicitly deals with obstructions, i.e., situations in which the path of one robot is blocked by another. Zlot and colleagues [28] proposed an architecture for mobile robot teams in which the exploration is guided by a market economy. They consider sequences of potential target locations for each robot and trade tasks between the robots using single-item first-price sealed-bid auctions. Such auction-based techniques have also been applied by Gerkey and Matarić [9] to efficiently solve the task allocation problem with a group of robots.

Matarić and Sukhatme [17] consider different strategies for task allocation in robot teams and analyze the performance of the team in extensive experiments. Ko *et al.* [13] present an approach that uses the Hungarian method [15] to compute the assignments of frontier cells to robots. In contrast to our work, Ko *et al.* mainly focus on finding a common frame of reference in case the start locations of the robots are not known.

In a previous work [24], we considered the problem of integrating semantic background information into the coordination procedure. This technique is related to the method proposed in this paper, even if the methodology is substantially different. Compared to our previous approach [24], we obtain a significantly reduced exploration time also for small teams of robot. The map segmentation technique used throughout this work is related to the spatial semantic hierarchy introduced by Kuipers and Byun [16]. The difference lies in the fact that we do not learn a model based on distinct places but utilize this information for a better coordination. Learning topological maps is itself a research field on its own and different methods have been proposed [2], [8], [25], [27]. These approaches are related to the technique described in this paper as they can be applied to separate the environment into appropriate regions that are then assigned to the individual robots.

III. Target Assignment using the Hungarian Method

In 1955, Kuhn [15] presented a general method, which is often referred to as the Hungarian method, to assign a set of jobs to a set of machines given a fixed cost matrix. Consider a given $n \times n$ cost matrix which represents the cost of all individual assignments of jobs to machines. The Hungarian method, which is able to find the optimal solution with the minimal cost given this matrix, can be summarized by the following three steps:

1) Compute a reduced cost matrix by subtracting from each element the minimal element in its row. Afterwards, do the same with the minimal element in each column.
2) Find the *minimal number* of horizontal and vertical lines required to cover all zeros in the matrix. In case exactly n lines are required, the optimal assignment is given by the zeros covered by the n lines. Otherwise, continue with Step 3.
3) Find the smallest nonzero element in the reduced cost matrix that is not covered by a horizontal or vertical line. Subtract this value from each uncovered element in the matrix. Furthermore, add this value to each element in the reduced cost matrix that is covered by a horizontal and a vertical line. Continue with Step 2.

The computationally difficult part lies in finding the minimum number of lines covering the zero elements (Step 2). The overall algorithm has a complexity of $O(n^3)$. The method described above can directly be applied to assign a set of target locations (tasks) to the individual robots (machines). Here, each entry in the cost matrix can be the length of the path the corresponding robot has to travel to reach the designated target point.

Since the implementation of the Hungarian method described above requires the number of jobs and the number of machines to be equal, we need to slightly adapt the cost matrix computed in that way. This can be achieved by expanding the cost matrix using "dummy machines" (which will result in target locations that are not approached by any of the robots) and by duplicating existing targets. The Hungarian Method is then able to compute the optimal assignment, given the cost matrix.

IV. Map Segmentation

Several researchers investigated the problem of segmenting maps based on the partitioning of a graph [1], [8], [16], [25], [27]. A very popular graph-based representation in this context are *Voronoi Graphs* (VGs) [5]. To compute the Voronoi Graph $G(m) = (V, E)$ of a given map m, we consider the set $O_p(m)$ which contain for each point p in the free-space C of m the set of closest obstacle points. The

Voronoi Graph then is given by the set of points in $O_p(m)$ for which there are at least two obstacle points with an equal minimal distance:

$$V = \{p \in C \mid |O_p(m)| \geq 2\} \quad (1)$$

$$E = \{(p,q) \mid p,q \in V,\ p \text{ adjacent } q \text{ in } m\} \quad (2)$$

For each pair of nodes in $G(m)$ we add an edge if their corresponding points in m are adjacent. The Voronoi Graph can be generated from metric maps of the environment such as occupancy grid maps [6], [25]. In a practical implementation this can be efficiently done by applying the Euclidean distance transformation [18] to an occupancy grid map. This transformation results in a distance map which holds for each grid cell the distance to the closest obstacle. A Voronoi Graph can then be constructed using skeletonization on the distance map. Figure 2 illustrates the process of generating a Voronoi Graph for an example occupancy grid map.

After generating the Voronoi Graph we are now interested in creating a partitioning of the graph into k disjoint sets $V_1, V_2, \ldots, V_k$ with

$$V = \bigcup_{i=1}^{k} V_i \quad (3)$$

such that each cluster of nodes V_i corresponds to a segment we can assign robots to. Thrun *et al.* suggest the graph to be separated at so-called *critical points* [25]. Here, critical points are those nodes in the Voronoi Graph at which the distance to the closest obstacle in the map is a local minimum. This is usually the case in doorways or other narrow passages.

Whereas this approach is able to reliably find doorways, it also generates a lot of false positive candidates in cluttered environments. To eliminate these false positives, we constrain them in the following way: First, critical points have to be nodes of degree 2 (two edges) and second, need to have a neighbor of degree 3 (a junction node). In addition, we require the points to lead from known into unknown areas, since segments which do not contain unknown areas can safely be ignored in an exploration task. To verify this constraint, we compute the distance to the closest reachable unknown cell for each point. This can be done efficiently in a similar way as the computation of the distance map. Figure 3 shows a pruned version of the Voronoi graph and the critical points found by our algorithm. All doorways have been selected as candidates and the number of false positives is much smaller than the number of critical points according to the definition of Thrun *et al.* [25] which includes distance minima in the Euclidean distance transformation within corridors and rooms.

In the practical experiments described in this paper we found that this segmentation technique yields sufficient results and allows to nicely distribute the robots. In unmodified office environments, we can typically reliably separate rooms and segments of a corridor. Other, more complex environments may however suggest more sophisticated segmentation algorithms which rely on hand-labeled training data [2], [8] or more complex reasoning [1], [27].

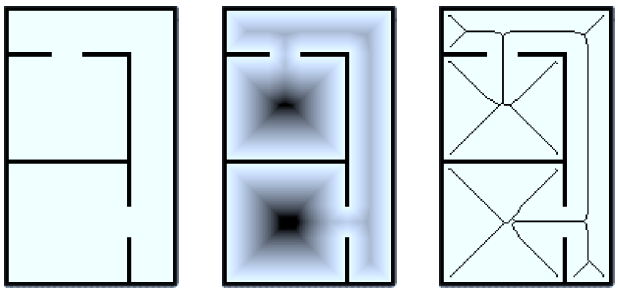


Fig. 2. Generation of the Voronoi Graph. Left: Example grid-map. Center: Map plus distance transform (the darker a point the larger the distance to the closest obstacle). Right: Map and Voronoi Graph generated from the distance transform using skeletonization.

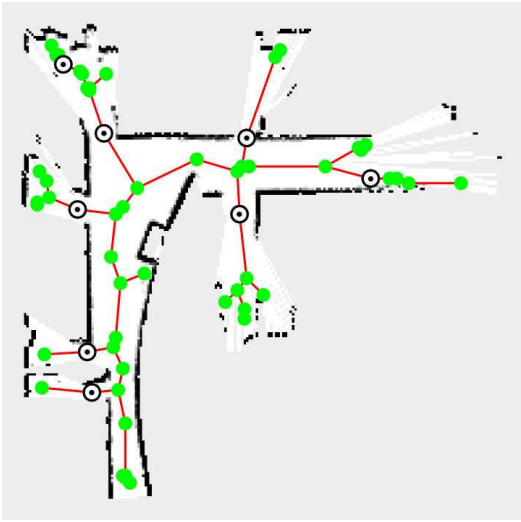


Fig. 3. Example segmentation of a small fraction of an environment. The marked nodes are the candidates for the partitioning of the graph calculated by our approach.

V. Assignment of Robots to Target Areas

Typical approaches to coordinated exploration seek to minimize the time needed to cover the whole environment with the robot's sensors. Therefore, it is often sub-optimal to explore the same (local) area with more than one robot. A cluster of robots which has a serious overlap in the field of view of the robots' sensors does not exploit its full potential. In practice, it is generally much more efficient to explore separate regions of the environment instead. For this reason, it is important to assign robots to exploration targets such that the robots do not get too close to each other during exploration.

Indoor environments are in general structured environments. Buildings are usually divided into rooms which can be reached via corridors. In many cases, it can be a disadvantage to assign more than one robot to one room. The room might, for example, be too small for a second robot to speed up it's exploration even though there initially is more than one frontier in the room. When the room is fully explored, robots might even block each other while trying to leave the room which will result in an increase in exploration time.

In our approach, we assign individual robots to different segments of unexplored space. Segments could be separate rooms, corridors, or parts of larger corridors or rooms. This takes into account the structure of the environment and prevents the forming of inefficient clusters of robots.

Algorithm 1 Target Assignment Using Map Segmentation.

1: Determine segmentation $S = \{s_1, ..., s_n\}$ of map.
2: Determine the set of frontier targets for each segment.
3: Compute for each robot i the cost C_s^i for reaching each map segment $s \in S$.
4: Discount cost C_s^i if robot i is already in segment s.
5: Assign robots to segments using the Hungarian Method.
6: **for** all segments s **do**
7: Assign robot(s) to frontier targets in s w.r.t. path costs using the Hungarian Method.
8: **end for**

Our assignment algorithm is summarized in Algorithm 1. An assignment is determined whenever one of the robots requests a new exploration target. First, a partition of the partial map of the environment is created using the graph-based method described in Section IV. To generate targets within the segments, we then determine the set of frontier cells. The cost C_s^i for reaching segment s with robot i is defined as the expected path cost to the nearest frontier cell within s. This cost is discounted by a constant factor if robot i is already located in segment s. This has the effect that the robots stay in their assigned segment until it is completely explored. After computing the costs of a segment, an assignment is calculated by applying the Hungarian method (see Section III) based on the cost matrix.

The Hungarian method does not assign more than one robot to the same segment unless there are more robots available than there are unexplored segments. To appropriately handle those cases in which multiple robots are assigned to a single segment, we apply a local assignment based on the cost-optimal frontier within a segment. For this reason, our algorithm is equivalent to a purely frontier-based assignment if the environment cannot be partitioned, i.e., there is only one segment.

By assigning robots to separate segments, an appropriate distribution of the robots can be achieved. As we will demonstrate in the experiments, this leads to a significant reduction in exploration time. Instead of aiming at the closest frontier, robots share work more efficiently. A typical office environment, for example, contains corridors and rooms. Using our approach, each of the corridors is explored completely by one of the robots. In this way, the rough structure of the building will quickly be revealed. Meanwhile other robots will be assigned to the rooms reachable from the corridors, one at a time. This behavior does not only appear to be a natural way of exploring an unknown environment, our experiments also revealed that it significantly increases the efficiency of the robot team compared to approaches which ignore the structure of the building.

Fig. 4. Maps used in our simulated experiments: Building 079 of the Freiburg University (top) and Bremen University Cartesium (bottom).

Note that our algorithm is not limited to homogenous teams of robots. Consider the situation in which one particular robot cannot enter a certain part of the environment while another robot can. The assignment algorithm described above can be applied in this case by using modified segment costs $\bar{C}_s^i$ defined as:

$$\bar{C}_s^i = \begin{cases} C_s^i & \text{, if robot } i \text{ can enter segment } s \\ \infty & \text{, otherwise.} \end{cases} \quad (4)$$

VI. Experimental Results

Our approach has been implemented and evaluated using simulated as well as with real teams of robots. The real world experiments were conducted using two ActivMedia Pioneer II robots equipped with a laser range finder with a 180 degrees field of view. For generating the simulation results, we used the Carnegie Mellon Robot Navigation Toolkit. In all our experiments we assumed that the robots share a joint occupancy grid map, which is generated based on the sensor readings of all robots and under the assumption that all positions of the vehicles are known. This map is used for coordination, path planning, and path execution. We also assume that there is a central planning component which can communicate with all robot and can assign exploration targets to them. If there is only a limited communication range, then clusters of robots can be coordinated if one selects one individual planning agent per cluster [13], [22]. The experiments have been designed to verify that our exploration approach leads to significantly shorter exploration time compared to a standard frontier-based approach.

A. Simulation Results

To evaluate our robot coordination algorithm, we simulated teams of robots in various environments. We compared our segmentation-based approach to a frontier-based approach in which each robot is assigned to the closest frontier which has not been assigned to another robot yet. Since this strategy does not consider the structure of the environment, it will in general also assign more than one robot to one room or corridor if they contain more than one frontier.

To eliminate influences from the segmentation algorithm used in the real world experiment, we assumed a given segmentation of the environment into rooms and corridors in our simulation experiments. As mentioned above, such a segmentation could also be reliably generated from the partial map alone.

Figure 4 depicts two maps of real environments used for the simulation (see also real world experiments). Both of them are office environments, one at the University of Freiburg and the other at the University of Bremen. To make the maps more different, we added clutter to the map representing the office environment located at the University of Bremen.

We varied the size of the simulated team from two to six robots (Freiburg map) respectively from two to eight robots (Bremen map). Since the Bremen map is considerably bigger than the Freiburg map, we simulated larger teams of robots there. For each team size, we conducted a series of simulated exploration runs starting from 20 different starting positions.

The results of our experiments can be seen in Figure 5. We measured the runtime gain of our approach which uses the assignment described in Section V compared to the alternative assignment described above. We plotted the runtime gain in percent of the total runtime against the size of the robot team. The error bars in the plots indicate the 95% confidence level. It can be seen that our approach significantly outperforms the approach which does not use a segmentation based assignment.

The runtime gain is bigger for the Cartesium map since this map features several large rooms. This observation can be seen as an indicator as to when our approach will lead to especially good results. Whenever the environment can be divided into reasonably large and separated segments, our technique substantially reduces the overall exploration time.

In general, our strategy assigns one robot to one segment. As soon as there are more robots than segments multiple robots may be assigned to the same segment as mentioned in Section V. For this reason, the runtime gain of our strategy will decrease for large teams of robots in small environments. This can be seen in Figure 5. Note however, that the overall time to complete the mission still gets smaller the more robots are added to the task (the plot only shows the improvement of our approach vs. the frontier-based approach).

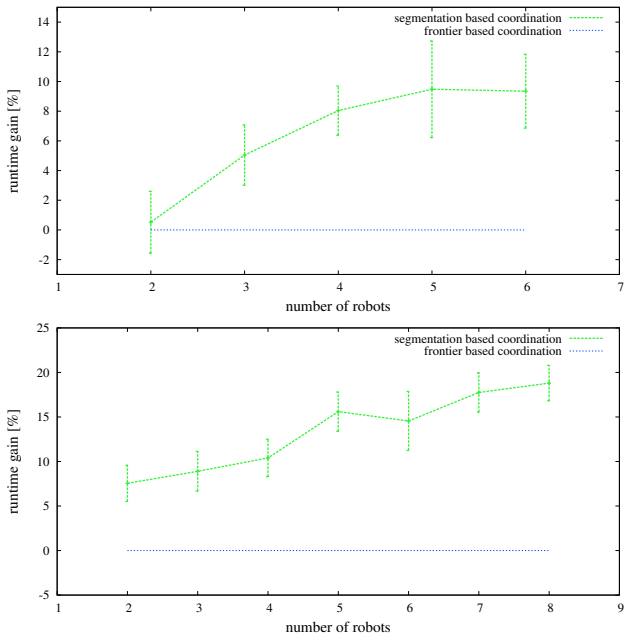


Fig. 5. Exploration time gain of our approach compared to a frontier-based approach for the AIS lab in Freiburg (top) and the Bremen Cartesium (bottom).

Fig. 6. Two robots exploring the AIS laboratory of the University of Freiburg using our coordination approach.

B. Real Robot Experiments

Our coordination algorithm has been evaluated using a team of real robots. For this experiment, we used two identical Pioneer II robots equipped with a laser range finder and a standard laptop-computer. During the experiment both robots were connected via a wireless network. The robot localization was achieved using a standard scan-matching approach. The relative starting poses of the robot were manually set in the beginning. Figure 6 depicts the two robots during their exploration mission.

The experiments were conducted in the lower floor of building 079 of the Freiburg computer science campus. The building has a size of approximately 37m x 14m and consists of numerous office rooms and two long corridors divided by a door.

The team of robots was able to successfully explore the environment using our coordination approach. The result of one of the experiments can be seen in Figure 7. The figure shows the combined map of both robots after the exploration had finished. It also shows the trajectories of both robots during the exploration. The total exploration time was less than nine minutes, each of the robots traveled approximately 120m.

It can be seen that each of the rooms was explored by exactly one of the robots. It can also be seen that both corridors have been explored completely by one of the robots while the other one was exploring rooms reachable from the corridor. Another interesting effect is that the robots did not

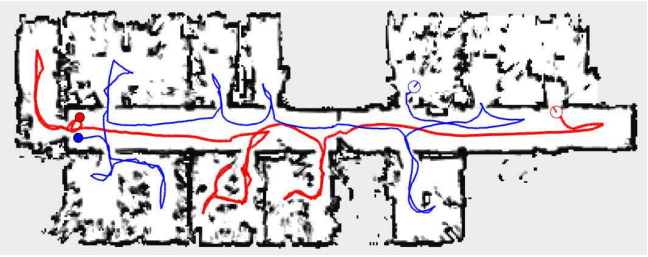


Fig. 7. Resulting map of the real world experiment including the trajectories of the two individual robots.

block each other during the execution of their tasks.

VII. Conclusion

In this paper, we proposed a novel technique for coordinating a team of exploring robots. We use a segmentation of the environment to determine exploration targets for the individual robots. By assigning each robot to a separate segment, a balanced distribution of the robots over the environment is achieved. This leads to a shorter overall exploration time compared to an approach which does not use our segmentation. Thus, our approach reduces the risk of interference between robots and the amount of redundant work. We also introduced an efficient graph-based segmentation technique for partially explored environments. Our approach has been implemented and evaluated in simulation as well as with a team of real robots. The experiments show a significant improvement of the segmentation-based approach compared to a standard frontier-based approach for structured indoor environments. Note that our approach is not limited to our segmentation method. Using a heterogenous team of robots, for example, such a segmentation can be defined based on traversability constraints of the different robots.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A3).

References

[1] P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[2] E. Brunskill, T. Kollar, and N. Roy. Topological mapping using spectral clustering and classification. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, October 2007.

[3] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[4] Y.U. Cao, A.S. Fukunaga, and A.B. Khang. Cooperative mobile robotics: Antecedents and directions. *Journal of Autonomous Robots*, 4(1):7–27, 1997.

[5] H. Choset, , and Burdick J. Sensor-based exploration: The hierarchical generalized voronoi graph. *J. of Robotics Research*, 19(2), 2000.

[6] H. Choset and J. Burdick. Sensor based planning, part i: The generalized voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Nagoya, Japan, 1995.

[7] G. Dudek, M.. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Journal of Autonomous Robots*, 3(4):375–397, 1996.

[8] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In Manuela M. Veloso, editor, *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 2109–2114, 2007.

[9] B.P. Gerkey and M.J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758– 768, 2002.

[10] D. Goldberg and M.J. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. *Journal of Robotics & Autonomous Systems*, 8:637–642, 1997.

[11] D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. Many robots make short work. *AI Magazine*, 18(1):55–64, 1997.

[12] A. Howard, M.J. Matarić, and S. Sukhatme. An incremental deployment algorithm for mobile robot teams. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2849–2854, Lausanne, Switzerland, 2002.

[13] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.

[14] S. Koenig, B. Szymanski, and Y. Liu. Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31:41–76, 2001.

[15] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1):83–97, 1955.

[16] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics & Autonomous Systems*, 8:47–63, 1991.

[17] M.J. Matarić and G. Sukhatme. Task-allocation and coordination of multiple robots for planetary exploration. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 61–70, Budapest, Hungary, 2001.

[18] A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink. *Mathematical Morphology and its Applications to Image and Signal Processing*, chapter A General Algorithm for Computing Distance Transforms in Linear Time, pages 331–340. Kluwer Academic Publishers, 2000.

[19] I. Rekleitis, V. Lee-Shue, A. Peng New, and H. Choset. Limited communication, multi-robot team based coverage. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3462–3468, New Orleans, LA, USA, 2004.

[20] M. Schneider-Fontan and M.J. Matarić. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822, 1998.

[21] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 254–259, Atlanta, GA, USA, 1993.

[22] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, April 2006.

[23] C. Stachniss, G. Grisetti, O. Martínez-Mozos, and W. Burgard. Efficiently learning metric and topological maps with autonomous service robots. *it – Information Technology*, 49(4):232–238, 2007.

[24] C. Stachniss, O. Martínez-Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1692–1697, Orlando, FL, USA, 2006.

[25] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[26] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Second International Conference on Autonomous Agents*, pages 47–53, Minneapolis, MN, USA, 1998.

[27] Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building and planning based on graph partitioning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 803–809, 2006.

[28] R. Zlot, A.T. Stenz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Washington, DC, USA, 2002.

[C13] C. Stachniss, C. Plagemann, A.J. Lilienthal, and W. Burgard. Gas distribution modeling using sparse gaussian process mixture models. In *Proc. of Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

Gas Distribution Modeling using Sparse Gaussian Process Mixture Models

Cyrill Stachniss[1] Christian Plagemann[1] Achim Lilienthal[2] Wolfram Burgard[1]

[1]*Albert-Ludwigs-University Freiburg, Dept. for Computer Science, D-79110 Freiburg, Germany*
[2]*Örebro University, AASS, Dept. of Technology, S-70182 Örebro, Sweden*
{stachnis, plagem, burgard}@informatik.uni-freiburg.de, achim.lilienthal@tech.oru.se

***Abstract*— In this paper, we consider the problem of learning a two dimensional spatial model of a gas distribution with a mobile robot. Building maps that can be used to accurately predict the gas concentration at query locations is a challenging task due to the chaotic nature of gas dispersal. We present an approach that formulates this task as a regression problem. To deal with the specific properties of typical gas distributions, we propose a sparse Gaussian process mixture model. This allows us to accurately represent the smooth background signal as well as areas of high concentration. We integrate the sparsification of the training data into an EM procedure used for learning the mixture components and the gating function. Our approach has been implemented and tested using datasets recorded with a real mobile robot equipped with an electronic nose. We demonstrate that our models are well suited for predicting gas concentrations at new query locations and that they outperform alternative methods used in robotics to carry out in this task.**

***Index Terms*— Gas distribution modeling, gas sensing, Gaussian processes, mixture models**

I. Introduction

Gas distribution modeling has important applications in industry, science, and every-day life. Mobile robots equipped with gas sensors are deployed, for example, for pollution monitoring in public areas [1], surveillance of industrial facilities producing harmful gases, or inspection of contaminated areas within rescue missions.

Although humans have a natural odor sensor, it is hard for us to build a spatial representation of a sensed gas distribution. Building gas distribution maps is actually a challenging task due to the chaotic nature of gas dispersal. The complex interaction of gas with its surroundings is dominated by two physical effects. First, on a comparably large timescale, diffusion mixes the gas with the surrounding atmosphere to achieve a homogeneous mixture of both in the long run. Second, turbulent air flow fragments the gas emanating from a source into intermittent *patches* of high concentration with steep gradients at their edges [16]. Especially this chaotic system of localized patches of gas makes the modeling problem a hard one. In addition to that, gas sensors provide information about a small spatial region only since gas sensor measurements require direct interaction between the sensor surface and the analyze molecules. This makes gas sensing different to perceiving the environment with laser range finders or other popular robotic sensors.

Fig. 1 illustrates actual gas concentration measurements recorded with a mobile robot along a corridor containing a single gas source. The distribution consists of a rather smooth "background" signal and several peaks indicating high gas concentrations. The challenge in gas distribution mapping is to model this background signal while being able to cover also the areas of high concentration and their sharp boundaries. Since performing measurements is a comparably costly operation, one is also interested in reducing the number of samples needed to build a representation. It is important to note that the noise is dominated by the large fluctuations of the instantaneous gas distribution and not by the electronic noise of the gas sensors. From a probabilistic point of view, the task of modeling a gas distribution can be described as finding a model that best explains the observations and that is able to accurately predict new ones. Thus, the data likelihood in combination with cross validation is the standard criterion to evaluate such a model.

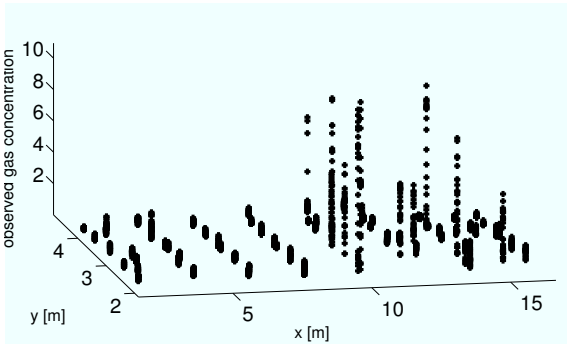


Fig. 1. Gas concentration measurements acquired by a mobile robot in a corridor. The distribution consists of a rather smooth "background" signal and several peaks indicating high gas concentrations.

Simple spatial averaging, which represents a straightforward approach to the modeling problem, disregards the different nature of the background noise and the peaks resulting from areas of high gas concentrations and, thus, achieves only limited prediction accuracy. On the other hand, precise physical simulation of the gas dynamics in the environment would require immense computational resources as well as precise knowledge about the physical conditions, which is not known in most practical scenarios.

To achieve a balance between model accuracy and tractability, we treat gas distribution mapping as a two-dimensional regression problem. We derive a solution by means of a sparse mixture model of Gaussian process experts [21] that is able to handle both physical phenomena highlighted above.

Formally, we interpret gas sensor measurements obtained from static sensors or from a mobile robot at locations as noisy samples from a time-constant distribution. This implies that the gas distribution in fact exhibits a time-constant structure, an assumption that is often made in unventilated and un-populated indoor environments [22].

While existing approaches to gas distribution mapping such as local averaging [6, 11], kernel extrapolation techniques [7], or standard GP models represent the average concentration per location only, our mixture model explicitly distinguishes different components of the distribution, i.e., concentration layers varying smoothly due to dispersion processes versus those containing localized *patches* of gas. This leads to a more accurate prediction of the gas concentration. Our model actually allows us to do both, computing the average gas concentration per location (as existing models supply) as well as the multi-modal predictive densities.

The contribution of this paper is a novel approach that learns gas distribution models from sensor data using a sparse Gaussian process mixture model. As a by-product, we present an algorithm that learns a GP mixture model and simultaneously reduces the model complexity in order to achieve an efficient representation even for large data sets. Our technique provides gas concentration estimates for each location in space and also the corresponding predictive uncertainties. The mixture model allows us to improve the gas concentration estimate close to the boundaries and in areas with high gas concentration compared to standard models. As we will demonstrate in experiments carried out with a real robot, our model has a lower mean squared error and a higher data likelihood than other methods and thus allows to more accurately predict gas concentration at query locations.

This paper is organized as follows. After a discussion of related work, we introduce in Sec. III Gaussian processes for regression. Then, Sec. IV explains our approach to learn a sparse GP mixture to model gas distributions from observations. Finally, we present the experimental evaluation of our work with a real mobile robot.

II. Related Work

A straightforward method to create a representation of the time-averaged concentration field is to perform measurements over a prolonged time with a grid of gas sensors. Equidistant gas sensor locations can be used to represent the average concentration values directly on a grid map. This method, though with partially simultaneous measurements, was applied by Ishida *et al.* [6]. A similar method was used in [11] but instead of the average concentration, the peak concentration observed during a sampling period of 20 s was considered to create the map.

Consecutive measurements with a single sensor and time-averaging over 2 minutes for each sensor location were used by Pyk *et al.* [12] to create a map of the distribution of ethanol. Methods, which aim at determining a map of the instantaneous gas distribution from successive concentration measurements, rely on the assumption of a time-constant distribution profile, i.e., uniform delivery and removal of the analyze gas and stable environmental conditions. Thus, the experiments of Pyk *et al.* were performed in a wind tunnel with a constant airflow and a uniform gas source. To make predictions at locations different from the measurement points, they apply bi-cubic interpolation in the case of equidistant measurements and triangle-based cubic filtering in the case where the measurement points are not equally distributed [12]. A problem with these interpolation methods is that there is no means of "averaging out" instantaneous response fluctuations at measurement locations. Even if response values were measured very close to each other, they will appear independently in the gas distribution map with interpolated values in-between. Consequently, interpolation maps tend to get more and more jagged while new measurements are added [8].

Histogram methods take the spatial correlation of concentration measurements into account because of the implicit extrapolation on the measurements by the quantization into histogram bins. Hayes *et al.* [5] suggest a two-dimensional histogram where the bins contain the accumulated number of "odor hits" received in the corresponding area. Odor hits are counted whenever the response level of a gas sensor exceeds a defined threshold. In addition to the dependency of the gas distribution map on the selected threshold, a problem with using only binary information from the gas sensors is that much useful information about fine gradations in the average concentration is discarded. A further disadvantage of histogram methods for gas distribution modeling is their dependency on the bin size and that they require perfectly even coverage of the inspected area.

Kernel extrapolation gas distribution mapping, which can be seen as an extension of histogram methods, was introduced by Lilienthal and Duckett [7]. Spatial integration is carried out by convolving sensor readings and modeling the information content of the point measurements with a Gaussian kernel. As discussed in [8], this method has also an analogy with non-parametric estimation of density functions using a Parzen window method.

Model-based approaches as in Ishida *et al.* [6] infer the parameters of an analytical gas distribution model from the measurements. They depend crucially on the underlying model. Complex numerical models based on fluid dynamics simulations are computationally expensive and depend sensitively on accurate knowledge of the state of the environment (boundary conditions) which is not available in practical situations. Simpler analytical models, on the other hand, often rest on rather unrealistic assumptions and are of course only applicable for situations in which the model assumptions hold. Model-based approaches also rely on well-calibrated gas sensors and an established understanding of the sensor-environment interaction.

The majority of approaches proposed in the literature create a two-dimensional representation and represent time-constant structures in the gas distribution. Also the effort (either in terms of time consumption or the number of sensors) of the model-free approaches to converge to a stable representation, scales quadratically with the size of the environment. None of the approaches suggested so far models the variance together with the time-average of the concentration field.

In contrast to those approaches, we apply Gaussian processes in a mixture model setting to learn probabilistic gas distribution maps. GPs allow us to model the dependency between nearby locations by means of a covariance function. They enable us to make predictions at locations not observed so far and do not only provide the mean gas distribution but also a predictive variance. Our mixture model can furthermore model sharp boundaries of areas with high gas concentration.

Gaussian processes (GPs) are a non-parametric method frequently used to solve regression and classification problems [13]. A drawback of the standard GP approach is its computational complexity. However, several methods for learning sparse GP models [18, 19] have been presented that overcome this limitation and lead to a near-linear complexity [19]. Tresp [21] introduced a mixture model of GP experts to better deal with spatially varying properties in the data. Extensions of this technique using infinite mixtures have been proposed by Rasmussen and Ghahramani [15] and Meeds and Osindero [9].

GPs have already received considerable attention within the robotics community. Schwaighofer *et al.* [17] introduced a positioning system for cellular networks based on Gaussian processes. Brooks *et al.* [2] proposed a Gaussian process model in the context of appearance-based localization with an omni-directional camera. Ferris *et al.* [3] applied Gaussian processes to locate a mobile robot from wireless signal strength. Related Bayesian regression approaches have been also followed for example by Ting *et al.* [20] to identify rigid body dynamics and Grimes *et al.* [4] to learn imitative whole-body motions.

III. Gaussian Processes for Regression

The general gas distribution mapping problem, given a set of gas concentration measurements $y_{1:n}$ acquired at locations $\mathbf{x}_{1:n}$, is to learn a predictive model $p(y_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, y_{1:n})$ for gas concentrations y_* at a query location $\mathbf{x}_*$. We address this estimation problem as a regression problem. Gaussian processes (GPs) offer a flexible way of solving such regression problems [13]. GPs are a "non-parametric" method, since no parametric form of the underlying function $\mathbf{x} \mapsto y$ is assumed. The model is represented directly using the given training data. GPs can be seen as a generalization of the Gaussian probability distribution to a distribution over functions. A GP for real-valued functions f is defined by a mean function $m(\cdot)$ and a covariance function $k(\cdot,\cdot)$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \quad (1)$$

$$k(\mathbf{x}_p, \mathbf{x}_q) = \mathbb{E}[(f(\mathbf{x}_p) - m(\mathbf{x}_p))(f(\mathbf{x}_q) - m(\mathbf{x}_q))]. \quad (2)$$

In the following, we set $m(\mathbf{x}) = 0$ for simplicity of notation and apply the squared exponential covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}\frac{|\mathbf{x}_p - \mathbf{x}_q|^2}{l^2}\right). \quad (3)$$

Observations y obtained from the process are assumed to be affected by Gaussian noise, $y \sim \mathcal{N}(m(\mathbf{x}), \sigma_n^2)$. The variables $\Phi = \{\sigma_f, l, \sigma_n\}$ are the so-called hyperparameters of the process which have to be learned from data.

Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of training data where $\mathbf{x}_i \in \mathbb{R}^d$ are the inputs and $y_i \in \mathbb{R}$ the targets, the goal in regression is to predict target values $y_* \in \mathbb{R}$ at a new input point $\mathbf{x}_*$. Let $X = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ be the $n \times d$ matrix of the inputs and X_* be defined analogously for multiple test data points. In the GP model, any finite set of samples is jointly Gaussian distributed

$$\begin{bmatrix} \mathbf{y} \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right), \quad (4)$$

where $K(\cdot,\cdot)$ refers to the matrix with the entries given by the covariance function $k(\cdot,\cdot)$ and $\mathbf{y}$ the vector of the (observed) targets y_i. To actually make predictions at X_*, we obtain for the predictive mean

$$\bar{f}(X_*) := \mathbb{E}[f(X_*)] = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y} \quad (5)$$

and for the (noise-free) predictive variance

$$\mathbb{V}[f(X_*)] = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1}K(X,X_*), \quad (6)$$

where I is the identity matrix. The corresponding (noisy) predictive variance for an observation $\mathbf{y}_*$ can be obtained by adding the noise term σ_n^2 to the individual components of $\mathbb{V}[f(X_*)]$.

GPs are a sound mathematical framework with many practical applications. The standard GP model as described above, however, has two major limitations in our problem domain. First, the computational complexity is high, since to compute the predictive variance given in Eq. (6), one needs to invert the matrix $K(X,X) + \sigma_n^2 I$, which introduces a complexity of $\mathcal{O}(n^3)$ where n is the number of training examples. As a result, an important issue for GP-based solutions to practical problems is the reduction of this complexity. This can, as we will show in Sec. IV, be achieved by artificially limiting the training data set in a way that introduces small loss in the data likelihood of the whole training set while at the same time minimizing the runtime. As a second limitation, the standard GP model generates a *uni-modal* distribution per input location $\mathbf{x}$. This assumption hardly fits our application domain in which a relatively smooth "background" signal is typically *mixed* with high-concentration "packets" of gas. In the following, we address this issue by deriving a *mixture model* of Gaussian processes.

A. Mixtures of Gaussian Process Models

The GP mixture model [21] constitutes a locally weighted sum of several Gaussian process models. For simplicity of notation, we consider without loss of generality the case of single predictions only ($\mathbf{x}_*$ instead of X_*). Let $\{\mathcal{GP}_1, \ldots, \mathcal{GP}_m\}$ be a set of m Gaussian processes representing the individual mixture components. Let $P(z(\mathbf{x}_*) = i)$ be the probability that $\mathbf{x}_*$ is associated with the i-th component of the mixture. Let $\bar{f}_i(\mathbf{x}_*)$ be the mean prediction of the $\mathcal{GP}_i$ at $\mathbf{x}_*$. The likelihood of observing y_* in such as model is thus given by

$$h(\mathbf{x}_*) := p(y_* \mid \mathbf{x}_*) = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i) \cdot \mathcal{N}_i(y_*; \mathbf{x}_*), \quad (7)$$

where we define $\mathcal{N}_i(y; \mathbf{x})$ as the Gaussian density function with mean $\bar{f}_i(\mathbf{x})$ and variance $\mathbb{V}[f_i(\mathbf{x})] + \sigma_n^2$ evaluated at

y. One can sample from such a mixture by first sampling the mixture component according to $P(z(\mathbf{x}_*) = i)$ and then sampling from the corresponding Gaussian. For some applications such as information-driven exploration missions, it is practical to estimate the mean and variance for this multi-modal model. The mean $\mathbb{E}[h(\mathbf{x}_*)]$ of the mixture model is given by

$$\bar{h}(\mathbf{x}_*) := \mathbb{E}[h(\mathbf{x}_*)] = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i) \cdot \bar{f}_i(\mathbf{x}_*) \tag{8}$$

and the corresponding variance is computed as

$$\begin{aligned} \mathbb{V}[h(\mathbf{x}_*)] \quad &= \quad \sum_{i=1}^{m} (\mathbb{V}[f_i(\mathbf{x}_*)] + (\bar{f}_i(\mathbf{x}_*) - \bar{h}(\mathbf{x}_*))^2) \\ &\quad \cdot P(z(\mathbf{x}_*) = i). \end{aligned} \tag{9}$$

IV. Learning the Mixture Model from Data

Given a training set $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n}$ of gas concentration measurements y_j and the corresponding sensing locations $\mathbf{x}_j$, the task is to jointly learn the assignment $z(\mathbf{x}_j)$ of data points to mixture components and, given this assignment, the individual regression models $\mathcal{GP}_i$. Tresp [21] describes an approach based on Expectation Maximization (EM) for solving this task. We take his approach, but also seek to minimize the model complexity to achieve a computationally tractable model even for large training data sets $\mathcal{D}$. This is of major importance in our application, since typical gas concentration data sets easily exceed $n = 1000$ data points and the standard GP model (see Sec. III) is of cubic complexity $\mathcal{O}(n^3)$. Different solutions have been proposed for lowering this upper bound, such as dividing the input space into different regions and solving these problems individually or the usage of the so called *sparse GPs*. Sparse GPs [18, 19] use a reduced set of inputs to approximate the full space. This new set can be either a subset of the original inputs [18] or a set of new *pseudo-inputs* [19] which are obtained using an optimization procedure. This reduces the complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2)$ with $m \ll n$, which in practice results in a nearly linear complexity. In this section, we describe a greedy forward-selection algorithm integrated into the EM-learning procedure which achieves a sparse mixture model while also maximizing the data likelihood of the whole training set $\mathcal{D}$.

A. Initializing the Mixture Components

In a first step, we subsample n_1 data points and learn a standard GP for this set. This model $\mathcal{GP}_1$ constitutes the first mixture component. To cover areas of gas concentrations that are poorly modeled by this initial model, we learn an "error GP" which models the absolute differences between a set of target values and the predictions of $\mathcal{GP}_1$. We then sample points according to the error GP and use them as the initialization for the next mixture component. In this way, the new mixture is initialized with the data points that are poorly approximated by the first one. This process is continued until the desired number m of model components is reached. For typical gas modeling scenarios, we found that two mixture components are often sufficient to achieve good results. In our experiments, the converged mixture models nicely reflect the bi-modal nature of gas distributions, having one smooth "background" component and a layer of locally concentrated structures as outlined in the introduction of this paper.

B. Iterative Learning via Expectation-Maximization

The Expectation Maximization (EM) algorithm can be used to obtain a maximum likelihood estimate when hidden and observable variables need to be estimated. It consists of two steps, the so-called estimation (E) step and the maximization (M) step which are executed alternately.

In the E-step, we estimate the probability $P(z(\mathbf{x}_j) = i)$ that the data point j corresponds to the model i. This is done by computing the marginal likelihood of each data point for all models individually. Thus, the new $P(z(\mathbf{x}_j) = i)$ is computed given the previous one as

$$P(z(\mathbf{x}_j) = i) \quad \leftarrow \quad \frac{P(z(\mathbf{x}_j) = i) \cdot \mathcal{N}_i(y_j; \mathbf{x}_j)}{\sum_{k=1}^{m} P(z(\mathbf{x}_j) = k) \cdot \mathcal{N}_k(y_j; \mathbf{x}_j)} \tag{10}$$

In the M-step, we update the components of our mixture model. This is achieved by integrating the probability that a data point belongs to a model component into the individual GP learning steps (see also [21]). This is achieved by modifying Eq. (5) to

$$\bar{f}_i(X_*) \quad = \quad K(X_*, X)[K(X, X) + \Psi^i]^{-1}\mathbf{y}, \tag{11}$$

where Ψ^i is a matrix with

$$\Psi^i_{jj} \quad = \quad \frac{\sigma_n^2}{P(z(\mathbf{x}_j) = i)} \tag{12}$$

and zeros in the off-diagonal elements. Eq. (6) is updated respectively. The matrix Ψ^i allows us to consider the probabilities that the individual inputs belong to the corresponding components. The contribution of an unlikely data point to a model is reduced by increasing the data point specific noise term. If the probability, however, is one, only σ_n^2 remains as in the standard GP model.

Learning a GP model also involves the estimation of its hyperparameters $\Phi = \{\sigma_f, l, \sigma_n\}$. To estimate them for $\mathcal{GP}_i$, we first apply a variant of the hyperparameter heuristic used by Snelson and Ghahramani [19] in their open-source implementation. We extended it to incorporate the correspondence probability $P(z(\mathbf{x}_k) = i)$ into this initial guess

$$l \quad \leftarrow \quad \max_{\mathbf{x}_j} \; P(z(\mathbf{x}_j) = i) \; ||\mathbf{x}_j - \bar{\mathbf{x}}|| \tag{13}$$

$$\sigma_f^2 \quad \leftarrow \quad \frac{\sum_{j=1}^{n} P(z(\mathbf{x}_j) = i) \; (y_j - \mathbb{E}[y])^2}{\sum_{j=1}^{n} P(z(\mathbf{x}_j) = i)} \tag{14}$$

$$\sigma_n^2 \quad \leftarrow \quad 0.25 \cdot \sigma_f^2 \; , \tag{15}$$

where $\bar{\mathbf{x}}$ refers to the weighted mean of the inputs—each $\mathbf{x}_j$ having a weight of $P(z(\mathbf{x}_j) = i)$.

To optimize the hyperparameters further given this initial estimate, one could apply, for example, Rasmussen's conjugate-gradient–based optimization technique [14] to minimize the negative log marginal likelihood. In our experiments, however, this approach lead to serious overfitting and we therefore resorted to cross validation-based optimization. Concretely, we

randomly sample the hyperparameters and evaluate the model accuracy according to Sec. IV-B on a separate validation set. As a sampling strategy, we draw in each even iteration new parameters from an uninformed prior and in each odd iteration, we improve the current best parameters Θ' by sampling from a Gaussian with mean Θ'. The standard deviation of that Gaussian decreases with the iteration. In our experiments, this strategy found appropriate hyperparameters quickly while significantly reducing the risk of overfitting.

C. Learning the Gating Function

In our mixture model, the gating function defines for each data point the likelihood that it belongs to the individual mixture components. The EM algorithm learns these assignment probabilities for all inputs $\mathbf{x}_j$, maximizing the overall data likelihood. These learned hidden variables are then used to estimate the assignment at an unknown location $\mathbf{x}_*$ by means of regression. Concretely, we learn a gating GP for each component i that uses the $\mathbf{x}_j$ as inputs and the $z(\mathbf{x}_j)$ obtained from the EM algorithm as targets. Let $\bar{f}_i^z(\mathbf{x})$ be the prediction of z for $\mathcal{GP}_i$. Given this set of m GPs, we can compute the correspondence probability for a new test point $\mathbf{x}_*$ as

$$P(z(\mathbf{x}_*) = i) \quad = \quad \frac{\exp(\bar{f}_i^z(\mathbf{x}_*))}{\sum_{j=1}^{m} \exp(\bar{f}_j^z(\mathbf{x}_*))}. \qquad (16)$$

D. Illustrating Example

We have specified all quantities that are needed to model gas distributions with sparse Gaussian process mixture models. To summarize the approach, we use a a simple, simulated, one-dimensional example.

The first part of the data points where uniformly distributed around a y value of 2 while the second part was generated with higher noise at two distinct locations. The left image of Fig. 2 depicts the standard GP learned from the input data and the right one the resulting error GP. Based on the error GP, a second mixture component is initialized and used as the input to the EM algorithm.

The individual images in Fig. 3 illustrate the iterations of the EM algorithm. They depict the two components of the mixture model. After convergence, the gating function is learned using the hidden variables reported by the EM algorithm. The learned gating function is depicted in the left image of Fig. 4 and the final GP mixture model is shown in the right image. It is obvious that this model is a better representation of the distribution than the standard GP model shown in the left image of Fig. 2 (averaged negative log likelihood of -1.70 vs. -0.24).

V. Experiments

We carried out pollution monitoring experiments in which the robot followed a predefined sweeping trajectory covering the area of interest. Along its path, the robot was stopped at a pre-defined set of grid points to carry out measurements on the spot between 10 s (outdoors) and 30 s (indoors). The spacing between the grid points was set to values between 0.5 m to 2.0 m depending on the topology of the available space. The

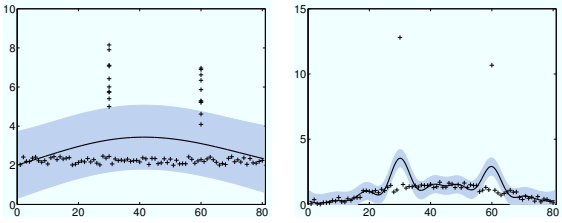


Fig. 2. Left: The standard GP used to initialize the first mixture component. Right: The error GP used to initialize the next mixture component.

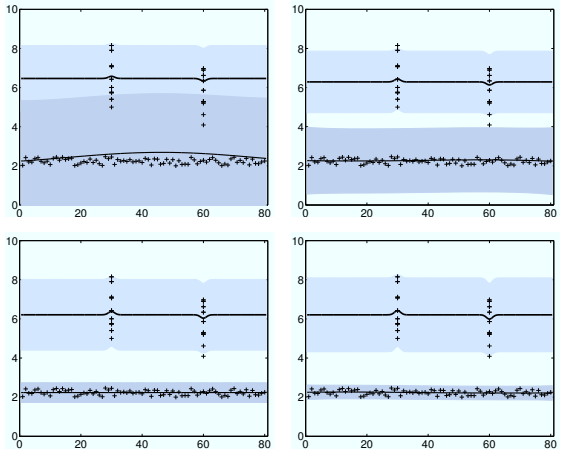


Fig. 3. Components during different iterations of the EM algorithm.

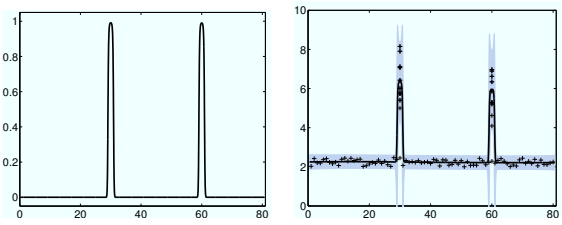


Fig. 4. Left: The learned gating function. Right: Resulting distribution of the GP mixture model.

sweeping motion was performed twice in opposite directions and the robot was driven at a maximum speed of 5 cm/s in between the stops (to reduce the risk of turbulent air flow due to the motion of the robot). The gas source was a small cup filled with ethanol.

Apart from a SICK laser range scanner used for pose correction, the robot was equipped with an electronic nose and an anemometer. The electronic nose comprises six Figaro gas sensors (2 $\times$ TGS 2600, TGS 2602, TGS 2611, TGS 2620, TGS 4161) enclosed in an aluminum tube. This tube is horizontally mounted at the front side of the robot (see also Fig. 5). The electronic nose is actively ventilated through a fan that creates a constant airflow towards the gas sensors. This lowers the effect of external airflow and the movement of the robot on the sensor response.

Note that in this work, we concentrate only on the gas concentration measurements and do not consider the pose uncertainty of the vehicle. One can apply one of the various SLAM systems available to account for the uncertainty in the robot's pose.

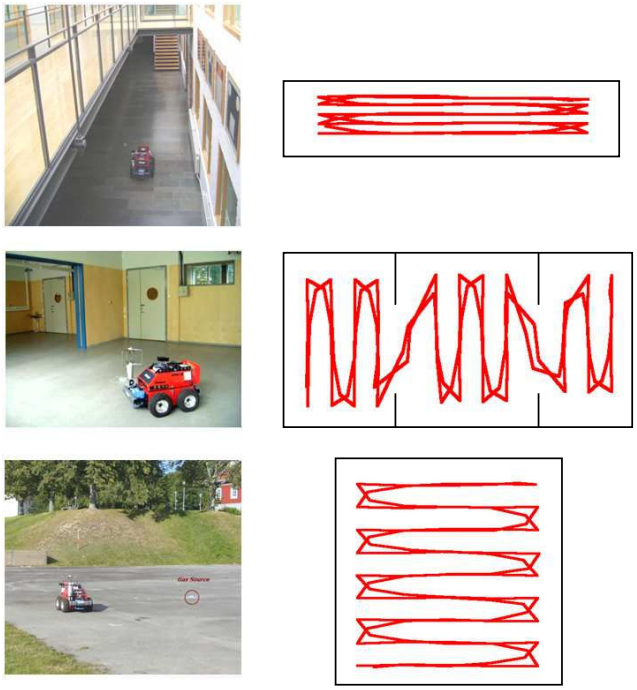


Fig. 5. Pictures of the robot inspecting three different environments as well as the corresponding sweeping trajectories.

TABLE I

AVERAGED NEGATIVE LOG LIKELIHOODS OF TEST DATA POINTS GIVEN THE DIFFERENT MODELS

Dataset	GP	GPM	GPM avg
3-rooms	-1.22	-1.54	-1.50
corridor	-0.98	-1.60	-1.58
outdoor	-1.01	-1.77	-1.69

Three environments with different properties have been selected for the pollution monitoring experiments. The first experiment (*3-rooms*) was carried out in an enclosed indoor area that consists of three rooms which are separated by slightly protruding walls in between them. The area covered by the path of the robot is approximately $14\times6\,\mathrm{m}^2$. There is very little exchange of air with the "outer world" in this environment. The gas source was placed in the central room and all three rooms were monitored by the robot. The second location was a part of a *corridor* with open ends and a high ceiling. The area covered by the trajectory of the robot is approximately $14\times2\,\mathrm{m}^2$. The gas source was placed on the floor in the middle of the investigated corridor segment. Finally, an *outdoor* scenario was considered. Here, the experiments were carried out in an $8\times8\,\mathrm{m}^2$ region that is part of a much bigger open area.

We used the raw sensor readings in all three environments and applied our approach to learn gas distribution models. In the experiments shown here, the robot moved through the environment twice. Therefore, we used the first run for learning the model and the second one for evaluating it. For a comparison with our technique, we also computed a gas distribution model using a standard GP. We furthermore compared our mean estimates to the one of the grid-based method with interpolation and the kernel extrapolation technique.

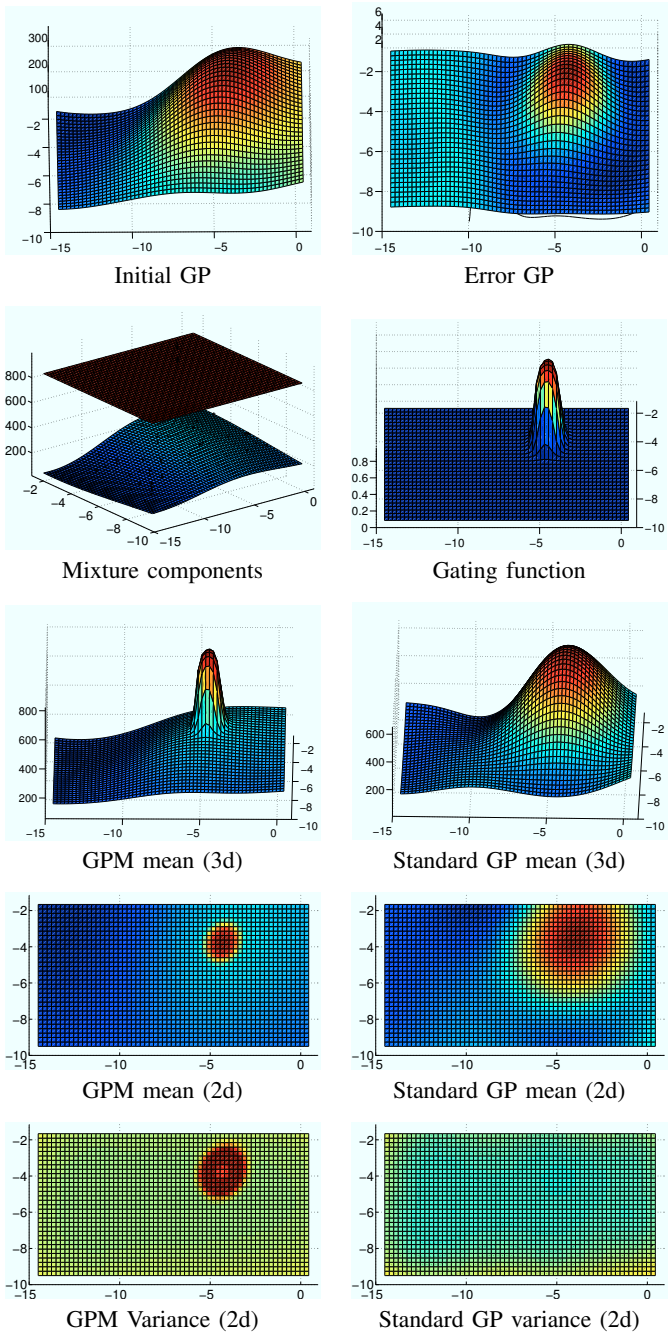


Fig. 6. The 3-rooms dataset with one ethanol gas source in the central room. The room structure itself is not visualized here. In all plots, blue represents low, yellow reflect medium, and red refers to high values.

Fig. 6 depicts the learned models for the 3-room dataset. The left plot in the first row illustrates the mean prediction for the standard GP on the sub-sampled training set which serves as the first mixture component. The right image depicts the error GP representing the differences between the initial prediction and a set of observations. Based on the error GP, a new mixture component is initialized and the EM algorithm is carried out. After convergence, the gating function is learned based on the hidden variables reported by the EM (right image, second row). The left image in the third row shows the final mean prediction of our mixture model. As can be seen, the "background" distribution is smoothly modeled while at the

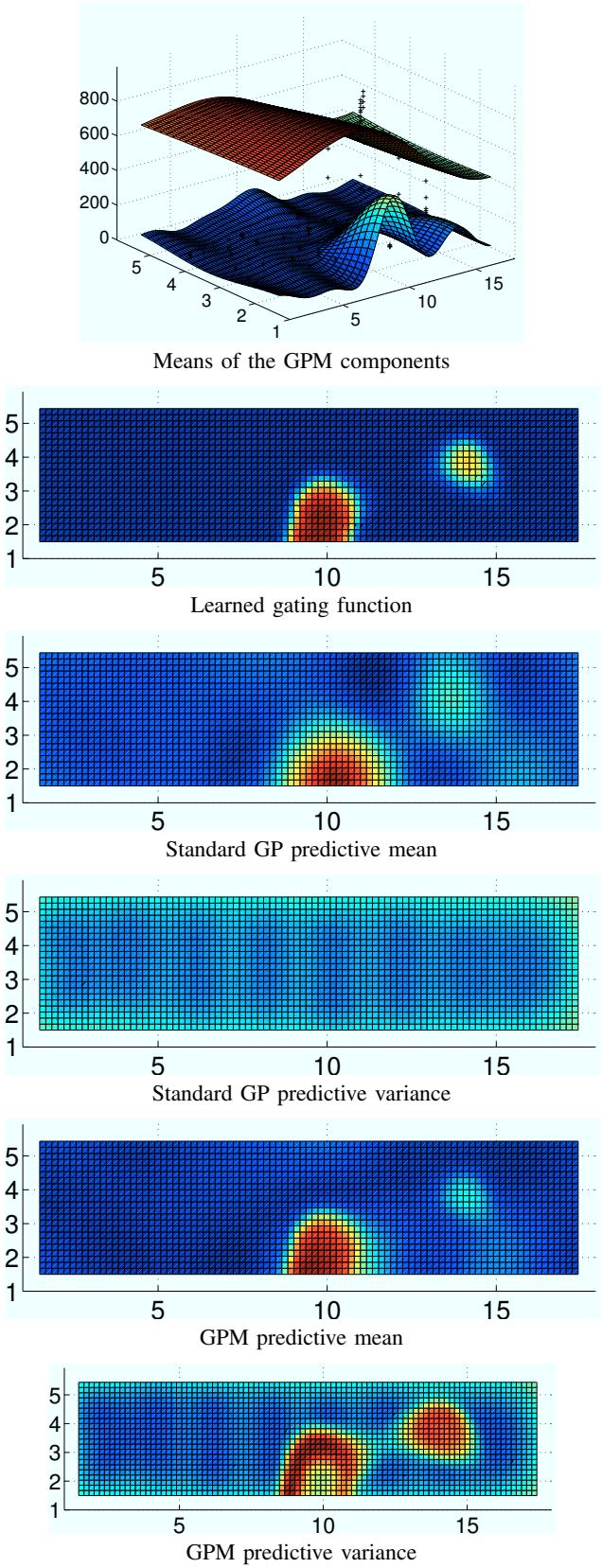


Fig. 7. Models learned from concentration data recorded in the corridor environment (see Fig. 1 for the raw data). The gas source was placed at the location 10, 3. The standard GP and our GPM model provide similar mean estimates. Our approach, however, provides a better predictive uncertainty and thus a higher likelihood given the test data (see Tab. I).

same time the gas concentration peak close to the gas source has a sharp boundary. In contrast to this, the standard GP learned using the same data is unable to provide an appropriate estimate since the area around the peak is to smoothed too much.

Tab. I summarizes the negative log likelihoods of the test data (second part of the dataset) given our mixture model as well as the standard GP model. We provide two likelihoods for our model, the one given in Eq. (7) (called 'GPM' in the table) and the one computed based on the averaged prediction specified in Eq. (8) and Eq. (9) (called 'GPM avg'). As can be seen, our GPM method outperforms the standard GP model in all our experiments since it provides the best data likelihood. Note that we repeated the experiment 10 times and the t-test shows that the results are significant.

By considering the 2d plots in the last two rows of Fig. 6, two reasons for this fact can be observed easily. First, as already mentioned before, the standard GP smoothes too much in the area close to the gas source while this smoothing is fine for the rest of the scene. Second, the variance around the source is too small (standard GPs assumes constant noise for all inputs).

In the corridor experiment, the area of high gas concentration was mapped appropriately also by the standard GP, but again the variance was too small close to the area of high gas concentration. This can be observed by considering Fig. 7. In contrast to this, our GPM model provides a high variance in this area – which actually models the observations in a more precise way. Similar results are obtained in the outdoor dataset. Mean and variance predictions of the standard GP and our model are provided in Fig. 9.

In all our experiments, we limited the number of data points in the reduced input set to $n_1 = 100$ (taken from the first part of the datasets). The datasets itself contained between 2,500 and 3,500 measurements so our model was able to make accurate predictions with less than 5% of the data. Matrices of that size can be easily inverted and as a result the overall computation time to learn our model including cross validation using unoptimized Matlab code on a notebook computer takes around 1 minute for all datasets shown above.

Finally, we compared the mean estimates of our mixture model to the results obtained with the method of Lilienthal and Duckett [7] as well as with the standard approach of using a grid in combination with interpolation. The results of this comparison is shown in Fig. 8. As can be seen, our method outperforms both alternative methods.

VI. Conclusions

In this paper, we considered the problem of modeling gas distributions from sensor measurements by means of sparse Gaussian process mixture models. Gaussian processes are an attractive modeling technique in this context since they do not only provide a gas concentration estimate for each point in the space but also the predictive uncertainty. Our approach learns a GP mixture model and simultaneously decreases the model complexity by reducing the training set in order to achieve an efficient representation even for a large number of observations. This overcomes the major drawback of GPs, their high

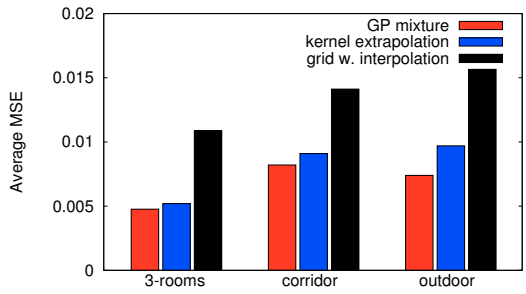


Fig. 8. Mean squared error of the GP mixture model mean and the kernel extrapolation technique and the grid approximation with interpolation.

computational complexity. The mixture model allows us to explicitly distinguish the different components of the spatial gas distribution, namely areas of high gas concentration from the smoothly varying background signal. This improves the accuracy of the gas concentration prediction.

Our method has been implemented and tested using gas sensors mounted on a real robot. With our method, we obtain gas distribution models that better explain the sensor data compared to techniques such as the standard GP regression for gas distribution mapping. Our approach and the one of Lilienthal and Duckett [7] provide similar mean gas concentration estimates, their approach as well as the majority of techniques in the field, however, lack the ability of estimating their predictive uncertainties.

Despite this encouraging results, there is space for further optimizations. Considering non-stationary kernels [10] might further improve the estimates or might serve as an alternative to explictly modeling mixtures. In addition, we are currently exploring the possibility to model the diffusion in high concentration areas by smoothing the gating function over time.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8 as well as by the EC under contract number FP6-IST-34120-muFly and FP6-2005-IST-6-RAWSEEDS.

References

[1] DustBot - Networked and Cooperating Robots for Urban Hygiene. http://www.dustbot.org.

[2] A. Brooks, A. Makarenko, and B. Upcroft. Gaussian process models for sensor-centric robot localisation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[3] B. Ferris, D. Haehnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proceedings of Robotics: Science and Systems*, 2006.

[4] D. Grimes, R. Chalodhorn, and R. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Proceedings of Robotics: Science and Systems*, 2006.

[5] A.T. Hayes, A. Martinoli, and R.M. Goodman. Distributed Odor Source Localization. *IEEE Sensors Journal, Special Issue on Electronic Nose Technologies*, 2(3):260–273, 2002.

[6] H. Ishida, T. Nakamoto, and T. Moriizumi. Remote Sensing of Gas/Odor Source Location and Concentration Distribution Using Mobile System. *Sensors and Actuators B*, 49:52–57, 1998.

[7] A. Lilienthal and T. Duckett. Building Gas Concentration Gridmaps with a Mobile Robot. *Robotics and Autonomous Systems*, 48(1):3–16, 2004.

[8] A. Lilienthal, A. Loutfi, and T. Duckett. Airborne Chemical Sensing with Mobile Robots. *Sensors*, 6:1616–1678, 2006.

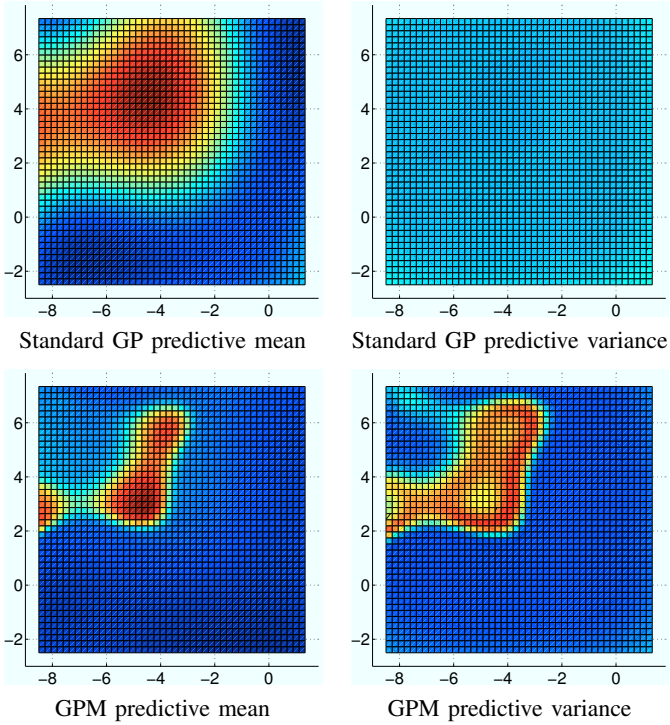


Fig. 9. Outdoor dataset of a 8 m by 8 m area with an ethanol source in the center and airflow. approximatively from south-east to north-west.

[9] E. Meeds and S. Osindero. An alternative infinite mixture of gaussian process experts. In *Advances in Neural Information Processing Systems*, 2006.

[10] C. Plagemann, K. Kersting, and W. Burgard. Nonstationary gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*, Antwerp, Belgium, 2008.

[11] A.H. Purnamadjaja and R.A. Russell. Congregation Behaviour in a Robot Swarm Using Pheromone Communication. In *Proc. of the Australian Conf. on Robotics and Automation*, 2005.

[12] P. Pyk et al. An Artificial Moth: Chemical Source Localization Using a Robot Based Neuronal Model of Moth Optomotor Anemotactic Search. *Autonomous Robots*, 20:197–213, 2006.

[13] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[14] C.E. Rasmussen. Minimize. http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize, 2006.

[15] C.E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems 14*, 2002.

[16] P.J.W. Roberts and D.R. Webster. Turbulent Diffusion. In H. Shen, A. Cheng, K.-H. Wang, M.H. Teng, and C. Liu, editors, *Environmental Fluid Mechanics - Theories and Application*. ASCE Press, Reston, Virginia, 2002.

[17] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. Gpps: A gaussian process positioning system for cellular networks. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2003.

[18] A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In *NIPS*, pages 619–625, 2000.

[19] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1259–1266, 2006.

[20] J. Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Proceedings of Robotics: Science and Systems*, 2006.

[21] V. Tresp. Mixtures of gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2000.

[22] M. Wandel, A. Lilienthal, T. Duckett, U. Weimar, and A. Zell. Gas distribution in unventilated indoor environments inspected by a mobile robot. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 507–512, 2003.

[C14] C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

How to Learn Accurate Grid Maps with a Humanoid

Cyrill Stachniss Maren Bennewitz Giorgio Grisetti Sven Behnke Wolfram Burgard

***Abstract*— Humanoids have recently become a popular research platform in the robotics community. Such robots offer various fields for new applications. However, they have several drawbacks compared to wheeled vehicles such as stability problems, limited payload capabilities, violation of the flat world assumption, and they typically provide only very rough odometry information, if at all. In this paper, we investigate the problem of learning accurate grid maps with humanoid robots. We present techniques to deal with some of the above-mentioned difficulties. We describe how an existing approach to the simultaneous localization and mapping (SLAM) problem can be adapted to robustly learn accurate maps with a humanoid equipped with a laser range finder. We present an experiment in which our mapping system builds a highly accurate map with a size of around 20 m by 20 m using data acquired with a humanoid in our office environment containing two loops. The resulting maps have a similar accuracy as maps built with a wheeled robot.**

I. Introduction

In the last few years, humanoid robots have become a popular research tool. They are assumed to offer new perspectives compared to wheeled vehicles since they are, for example, able to access different types of terrain and climb stairs. Generally, their human-like body plan helps when acting in a world designed for humans. The drawback of humanoids is that several tasks that can be easily carried out with wheeled robots are hard to achieve with legged systems. This includes, for example, stable motion with payload and the accurate execution of motion commands.

Maps of the environment are needed for a wide range of robotic applications including search and rescue, automated vacuum cleaning, home assistance, and several other service robotic tasks. Learning maps has therefore been a major research topic in the robotics community over the last decades. In the literature, the mobile robot mapping problem is often referred to as the simultaneous localization and mapping (SLAM) problem. It is considered to be a complex problem, because for localization a robot needs a consistent map and for acquiring a map a robot requires a good estimate of its location. This mutual dependency between the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space. Several techniques to the SLAM problem have been developed for wheeled robots but only a few of them have been shown to work on humanoid robots.

The central question in this context is what makes the data acquired with a humanoid different from data obtained with a wheeled platform. First, wheeled platforms are typically equipped with sensors that provide rather accurate odometry information. This yields good estimates of the relative movement or at least a reasonable starting point for local pose correction methods such as scan-matching. Compared to that, most humanoid robots (at least the ones that are affordable) do not have any odometry sensor. Furthermore, wheeled robots provide significantly more stable and smooth motion behaviors. This allows most robots to make the 2D plane assumption which means that the robot moves on a plane and the sensor is located parallel to that plane. Typically, this is not the case with a humanoid robot since they need to keep their balance at all time, even while standing. The attitude (roll and pitch angle) of the robot's sensors can easily change up to 20°. Due to the very limited payload of most humanoid robots and to keep the motion behavior stable, sensors have to be light-weight and small. A SICK LMS sensor for example, cannot be mounted on most humanoids. Therefore, one typically has to deal with rather noisy and short-range sensor data resulting from light-weight laser scanners such as Hokuyo URGs.

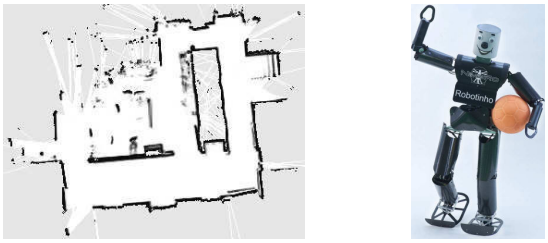


Fig. 1. A learned map by our approach using noisy and short-range laser data acquired with the humanoid Robotinho.

In this paper, we investigate humanoid-specific adaptations of a mapping approach that has been successfully used on wheeled vehicles equipped with a SICK laser range finders. We present a variant of our Rao-Blackwellized particle filter for learning grid maps [6] that can be used on a humanoid. This includes corrections for changing attitude (roll and pitch) of the sensor, dealing with missing odometry information, and scan-matching with a few distinct features only. In contrast to other mapping system that operate on humanoid robots, our approach is able to learn maps of comparably large indoor environments. We present an experiment in which our system built a map of an environment with a size of 20 m by 20 m containing two loops. The map has a quality comparable to the ones generated by a wheeled robot. The map shown in Fig. 1 illustrates the result of our mapping system given data acquired with our humanoid robot.

All authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

II. Related Work

The majority of approaches described in the SLAM literature addressed the problem of building maps with wheeled platforms. So far, only few researchers have addressed the problem of learning maps with a humanoid robot. Gutmann *et al.* [7] presented an approach to learn occupancy grid maps including elevation information with the Sony humanoid QRIO using stereo vision. In this context, they consider mapping mainly as a local problem to support collision avoidance or path planning tasks but they do not address issues such as loop-closing or place-revisiting.

A system that performs real-time localization and mapping with a humanoid robot was developed by Ozawa *et al.* [14]. Their approach is mainly based on 3D visual odometry and uses dense feature maps to estimate the position of the camera. A well-known drawback of this incremental approach is the drift created by the accumulation of errors.

There exist systems that concentrate on localization with a humanoid. Bennewitz *et al.* [1] presented an approach to visual localization of a humanoid that relies on a particle filter. Thompson *et al.* [16] performed localization with a humanoid equipped with a Hokuyo URG laser scanner. They use a known 2.5-dimensional map for a relatively small operational range of the robot. They do not suggest how to automatically learn such a map.

Solutions to the SLAM problem for wheeled vehicles often use EKFs. The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior over landmark maps and robot poses [15]. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. If these assumptions are violated, the filter is likely to diverge [9]. The unscented Kalman filter described in [9] is one way of better dealing with the non-linearities in the motion model of the vehicle. Moreover, the landmarks are assumed to be uniquely identifiable, even so, there exist techniques to deal with unknown data association in the SLAM context [13]. Lidoris *et al.* [10] presented an approach for motion planning in the context of EKF-based map learning with humanoids. They select gaze actions based on the expected entropy reduction in their model. They showed in simulation that such gaze actions can improve the pose estimate.

A full vision-based SLAM system that considers all 6 DoF and enables a humanoid robot to learn landmark maps has recently been presented by Davison *et al.* [2]. They extract features from a monocular camera and create a sparse map of high-quality stable features. The location of the features are tracked by applying an EKF.

Thrun *et al.* [17] describe a mapping approach that has been proven to work without odometry information. However, it requires an accurate laser range finder such as a SICK LMS sensor which cannot be carried by most humanoid robots. Furthermore, the sensor is assumed to have constant attitude angles. For robots equipped with a stereo camera, Elinas *et al.* [5] presented a SLAM system that does not need any odometry information.

In a work by Murphy [12], Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a possible robot trajectory and a map. The framework has been subsequently extended by Montemerlo et al. [11] for approaching the SLAM problem with landmark maps. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [4] and Hähnel *et al.* [8]. Whereas the first work describes an efficient map representation, the second presents an improved motion model that reduces the number of required particles.

In this paper, we apply a variant of our mapping approach [6] that applies a Rao-Blackwellized particle filter with an informed proposal distribution to efficiently sample the next generation of particles. We adapted our approach to explicitely address the problems that appear in the context of humanoid robots. This includes missing odometry information, comparably noisy data from light-weight proximity sensors, as well as a non-constant attitude (roll and pitch angle) resulting from the walking behavior.

III. Learning Maps with Rao-Blackwellized Particle Filters

Mapping with Rao-Blackwellized particle filters has been first introduced by Murphy [12]. The goal is to estimate the trajectory of the robot as well as a map of the environment up to time t. The key idea of a Rao-Blackwellized particle filter for map learning is to separate the estimate of the trajectory $x_{1:t}$ of the robot from the map m of the environment. This is done by the following factorization

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \cdot p(m \mid x_{1:t}, z_{1:t}), \quad (1)$$

where $z_{1:t}$ is the observation sequence and $u_{1:t-1}$ the odometry information. In practice, the first term of Eq. (1) is estimated using a particle filter and the second term turns into "mapping with known poses".

A particle filter requires three sequential steps to update its estimate. Firstly, one draws the next generation of samples from the so-called proposal distribution π. Secondly, one assigns a weight to each sample. The weights account for the fact that the proposal distribution is in general not equal to the target distribution. The third step is the resampling step in which the target distribution is obtained from the weighted proposal by drawing particles according to their weight.

One of the main challenges in particle filtering is to choose an appropriate proposal distribution. The closer the proposal is to the true target distribution, the more precise is the estimate represented by the sample set. Typically, one requires the proposal π to fulfill the assumption

$$\pi(x_{1:t} \mid z_{1:t}, u_{1:t-1}) = \pi(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t-1}) \cdot \pi(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-2}). \quad (2)$$

According to Doucet [3], the distribution

$$p(x_t^{(i)} \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \quad (3)$$

is the optimal proposal for particle i with respect to the *variance of the particle weights* that satisfies Eq. (2). This proposal minimizes the degeneracy of the algorithm (Proposition 4 in [3]). In Eq. (2), $z_{1:t-1}$ and $x_{1:t-1}$ represent the map m_{t-1}. Based on the importance sampling principle, the weights have to be computed as follows [6]

$$\begin{aligned} w_t^{(i)} &= \frac{\text{target distribution}}{\text{proposal distribution}} && (4)\\ &= w_{t-1}^{(i)} \frac{\eta p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t^{(i)} \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} && (5)\\ &\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} && (6)\\ &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}). && (7) \end{aligned}$$

Unfortunately, the optimal proposal distribution is in general not available in closed form or in a suitable form for efficient sampling. As a result, most efficient mapping techniques use a Gaussian approximation of the optimal proposal. This approximation can easily be computed and allows the robot to sample efficiently.

For each particle i, the parameters $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ of the Gaussian are determined individually by sampling J test points $\{x_j\}_{j=1}^J$. The test points have to be sampled close to the expected location of the robot. A good guess for the expected location of the robot can be determined by scan-matching, a method to find the pose with a locally optimal match of the current scan with the map constructed so far. *Note that a robust scan-matching method is an important prerequisite for successfully applying the RBPF mapping technique*. Otherwise, the Gaussian approximation that is used as the proposal distribution is not valid. The Gaussian is computed for each sample based on the set $\{x_j\}_{j=1}^J$

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{J} x_j \cdot p(z_t \mid m_{t-1}^{(i)}, x_j) p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \quad (8)$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{J} p(z_t \mid m_{t-1}^{(i)}, x_j) p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T, \quad (9)$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^{J} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}). \quad (10)$$

With Eq. (8)-(10), we obtain a closed form approximation of the optimal proposal which enables us to efficiently obtain the next generation of particles. Using this proposal distribution, the weights have to be computed as

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) && (11)\\ &= w_{t-1}^{(i)} \cdot \int p(z_t \mid m_{t-1}^{(i)}, x') \cdot p(x' \mid x_{t-1}^{(i)}, u_{t-1})\, dx' \\ &\simeq w_{t-1}^{(i)} \cdot \eta^{(i)}. && (12) \end{aligned}$$

As we showed in previous work [6], such an approach is well suited to robustly learn accurate maps of the environment with wheeled robot such as ActivMedia Pioneer robots equipped with SICK laser range finders.

IV. Adaptations for Mapping with a Humanoid

In this section, we present the modifications to our mapping system so that it can be applied to successfully learn accurate maps with data acquired by a humanoid.

A. Attitude-based Scan Correction

When applying the approach described in the previous section to laser data recorded by a humanoid, the quality of the map will typically be poor. One reason for this is that the humanoid cannot move and at the same time keep the sensor with zero roll and pitch angles. As a result, the object that caused a reflection of the laser beam is observed at different heights in consecutive scans. This makes it nearly impossible to match scans as it is a prerequisite for our approach. Hence, we have to account for the changing attitude. The roll and pitch angle of the laser range finder, however, can be quite accurately estimated using an attitude sensor or a low-cost IMU. As result, we are able to compute the three-dimensional position of the object that caused a reflection. Let ρ_k be the measured range of laser beam k and α_k the corresponding angle. The 3D position of the object is given by

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \rho_k R(\phi, \theta, \psi) \begin{pmatrix} \cos\alpha_k \\ \sin\alpha_k \\ 0 \end{pmatrix} + \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}, \quad (13)$$

where R is the 3×3 rotation matrix, ϕ, θ, and ψ refer to the roll, pitch, and yaw angle, and r_x, r_y, r_z to the position of the sensor in the world.

By assuming that the observed objects are walls, cupboards, or similar objects that have the same shape for all z, we can compute a corrected range ρ'_k as

$$\rho'_k = \rho_k \left\| \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R(\phi, \theta, 0) \begin{pmatrix} \cos\alpha_k \\ \sin\alpha_k \\ 0 \end{pmatrix} \right\|. \quad (14)$$

By replacing ρ_k by ρ'_k for each beam k in a laser range scan, we obtain a corrected range observation that stays constant under changes to the attitude of the sensor. Note that if the roll and pitch angle are too huge, scans might hit the floor. In this case, the measurements are neglected since they do not contain information about the walls.

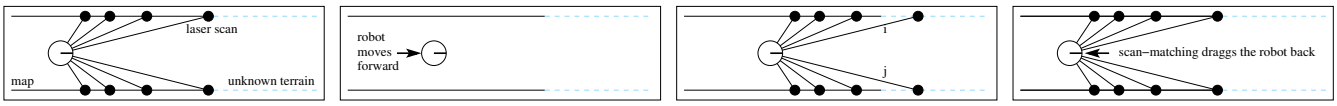


Fig. 2. The problem of scan-matching in a feature-less corridor: (1) The scan is integrated, (2) the robot moves forward, (3) the new scan is obtained, (4) the scan is matched against the map constructed so far and thus the robot is dragged back to the previous location.

B. Scan-Matching with Poor Features

To compute the Gaussian proposal in our mapping system, we perform scan-matching to find the most likely pose by matching the current observation against the map constructed so far

$$x_t^* = \operatorname*{argmax}_x p(x \mid m_{t-1}, z_t, x'_t), \tag{15}$$

where x'_t is the initial guess which is typically computed from odometry. In practice, one applies Bayes' rule and seeks for the pose with the highest observation likelihood $p(z_t \mid m_{t-1}, x)$. Often, a search technique similar to gradient descent is applied. To compute the likelihood of an observation, we use the so called "beam endpoint model". In this model, the likelihood of an individual beam is computed based on the distance between the endpoint of the beam and the closest obstacle from that point.

Using the Hokuyo URG scanner, we have a significantly reduced measurement range compared to traditional range scanners such as a SICK LMS. As a result, the robot can observe only a small area of the environment. Especially if a mobile robot moves along a symmetric structure with poor features, such as a long corridor, scan-matching becomes difficult since ambiguities cannot be resolved. Fig. 2 illustrates a typical situation in which scan-matching in a feature-less corridor fails.

In such a situation, observations are identical or at least very similar for all poses independent of the horizontal position. As a result, the scan-matching procedure reverts the movements of the robot in the horizontal direction. The reason for this is that the obtained scans match perfectly the map at the initial position. Therefore, corridors are often shorter in the maps than in reality. In case an accurate odometry information is available, such situations might be resolved. In general, the shorter the range of the laser, the higher the risk of ambiguities and the lower the quality of the odometry, the worse the initial guess for the matching algorithm. Since this problem occurs comparably often when building maps with a humanoid robot equipped with a Hokuyo URG scanner, we present a way to better deal with such ambiguities when matching scans.

To overcome this problem, we propose to use only a subset of the scan for finding the correspondence but to use the full scan to update the map. Even if this might sound counter-intuitive at first sight, it substantially improves the result of scan-matching in areas with poor features. By neglecting a small fraction of the scan, namely those beams that are close to the maximum usable range of the scanner, the problem of virtually dragging the robot backwards during the pose correction step can be reduced. If, however, in Fig. 2 the beams labeled as i and j are not used for matching but for updating the map, the scan-matcher will typically confirm the predicted position and will mainly align the robot in the vertical direction but not in the horizontal one. Note that after this correction is carried out, the full scan is used to update the map.

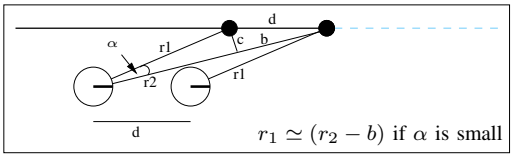


Fig. 3. Geometry used to determine the maximum length of a beam that should be used for scan-matching.

Based on this example, we can investigate which beams should be neglected in the matching phase. Consider the situation depicted in Fig. 3 in which the robot moves a distance d forward. After moving, the robot should only consider those beams for matching that are likely to hit an obstacle that was visible from the previous location of the robot. To consider only these obstacles, we have to neglect the beams that are longer than the maximum usable range of the scanner minus the distance b. Let α be the angular resolution between two beams. Then, we can compute the length b as

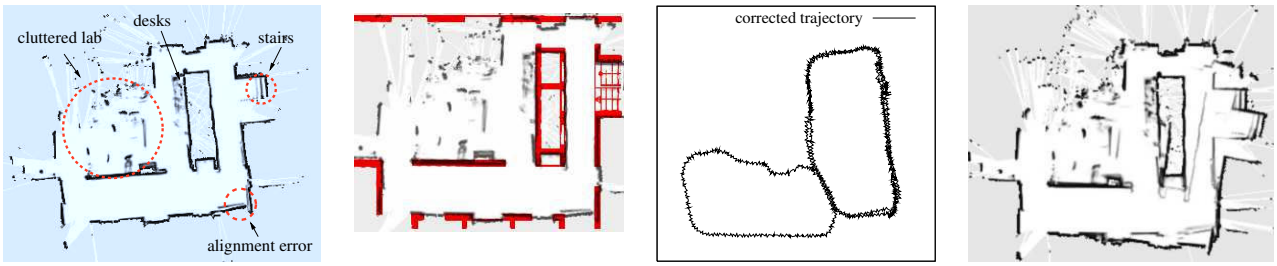
$$b \quad = \sqrt{d^2 - c^2} \quad = \sqrt{d^2 - \sin^2 \alpha \cdot r_1^2} \leq d. \tag{16}$$

As can be seen, b is bounded by the distance d moved by the robot. As a result, we can improve the scan-matching in feature-less corridors by using only those scans which are shorter than the maximum usable range of the scanner minus the estimated moved distance d of the robot when optimizing Eq. (15).

Note that an alternative strategy that neglects beams that end in an unknown cell in the previous map is not sufficient since this can lead to wrong corrections if appropriate features are visible and the motion of the robot is slightly overestimated.

C. Dealing with Missing Odometry

Today's expensive humanoid robots provide odometry information that allows a robot to incrementally build local maps (compare Gutmann *et al.* [7]). Low cost or self constructed humanoids, however, often do not provide such a reliable estimate about the motion of the robot. Dead reckoning, i.e., the prediction of the robot's pose based on executed motion commands, could be applied, however, it provides only very noisy estimates due to slippage on the ground. Our robot, for example, does not provide any usable information about its relative movement.



(a) (b) (c) (d)

Fig. 4. Image (a) depicts the grid map learned with data acquired by a humanoid. By considering an overlay with the floor-plan (walls in the floor plan are colored red), one can see that the environment is rather accurately mapped (b). Image (c) shows the corrected trajectory of the laser scanner projected on the x/y plane. Image (d) depicts a map learned without the attitude correction. The map is inconsistent and comparably blurred.

Reducing only the range during scan-matching as described in the previous paragraph is not sufficient when no odometry information at all is available. We furthermore need an initial estimate for the motion of the robot to obtain a good pose and thus map estimate. Assuming that the robot starts in a place at which sufficient distinct structure is available for scan-matching, it can quite accurately estimate its motion. Assuming an approximately constant speed of the vehicle, we set

$$u_{t-1} \quad = \quad x^*_{t-1} \ominus x^*_{t-2}, \tag{17}$$

where u_{t-1} is the motion estimate that guides the robot from x_{t-1} to x_t, $\ominus$ is the inverse of the motion composition operator, and x^* results from Eq. (15). This estimate leads to a reasonable odometry guess in case the robot moves with constant speed through passages that do not provide distinct features in the sensor data. In all other passages, the scan-matching technique will anyway find an acceptable pose estimate. In practice, Eq. (15) and Eq. (17) are always computed in an alternating way. One starts with scan-matching without odometry, then estimates the odometry, which is in the subsequent step used to initialize the scan-matcher. Even though, the techniques presented in this paper do not describe a new mapping framework, we found that they are relevant to solve the mapping problems with humanoid robots. Furthermore, we believe that they can be easily integrated into other mapping frameworks which apply scan-matching as an intermediate procedure.

V. Experiments

The humanoid robot Robotinho used for the experiments is depicted in Fig. 1. It is self-constructed, around 1 m tall with a total weight of about 5.2 kg, and has 23 degrees of freedom. For our experiments, we equipped it with a Hokuyo URG laser range finder and a XSens IMU (here, only the attitude information is used). The Hokuyo URG is a light-weight scanner with a maximum range of 4.2 m. A measurement range of 4 m, however, can only be obtained with bright and highly reflective obstacles. Dark doors, badly reflecting furniture, or even grayish concrete walls lead to a significantly reduced measurement range of the scanner if the obstacle is not hit perpendicularly. The XSens is used to estimate the attitude (roll and pitch angle) of the chest of the robot. The robot does not possess any odometry sensors.

Fig. 5. Grid map learned from data obtained by a human carrying a laser range finder in the hand. The arrows indicate parts of the environment in which the sensor perceived mainly invalid observations. The estimated trajectory is shown in red.

A. Learning Accurate Grid Maps

The first experiment is designed to show that our approach is well suited to learn accurate grid maps with a humanoid robot. We steered our robot with a joystick through our lab environment. It consists of two corridors which result in two loops the robot traversed, one of them three times. As can be seen in Fig. 4 (a) and (b), our system maps the environment rather accurately. Only one small alignment error occurred at a part of the wall/door. One interesting observation is that the stairs can be identified quite well in the map. The parallel lines are not alignment errors but result from reflection of the individual steps while the robot was walking and thus changing the attitude of the sensor. Fig. 4 (c) shows the estimated trajectory of the robot during that experiment.

Additionally, we disabled the attitude-based scan correction to illustrate its effect. The right image in Fig. 4 depicts the result. The map is more blurred since the laser beams often hit the wrong grid cells. Furthermore, the filter made one wrong pose correction which leads to an inconsistent estimate.

We furthermore performed a second experiment in which a person was holding the laser range scanner and was walking through the environment. As can be seen in Fig. 5, also here we obtain a comparably good map. By looking closely, one can see that the corridors are not perfectly matched and one corridor is slightly too short. This is due to the fact that on one side, the corridor consists of glass panes only and the Hokuyo scanner does not provide any valid data in case a beam hits glass. Therefore, several observations contain no useful information which results in an imperfectly aligned

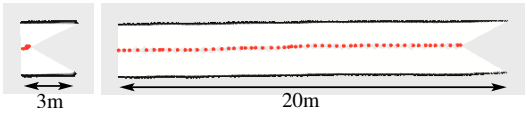


Fig. 6. Scan-matching in a feature-less corridor. Left: Using standard scan-matching, the estimated pose of the robot is always the same. Right: By using our approach, the robot performs much better even if the corridor is still too short (20 m vs. 22.3 m). The estimated trajectory is shown in red.

beam length reduction	forward movement d 0.3m	0.4m	0.5m	0.75m	1.0m
0.3m	19.9m	16.3m	14.1m	10.1m	9.5m
0.4m	19.7m	20.0m	17.2m	11.8m	9.6m

Fig. 7. Estimated corridor length for different movements (truth=22.3 m).

map. Nevertheless, the resulting map is sufficient for most tasks such as robot localization or motion planning.

B. Scan-Matching with Poor Features

A further set of experiments investigates the advantages of our scan-matching variant compared to the same approach lacking our technique. By neglecting long beams during the matching phase but integrating them into the map, a substantially better pose estimate can be obtained. Fig. 6 shows the result of the standard scan-matching approach (left image) and our variant (right image). As can be seen, our approach does not provide a perfect motion estimate since the corridor is still shorter than in reality (20 m instead if 22.3 m). As illustrated in Fig. 7, the estimated forward movement (d in Eq. (16)) is a good parameter to quantify the reduction of the maximum valid beam length. If the beam length reduction is chosen too small, the accuracy of the scan matcher drops substantially.

C. Limitations

Given the short range of the sensor, our robot is currently only able to map environments without large free spaces such as hallways. In large rooms such as entrance halls, the proposal cannot be computed accurately - especially without real odometry. In case the robot moves through areas with poor structure, we assumed a constant speed of the vehicle. Furthermore, our system might be less accurate in situations in which the attitude is affected by significant changes while at the same time most of the objects observed by the scanner look different in different heights such as desks for example. In our current configuration, however, the system appears to be comparably robust in practical scenarios.

VI. Conclusion

In this paper, we addressed the problem of learning accurate grid maps using laser data acquired by a humanoid. We present techniques to deal with the specific difficulties of typical humanoids such as changing roll and pitch angle of the sensor, missing odometry information, and comparably noisy and short-range sensor data. As a result, we are able to apply a Rao-Blackwellized particle filter to estimate the joint posterior about the trajectory of the robot and the map of the environment. In combination with the adaptations for the mentioned difficulties, this solution to the simultaneous localization and mapping problem allows a humanoid robot to robustly learn maps. As our experimental results show, the resulting grid maps have a high accuracy, similar to maps built with a wheeled robot. To the best of our knowledge, our system is the first one which is able to build such accurate grid maps containing several loops with a humanoid robot.

Acknowledgment

This work has partially been supported by the German Research Foundation (DFG) under the contract numbers SFB/TR-8 and BE 2556/2-2 as well as by the EC under contract number FP6-IST-34120-muFly.

References

[1] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 143–157, 2006.

[2] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[3] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge, 1998.

[4] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[5] P. Elinas, R. Sim, and J. J. Little. σSLAM: Stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[7] J.-S. Gutmann, M. Fukuchi, and M. Fujita. A floor and obstacle height map for 3D navigation of a humanoid robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Barcelona, Spain, 2005.

[8] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[9] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[10] G. Lidoris, K. Kühnlenz, D. Wollherr, and M. Buss. Information-based gaze direction planning algorithm for SLAM. In *Proc. of IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids)*, 2006.

[11] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1985–1991, Taipei, Taiwan, 2003.

[12] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[13] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[14] R. Ozawa, Y. Takaoka, Y. Kida, K. Nishiwaki, J. Chestnutt, J. Kuffner, S. Kagami, H. Mizoguchi, and H. Inoue. Using visual odometry to create 3d maps for online footstep planning. In *Proc. of IEEE Intl. Conf. on Systems, Man, and Cybernetics*, 2005.

[15] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[16] S. Thompson, S. Kagami, and K. Nishiwaki. Localisation for autonomous humanoid navigation. In *Proc. of IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids)*, 2006.

[17] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, San Francisco, CA, 2000.

[C15] B. Frank, M. Becker, C. Stachniss, M. Teschner, and W. Burgard. Efficient path planning for mobile robots in environments with deformable objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

Efficient Path Planning for Mobile Robots in Environments with Deformable Objects

Barbara Frank Markus Becker Cyrill Stachniss Wolfram Burgard Matthias Teschner

***Abstract*— The ability to reliably navigate through the environment is an important prerequisite for truly autonomous robots. In this paper, we consider the problem of path planning in environments with non-rigid obstacles such as curtains or plants. We present an approach that combines probabilistic roadmaps with a physical simulation of object deformations to determine a path that optimizes the trade-off between the deformation cost and the distance to be traveled. We describe how our approach utilizes Finite Element theory for calculating the deformation cost. Since the high computational requirements of the corresponding simulation prevent this method from being applicable online, we present an approximation that uses a preprocessing step to determine a deformation cost function for each object. This cost function allows us to estimate the deformation costs of arbitrary paths through the objects and is used to evaluate the trajectories generated by the roadmap planner online. We present experiments which demonstrate that the resulting algorithm plans nearly identical trajectories compared to the method that relies on computationally intense simulations. At the same time, our approach allows the robot to quickly calculate paths in environments with deformable objects.**

I. INTRODUCTION

Path planning is one of the fundamental problems in robotics, and the ability to plan collision-free paths is a precondition for numerous applications of autonomous robots. The path planning problem has traditionally received considerable attention in the past and has been well-studied. The majority of approaches, however, has focused on the problem of planning paths in static environments and with rigid obstacles [16, 7, 17]. In the real world, not all obstacles are rigid and considering this knowledge can enable a robot to accomplish navigation tasks that otherwise cannot be carried out. For example, in our everyday life we deal with many deformable objects such as plants, curtains, or cloth and we typically utilize the information about the deformability of objects when planning a movement. As a motivating example, consider the situation depicted in Fig. 1, in which the robot needs to pass through a curtain to move from its current position to the goal location since no other path exists in the environment. In this particular situation, traditional approaches that do not take the deformability of objects into account, will fail since no collision-free path exists. In contrast to this, the approach presented in this paper is able to determine the deformation cost introduced by passing the curtain and to utilize this information during path planning. The key idea of our approach is to use a heuristic function to estimate the deformation cost, which allows the robot to perform the necessary calculations online.

All authors are with the Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany.
{bfrank,mbecker,stachnis,burgard,teschner}@informatik.uni-freiburg.de

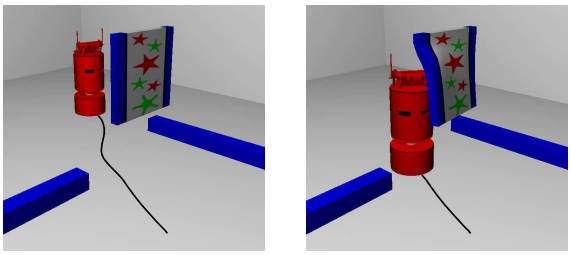


Fig. 1. Path planning in an environment with a deformable object. The robot (shown in red) deforms the curtain on its path to the goal.

One potential method of taking deformations of objects into account is by generating trajectories using a method such as probabilistic roadmaps (PRM) [15] and considering deformable objects as free space. When answering path queries, the planner has to simulate the deformation of the non-rigid objects resulting from the interaction with the robot. However, performing an appropriate simulation typically requires significant computational efforts which makes such an approach unsuitable for online trajectory planning. Therefore, we propose an approach to learn an approximative deformation cost function in a preprocessing step. The advantage of our method is that this function can be evaluated efficiently during planning. In this way, our approach reduces the time to solve a path query from several minutes to a few hundred milliseconds. The assumption made throughout this paper is that the robot can deform but cannot move objects in the environment. Additionally, we neglect the interactions between different deformable objects.

The contribution of this paper is an approach to mobile robot path planning that explicitly considers deformable objects in the environment. It employs the probabilistic roadmap method and learns a deformation cost function using a physical simulation engine that is based on Finite Element theory. Our approach trades off the travel cost with the deformation cost when answering path queries and can be executed online.

This paper is organized as follows: After discussing related work, we present in Section III our technique to compute the deformations of objects by means of physical simulation. We then describe how to plan a path in the presence of deformable objects. Additionally, we describe our approximation of the deformation cost function in Section IV. Finally, we present experiments that illustrate the advantages of our approach compared to previous methods.

II. RELATED WORK

The majority of approaches to mobile robot path planning assumes that the environment is static and that all objects are rigid [16, 15, 3]. Recently, several path planning techniques for deformable robots in static environments have been presented [4, 10, 14]. In the context of deformable objects, the underlying model has a substantial influence on the accuracy of the estimated deformations as well as on the performance of the planner. One typically distinguishes between geometrically and physically motivated approaches. Geometric approaches such as the free-form deformation (FFD) can be computed efficiently. For example, the FFD method of Sederberg and Parry [21] is based on trivariate Bernstein polynomials and allows for deformation by manipulating the control points.

To represent non-rigid objects and to calculate deformations, mass-spring systems have been frequently used. They are easy to implement and can be simulated efficiently. Whereas such models are able to handle large deformations, their major drawback is the tedious modeling as there is no intuitive relation between spring constants and physical material properties in general [19]. Finite Element Methods (FEMs) reflect physical properties of the objects in a better way. This allows for a more intuitive modeling since they require only a small number of parameters. The disadvantage of FEMs is the computational resources required to calculate deformations. In our current system we therefore use the computationally efficient co-rotational Finite Element approach [11, 18] which avoids nonlinear computations.

In the context of path planning, Kavraki *et al.* [14] developed the f-PRM-Framework that is able to plan paths for flexible robots of simple geometric shapes such as surface patches [13] or simple volumetric elements [2]. They apply a mass-spring model to compute deformations. The planner selects the deformation of the robot that minimizes its deformation energy. Similar to this technique, Gayle *et al.* [10] presented an approach to path planning for a deformable robot that is based on PRMs. To achieve a more realistic simulation of deformations they add constraints for volume preservation to the mass-spring model of the robot. Bayazit *et al.* [4] also studied planning for a deformable robot. Their algorithm proceeds in two steps. First, an approximate path is found in a probabilistic roadmap. In the next step, this path is refined by applying a free-form deformation to the robot and hence avoiding collisions with obstacles. This deformation method can be computed efficiently but is less accurate than physically motivated models. In contrast to our approach, these planners deform the robot rather than the obstacles to avoid collisions.

An approach to planning in completely deformable environments has been proposed by Rodríguez *et al.* [20]. They employ a mass-spring system with additional physical constraints for volume-preservation [23] to enforce a more realistic behavior of deformable objects. They use rapidly exploring random trees and apply virtual forces to expand the leaves of the tree until the goal state is reached. The obstacles in the environment are deformed through external forces resulting from collisions with the robot.

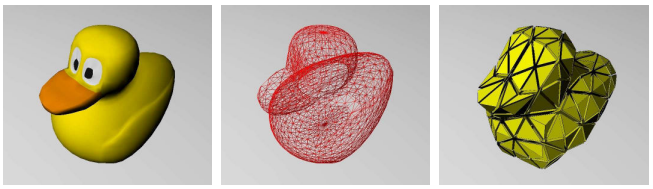


Fig. 2. Different levels of representation for a deformable object (left) in the simulation environment: fine surface mesh (middle) and coarse tetrahedral mesh (right).

All techniques mentioned above, however, require substantial computational resources and cannot be executed online in general. In contrast to this, our approach can efficiently answer path queries by estimating potential deformations of objects in a preprocessing step. This is achieved by approximating a deformation cost function which is then considered during the planning process. Furthermore, our deformable model is based on FEM, which allows for more accurate deformations.

III. SIMULATION OF DEFORMABLE OBJECTS

To consider non-rigid obstacles in the environment during planning, we need a model that allows us to compute the deformations given an external force. In this section, we describe how we achieve a physically realistic simulation of object deformations. We will first introduce the co-rotational FEM. Then, we describe how to detect collisions between deformable objects and the robot and how to compute contact forces resulting from collisions. Finally, we introduce the cost resulting from a deformation.

Our simulation system proceeds as follows: in each time step, it computes deformations and unconstrained motions of objects, then it detects collisions, computes contact forces for colliding points, and finally corrects the unconstrained motion with appropriate repulsion forces.

A. Deformable Modeling

The obstacles in the workspace are 3D objects. The surface of objects is represented by a fine resolution triangle mesh. A tetrahedral mesh is used to represent the interior of these volumetric objects (see Fig. 2). The actual deformations are computed based on the coarse resolution tetrahedral mesh.

To compute the deformation of our tetrahedral objects we use the co-rotational Finite Element formulation [11, 18]. The total potential energy of a single tetrahedral element e is given by

$$\Pi = U_e + WP, \tag{1}$$

where WP is the work potential. WP is determined by the external forces and inner energy U_e

$$U_e = \frac{1}{2} \int_e \sigma^T \epsilon \, dV. \tag{2}$$

Since we assume only linear isotropic materials, we have a linear relation between the stress σ and the strain ϵ given by the generalized Hooke's law.

The key idea of the Finite Element method is to discretize the object into a finite set of elements (in our case tetrahedrons) to compute the deformations based on Eq. (1) on the nodes and to interpolate the deformation in the elements using the nodal values. To compute the strain ϵ from the nodal deformations in our model, we use the linear Cauchy strain tensor which is efficient to compute. The Cauchy tensor, however, is not rotationally invariant. This leads to ghost forces which result in distortions for large rotational deformations. Therefore, we keep track of the rigid body motion for each element by extracting the rotation from the transformation matrix using polar decomposition. Applying the strain tensor in the rotated frame, leads to rotational invariance and has low computational costs compared to the nonlinear strain tensor. We will discuss the performance of our Finite Element approach compared to the versatile mass-spring approach used by Rodríguez *et al.* [20] in the experimental results section.

B. Collision Detection

For the realistic processing of interactions between the robot and the deformable objects, an efficient collision detection algorithm is required. We employ a spatial subdivision scheme in our simulation system, where the elements are stored in a hash grid [24]. The key idea of this approach is to implicitly discretize $\mathbb{R}^3$ by storing the elements and nodes in the hash grid. Since the space is usually filled sparsely and non-uniformly, this method consumes less memory than an explicit discretization. The hash key is computed from the coordinates of the corresponding grid cell. As a result, only the elements with the same hash key need to be checked for collisions.

To check for collisions, one computes the intersection between points and tetrahedra. This can be done efficiently using barycentric coordinates of the points with respect to the tetrahedra.

Methods commonly employed for rigid bodies such as bounding box hierarchies [8] are less suited for deformable objects, since these spatial data structures cannot be precomputed [25].

C. Computation of Contact Forces

To handle collisions between the robot and the deformable objects, we employ the force-based collision handling scheme proposed by Spillmann *et al.* [22]. It combines the advantages of penalty and constraint-based collision handling schemes. For a set of colliding points of a tetrahedral mesh, we compute a collision free state using a linearized relation between internal forces and displacements of all affected points. Contact forces can be computed analytically to obtain this collision-free state while conserving overall system energy.

Using this combination of FEM-based simulation and the collision handling described above, our system can simulate thousands of tetrahedra at interactive rates. An example implementation of the simulation system is available online [12].

D. Object Deformation Costs

The inner energy of an object, specified in Eq. (2), provides a measure of the deformation costs of a tetrahedral object and thereby of the additional energy consumption of the robot. In case of an undeformed object, the inner energy is zero. Otherwise, the inner energy increases depending on the deformation of the object. For an object O with tetrahedral elements $\{e_i\}$, we define the total inner energy U_O induced by a robot r at position $\mathbf{p}$ approaching from direction θ by the sum over the inner energies of all elements e_i of the object $U_O(r, \mathbf{p}, \theta) := \sum_{e_i \in O} U_{e_i}(r, \mathbf{p}, \theta)$.

For any given position $\mathbf{p}$ and direction θ we determine the total deformation cost $C_{def}(\mathbf{p}, \theta) := \sum_O U_O(r, \mathbf{p}, \theta)$ by summing over all objects O in our workspace. The direction θ has to be taken into account, since deformable objects might deform differently when approaching them from different directions.

The total deformation cost on a path Γ of the robot approaching from direction θ in our environment naturally results in the sum of the deformation costs of all objects that are deformed by the robot while moving on the path in discrete time steps t_i:

$$C_{def}(\Gamma, \theta) \quad = \quad \sum_{t=t_1}^{t_n} C_{def}(\mathbf{p}_r(t_i), \theta(t_i)). \tag{3}$$

Here $\mathbf{p}_r(t_i)$ is the position of the robot on Γ at time t_i. $\theta(t_1)$ is given by θ, all other directions $\theta(t_i)$ are determined as the difference between $\mathbf{p}_r(t_i)$ and $\mathbf{p}_r(t_{i-1})$.

IV. PATH PLANNING WITH DEFORMABLE OBJECTS

In this section, we introduce an approximative deformation cost function that allows a robot to plan a path in such environments online.

A. Overview of the Path Planning System

The general path planning problem is to find a sequence of valid robot configurations that lead from the starting to the goal configuration. Probabilistic roadmap planners achieve this by constructing a roadmap that represents the environment of the robot and by applying a graph search algorithm to find a path from the starting to the goal configuration. The roadmap is constructed in a preprocessing step by sampling points in the configuration space of the robot. These points have to satisfy certain feasibility constraints. In general, valid configurations are required to be part of the free configuration space $\mathcal{C}_{free}$. In our situation, however, we modify this constraint and require configurations to be in $\mathcal{C}_{free} \cup \mathcal{C}_{def}$. Thus, we also accept configurations that lead to collisions with deformable objects. In our current implementation, we use Hammersley-sampling [6] to generate configuration hypotheses in the space. This deterministic sampling method generates a sequence of points that are distributed with low discrepancy. After a designated number of samples has been generated, a local planner connects neighboring samples

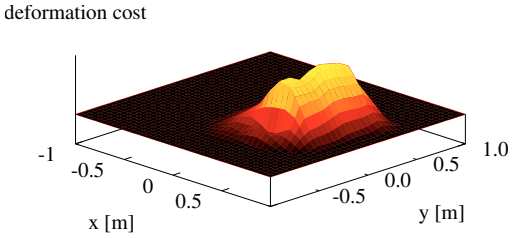


Fig. 3. Deformation costs for moving a robot along straight lines through a curtain. The lines are specified by starting points (x, y) and travel direction $\theta = 0°$ relative to the center of mass of the obstacle.

for which a valid path exists. This typically results in a roadmap that covers the environment of the robot and can be utilized for planning paths on which objects are allowed to be deformed by the robot. To answer a path query, we then insert the starting and the goal configuration into the roadmap and connect them to their neighbors. Finally, we apply $A^\star$ to find the best path from the starting to the goal location on the graph. Here, we search for the path with the best trade-off between travel costs and deformation cost. Therefore, we need to be able to estimate the expected deformation cost arising on the edges of the roadmap.

The simulation system presented in the previous section can be used inside the planning algorithm to compute the deformation cost $C_{def}(i)$ of an edge i by simulating a robot moving along this edge deforming the object. The edges considered during $A^\star$ planning are evaluated by trading off the deformation and travel cost. In our planning system, we assume the travel cost to be proportional to the length of the edge i. This results in the cost function

$$C(i) \quad := \quad \alpha\, C_{def}(i) + (1-\alpha)\, length(i), \qquad (4)$$

where $\alpha \in [0, 1]$ is a user-defined weighting coefficient. In order to obtain an admissible heuristic for $A^\star$, we use the Euclidean distance to the goal location weighted with $(1-\alpha)$. Thus, we are able to find the path in the roadmap that optimizes the trade-off between travel cost and deformation cost for a given user-defined parameter α.

This technique leads to a working planning system that considers deformations of the objects in the environment when planning a trajectory for a mobile robot. The drawback of this technique, however, is its high computational requirements. Answering a path query typically takes several minutes even for small examples. Therefore, we developed an alternative approach that computes an approximation of the cost function in advance and thus facilitates online path planning.

B. Approximative Deformation Cost Function

The goal of the approximative cost function is to quickly provide an estimate of the deformation costs for all objects along an edge in the roadmap. Such a function can be used in the planning approach described above to determine C_{def}. The actual value of the deformation cost function of an object mainly depends on the trajectory of the robot relative to the object. Therefore, we precompute the deformation cost for a number of line segments through each object.

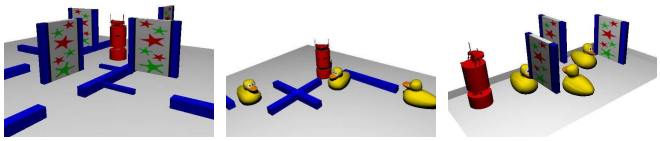


Fig. 4. Test environments: world 1 with curtains (left), world 2 with rubber ducks (middle), and world 3 with rubber ducks and curtains (right).

A line is specified by a starting location (x, y) and the traveling direction θ as well as the traveled distance l on the line segment. The traveled distance is constrained to the maximum distance that the robot can move while still deforming the object.

In a preprocessing step, we carry out the simulations for a uniform resolution of starting points and directions and store the deformation costs for a fine length resolution in a histogram. This leads to the approximate deformation cost function $\hat{C}_{def}(x, y, \theta, l) \rightarrow \mathbb{R}$ which returns the deformation cost for edges of the roadmap.

We compute the deformation cost $\hat{C}_{def}(x, y, \theta, l)$ of an arbitrary edge e in the roadmap by first determining the starting position (x, y), direction θ, and length l relative to the deformable object. We then apply a kernel smoother [1] considering all neighboring line segments e^t in the histogram

$$\hat{C}_{def}(e) \quad = \quad \frac{\sum_t K\left(\frac{e-e^t}{h}\right) \hat{C}_{def}(e^t)}{\sum_t K\left(\frac{e-e^t}{h}\right)}, \qquad (5)$$

where $K(u)$ is the multivariate Gaussian kernel with identity as covariance.

As distance metric between different line segments, we employ the Euclidean distance and normalize the orientation with respect to the positions.

To finally answer path queries, we apply the $A^\star$ algorithm on the roadmap. The cost of each edge in the graph is computed according to Eq. (4) using the precomputed approximation $\hat{C}_{def}$ of C_{def}. Computing the deformation costs in a preprocessing step substantially increases the performance of our planner as can be seen in the experiments.

Although the precomputation is computationally intense, it has to be done only once for each distinct object. Such a cost function can even be transfered between environments. The following section provides results on the runtime of the precomputation for different resolutions of the deformation cost grid. Fig. 3 illustrates the deformation cost $\hat{C}_{def}$ of the curtain shown in Fig. 1 along a series of straight lines.

V. EXPERIMENTS

We carried out different experiments to evaluate our path planning approach. In this section, we first compare the deformation cost obtained by the FEM-based simulation technique with our approximative solution that computes a deformation cost function for each object in a preprocessing step. Next, we investigate how the deformation cost weighting coefficient α influences the path search. Finally, we present examples of planned trajectories in environments with deformable objects.

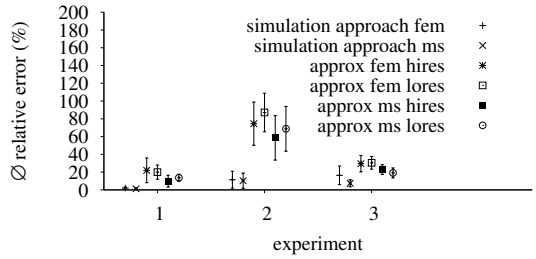


Fig. 5. Comparison of the error in the deformation cost for the simulation-based approach and our approximation.

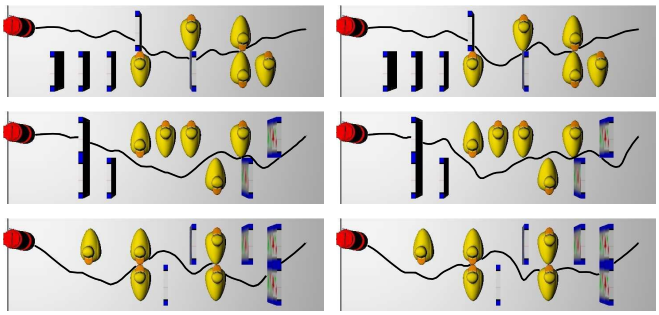


Fig. 6. Example trajectories generated by the simulation approach (left) vs. our approximation approach (right).

A. Cost Function and Runtime

In the first experiment, we compare the simulation approach to estimate the deformation cost with our approximative cost function. We chose curtains and rubberducks as deformable objects. The curtains are modeled to be easily deformable while the rubberducks have high deformation costs. Both approaches had to solve 25 path queries in the three test environments depicted in Fig. 4. After planning, the best trajectory is sent to a path execution module that guides the robot along that trajectory. In our simulation, the execution of motion commands is affected by noise.

The experiments are carried out for different resolutions of the approximate cost function. Furthermore, we compare our approach to a simulation system using the versatile mass-spring model. We evaluated the error between predicted and measured deformation cost. The results are shown in Fig. 5. As expected, the error of the simulation technique is typically smaller compared to our approximative approach. This, however, comes at the expense of running time as illustrated in Table I. While our approach answers path queries even for complex environments in a few hundred milliseconds on average, the simulation approach spends generally about half an hour on a single query. Thus, our approach is about four orders of magnitude faster than the simulation using FEM. We also compared our approach to a simulation system using the versatile mass-spring model. Although this simulation system can be evaluated faster, our approach still is about 2000 times faster. The runtime for the precomputation of the approximate cost function for different resolutions of the cost grid is summarized in Table II.

Additionally, we carried out an experiment in a randomized world, where we compare the computed paths for the simulation and the approximation approach. The

TABLE I

AVERAGE RUNTIME INCLUDING CONFIDENCE INTERVALS.

World	∅ Query (FEM)	∅ Query (mass-spring)	∅ Query (our approach)
1	36 m 41 s ± 347 s	12 m 45 s ± 112 s	0.4 s ± 0.04 s
2	30 m 33 s ± 512 s	8 m 10 s ± 73 s	0.2 s ± 0.02 s
3	29 m 43 s ± 130 s	7 m 27 s ± 36 s	0.3 s ± 0.04 s

TABLE II

RUNTIME FOR THE PRECOMPUTATION OF THE DEFORMATION COST FUNCTION FOR DIFFERENT RESOLUTIONS OF THE COST GRID.

Object	Mass-spring simulation		FEM simulation	
	coarse res. (200 lines)	fine res. (7056 lines)	coarse res. (200 lines)	fine res. (7056 lines)
curtain	17 m 48 s	10 h 37 m	1 h 11 m	41 h 16 m
rubberduck	18 m 21 s	10 h 56 m	1 h 16 m	44 h 44 m

generated trajectory points deviate on average by $0.09\,\mathrm{m}$ in an environment of $2.6 \times 9\,\mathrm{m}$ and the deformation costs of the trajectories deviate by $9.4 \pm 5.2\,\%$.

In most cases, the actual trajectories reported by the different planners do not deviate substantially. As the examples depicted in Fig. 6 illustrate, the resulting trajectories are similar. This suggests that our approximative solution provides appropriate trajectories for planning in environments with deformable objects.

B. Determining the Weighting Coefficient

Eq. (4) contains the weighting factor α that trades off the travel costs with deformation costs. To find good values for this factor, we carried out a series of planning experiments with varying values for α. Low values for α result in the fact that the robot traverses objects that are hard to deform in order to obtain a short trajectory. In contrast to this, high values for α will lead to a planning system that

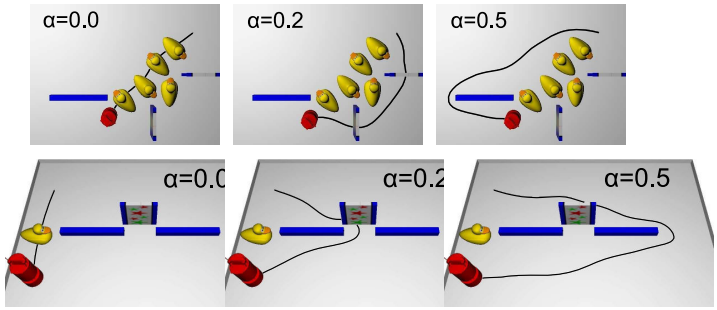


Fig. 7. Different trajectories obtained in two environments depending on the weighting coefficient α.

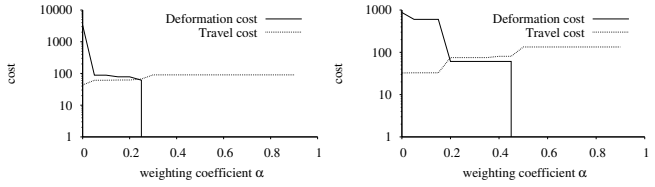


Fig. 8. Deformation and travel costs of executed trajectories depending on the weighting coefficient α. The left plot corresponds to the environment in the first row of Fig. 7 and the right one to the one in the second row.

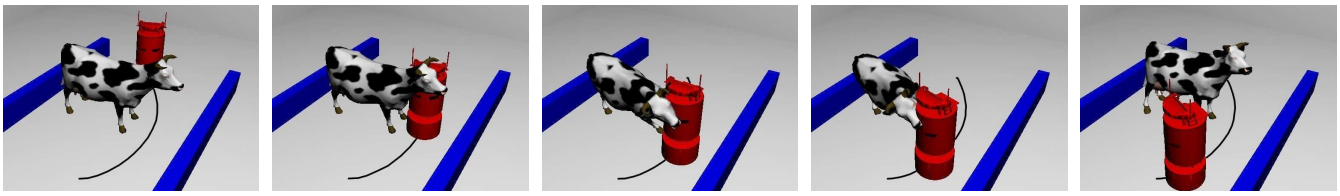


Fig. 9. Example trajectory guiding the robot through a deforming object.

entirely avoids deformations if possible. Examples for such trajectories are depicted in Fig. 7. The corresponding analysis of the deformation and travel costs is shown in Fig. 8. Based on these experiments, we generally set $\alpha = 0.2$. As a result, the robot selects trajectories through easily deformable objects such as curtains and tries to avoid objects that cause high deformation costs such as the rubber ducks.

Finally, Fig. 9 shows a sequence of snapshots taken during a planning experiment. They illustrate that a robot using our planning approach selects trajectories through deformable objects in case the deformation is not too expensive. More examples and animations are available at our web site [9].

VI. CONCLUSIONS

In this paper, we presented an approach to path planning in environments with non-rigid objects. Our planner takes potential deformations of objects into account using a simulation engine that is based on the physically accurate Finite Element method. To improve the efficiency of the planner, our approach uses a pre-computed and object dependent deformation cost function that estimates the deformation cost of path segments relative to the object. The cost function is learned offline and is integrated into a probabilistic roadmap planner. To calculate paths, our system trades off deformation and travel costs. As a result, we obtain highly efficient paths and at the same time avoid computationally expensive simulations during runtime.

Our approach has been implemented and tested exhaustively in environments with deformable objects. The utilization of the approximative cost function leads to a speedup of about four orders of magnitude compared to a system that performs the simulations at runtime.

Despite these encouraging results, we envision several aspects for further improvements. One of our next goals is to acquire models of real obstacles with our robot and estimate their elasto-mechanical parameters, for example by using the method proposed in our previous work [5]. This will allow for applying our system to real world settings and accurately considering the properties of real deformable objects.

VII. ACKNOWLEDGMENTS

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8.

REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[2] E. Anshelevich, S. Owens, F. Lamiraux, and L.E. Kavraki. Deformable volumes in path planning applications. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 2290–2295, 2000.

[3] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[4] O.B. Bayazit, J.-M. Lien, and N.M. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 2126–2133, 2002.

[5] M. Becker and M. Teschner. Robust and efficient estimation of elasticity parameters using the linear finite element method. In *Proc. of Simulation and Visualization*, pages 15–28, 2007.

[6] M.S. Branicky, S.M. LaValle, K. Olson, and L. Yang. Quasi-randomized path planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 1481–1487, 2001.

[7] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, 2005.

[8] C. Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, 2005.

[9] B. Frank. Motion planning with deformable objects, 2008. `http://www.informatik.uni-freiburg.de/~bfrank/defplan`.

[10] R. Gayle, P. Segars, M.C. Lin, and D. Manocha. Path planning for deformable robots in complex environments. In *Proc. of Robotics: Science and Systems (RSS)*, pages 225–232, 2005.

[11] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. In *WSCG*, pages 137–145, 2004.

[12] B. Heidelberger, M. Teschner, J. Spillmann, M. Mueller, M. Gissler, and M. Becker. DefColStudio – interactive deformable modeling framework. `http://cg.informatik.uni-freiburg.de/software.htm`.

[13] C. Holleman, L.E. Kavraki, and J. Warren. Planning paths for a flexible surface patch. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 21–26, 1998.

[14] L.E. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Robotics: The Algorithmic Perspective*, pages 313–325. A.K. Peters, 1998. Proc. of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR).

[15] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[16] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Pub., 1991.

[17] S.M. LaValle. *Planning Algorithms*. Cambridge Univ. Press, 2006.

[18] M. Mueller and M. Gross. Interactive Virtual Materials. In *Graphics Interface*, pages 239–246, 2004.

[19] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[20] S. Rodríguez, J.-M. Lien, and N.M. Amato. Planning motion in completely deformable environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 2466–2471, 2006.

[21] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. In *Proc. of the Conf. on Computer graphics and interactive techniques*, pages 151–160, 1986.

[22] J. Spillmann, M. Becker, and M. Teschner. Non-iterative computation of contact forces for deformable objects. *Journal of WSCG*, 15(1–3):33–40, 2007.

[23] M. Teschner, B. Heidelberger, M. Mueller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *Proc. of Computer Graphics International*, pages 312–319, 2004.

[24] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization (VMV)*, pages 47–54, 2003.

[25] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.P. Cani, F. Faure, N. Magnenat-Thalmann, and W. Strasser. Collision Detection for Deformable Objects. *Computer Graphics Forum*, 24(1):61–81, 2005.

[C16] C. Plagemann, F. Endres, J. Hess, C. Stachniss, and W. Burgard. Monocular range sensing: A non-parametric learning approach. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

Monocular Range Sensing: A Non-Parametric Learning Approach

Christian Plagemann Felix Endres Jürgen Hess Cyrill Stachniss Wolfram Burgard

***Abstract*— Mobile robots rely on the ability to sense the geometry of their local environment in order to avoid obstacles or to explore the surroundings. For this task, dedicated proximity sensors such as laser range finders or sonars are typically employed. Cameras are a cheap and lightweight alternative to such sensors, but do not directly offer proximity information. In this paper, we present a novel approach to learning the relationship between range measurements and visual features extracted from a single monocular camera image. As the learning engine, we apply Gaussian processes, a non-parametric learning technique that not only yields the most likely range prediction corresponding to a certain visual input but also the predictive uncertainty. This information, in turn, can be utilized in an extended grid-based mapping scheme to more accurately update the map. In practical experiments carried out in different environments with a mobile robot equipped with an omnidirectional camera system, we demonstrate that our system is able to produce proximity estimates with an accuracy comparable to that of dedicated sensors such as sonars or infrared range finders.**

Fig. 1. Our approach estimates proximity information from a single image after having learned how visual appearance is related to depth.

I. Introduction

Cameras have become popular sensors in the robotics community. Compared to proximity sensors such as laser range finders, they have the advantage of being cheap, lightweight, and energy efficient. The drawback of cameras, however, is the fact that due to the projective nature of the image formation process, it is not possible to sense depth information directly. From a geometric point of view, one needs at least two images taken from different locations to recover the depth information analytically. An alternative approach, that requires just one monocular camera and that we follow in this work, is to learn from previous experience how visual appearance is related to depth. Such an ability is also highly developed in humans who are able to utilize monocular cues for depth perception [22].

As a motivating example, consider Figure 1, which depicts the (warped) image of an office environment. Overlayed in white, we visualize the most likely area of free space that is predicted by our approach. We explicitly do not try to estimate a depth map for the whole image, as for example Saxana *et al.* [18]. Rather, we aim at learning the function that, given an image, maps measurement directions to their corresponding distances to the closest obstacles. We believe that such a function can be utilized to solve various tasks of mobile robots including local obstacle avoidance, localization, mapping, exploration, or place classification.

The authors are with the University of Freiburg, Department of Computer Science, Georges-Koehler-Allee 79, 79110 Freiburg, Germany
{plagem, endres, hess, stachnis, burgard} @ informatik.uni-freiburg.de

In this paper, we formulate the range estimation task as a supervised regression problem, in which the training set is build by acquiring images of the environment as well as proximity data provided by a laser range finder. We discuss how appropriate visual features can be extracted from the images using algorithms for edge detection and dimensionality reduction. We apply Gaussian processes as the learning framework in our proposed system, since this technique is able to model non-linear functions, offers a direct way of estimating uncertainties for its predictions, and it has proven successful in previous work involving range functions [15].

The paper is organized as follows. After discussing related work, we formalize our problem and introduce the proposed learning framework in Section III. In Section IV we then discuss appropriate visual features and how they can be extracted from images. Section V presents the experimental evaluation of our algorithm as well as an application to the mapping problem. Finally, we conclude in Section VI and give an outlook to future research.

II. Related Work

The problem of recovering geometric properties of a scene from visual measurements is one of the fundamental problems in computer vision and is also frequently addressed in the robotics literature. Stereo camera systems are widely used to estimate the missing depth information that single cameras cannot provide directly. Stereo systems either require a careful calibration to analytically calculate depth using geometric constraints or, as Sinz *et al.* [20] demonstrated,

can be used in combination with non-linear, supervised learning approaches to recover depth information. Often, sets of features such as SIFT [12] are extracted from two images and matched against each other. Then, the feature pairs are used to constrain the poses of the two camera locations and/or the point in the scene that corresponds to the image feature. In this spirit, the motion of the camera is considered by [5], [21]. Sim and Little [19] present a stereo-vision based approach to the SLAM problem, which also includes the recovery of depth information. Their approach contains both the matching of discrete landmarks as well as dense grid mapping using vision cues.

An active way of sensing depth using a single monocular camera is known as *depth from defocus* [8] or *depth from blur*. Corresponding approaches typically adjust the focal length of the camera and analyze the resulting local changes in image sharpness. Torralba and Oliva [24] present an approach for estimating the mean depth of full scenes from single images using spectral signatures. While their approach is likely to improve a large number of recognition algorithms by providing a rough scale estimate, the spatial resolution of their depth estimates does not appear to be sufficient for the problem studied in this paper. Dahlkamp *et al.* [3] learn a mapping from visual input to road traversability in a self-supervised manner.

The problem dealt with in this paper, is closely related to the work of Saxena *et al.* [18], who utilize Markov random fields (MRFs) for reconstructing dense depth maps from single monocular images. An alternative approach that predicts 2D range scans based using reinforcement learning techniques has been presented by Michels et al. [13]. Compared to these methods, our Gaussian process formulation provides the predictive uncertainties for the depth estimates directly, which is not straightforward to achieve in an MRF model. Hoiem *et al.* developed an approach to monocular scene reconstruction based on local features combined with global reasoning [11]. Whereas Han and Zhu presented a Bayesian method for reconstructing the 3D geometry of wire-like objects in simple scenes [10], Delage *et al.* introduced an MRF model on orthogonal plane segments to recover the 3D structure of indoor scenes [6].

As mentioned above, one potential application of the approach described in this paper is to learn occupancy grid maps. This type of maps and an algorithm to update such maps based on ultrasound data has been introduced by Moravec and Elfes [14]. In the past, different approaches to learn occupancy grid maps from stereo vision have been proposed [23], [17]. If the positions of the robot are unknown during the mapping process, the entire task turns into the so-called simultaneous localization and mapping (SLAM) problem. Vision-based techniques have been proposed by Elinas *et al.* [7] and Davison *et al.* [5] to solve this problem. In contrast to the mapping approach presented in this paper, these techniques mostly focus on landmark-based representations.

III. Learning Depth from Monocular Vision Features

The goal of this work is to learn the relationship between visual input and the extent of free space around the robot. By using a regular range sensors, it is comparably easy to acquire training data for this task, so that we can formulate the problem as a supervised learning problem. Figure 2 (a) depicts the configuration of our robot used for data acquisition. An omnidirectional camera system (catadioptric with a parabolic mirror) is mounted on top of a SICK laser range finder. This setup allows the robot to perceive the full surrounding area at every time step as the two example images in Figure 2 (b) and (c) illustrate. The omnidirectional images can be mapped directly to the laser scans, since both measurements can be represented in a common, polar coordinate system. Note that our approach is not restricted to omnidirectional cameras in principle. However, the correspondence between range measurements and omnidirectional images is a more direct one and the field of view is considerably larger compared to standard perspective optics.

A. A Gaussian Process Model for Range Functions

We extract for every viewing direction α a vector of visual features $\mathbf{x}$ from the images and phrase the problem as learning the range function $f(\mathbf{x}) = y$ that maps the visual input $\mathbf{x}$ to distances y. We learn this function in a supervised manner using a training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ of observed features $\mathbf{x}_i$ and corresponding laser range measurements y_i. If we place a Gaussian process (GP) prior [16] on the non-linear function f, i.e., we assume that all range samples y indexed by their corresponding feature vectors $\mathbf{x}$ are jointly Gaussian distributed, we obtain

$$f(\mathbf{x}^*) \quad \sim \quad \mathcal{N}(\boldsymbol{\mu}, \sigma) \tag{1}$$

with

$$\boldsymbol{\mu} \quad = \quad \mathbf{k}^{*T}(K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{2}$$

$$\sigma \quad = \quad k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^{*T}(K + \sigma_n^2 I)^{-1}\mathbf{k}^* \tag{3}$$

as the predictive distribution for the range function at new query points $\mathbf{x}^*$. Here, K denotes the $n \times n$-dimensional covariance matrix constructed as $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ using a covariance function k, which is parameterized by a set of hyper-parameters $\boldsymbol{\theta}$. The term $\mathbf{y}$ denotes the vector of given target values from the training set, $\mathbf{k}^*$ stands for the vector of covariances between the new query point $\mathbf{x}^*$ and the training points with $\mathbf{k}_i^* = k(\mathbf{x}^*, \mathbf{x}_i)$. Finally σ_n denotes a global noise parameter. In this work, we apply the often-used squared exponential covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2\ell^2}|\mathbf{x}_p - \mathbf{x}_q|\right), \tag{4}$$

which depends on the Euclidian distance between points $\mathbf{x}_p$ and $\mathbf{x}_q$ as well as on the amplitude parameter σ_f^2 and the length-scale ℓ. These parameters as well as the noise parameter σ_n in Eq. (2) and (3) can be learned from data. Starting from an initial guess, we apply conjugate gradient-based optimization to find the values for $\{\ell, \sigma_f^2, \sigma_n^2\}$ that

(a) (b) (c)

Fig. 2. The left diagram depicts our experimental setup: the training set was recorded using a mobile robot equipped with an omnidirectional camera (monocular camera with a parabolic mirror) as well as a laser range finder. The next two images illustrate two typical omnidirectional images recorded at the University of Freiburg (b) and at the DFKI in Saarbruecken (c).

minimize the negative log marginal likelihood of the GP model.

A particularly useful property of Gaussian processes for our application is the availability of the predictive uncertainty (see Eq. (3)) at every query point. This means, that visual features $\mathbf{x}^*$, which lie close to points $\mathbf{x}$ of the training set result in more confident predictions than features, which fall into a less densely sampled region of feature space.

IV. Features in Omnidirectional Images

The part of an omnidirectional image which is most strongly correlated with the distance to the nearest obstacle in a certain direction α is the strip of pixels oriented in the same direction and going from the center of the image to its margins. With the type of camera used in our experiments, such strips have a dimensionality of 420 (140 pixels, each having a *hue*, *saturation*, and a *value* component). In order to make these strips easier accessible to filter operators, we warp the omnidirectional images (e.g., see Figure 2 (b) and (c)) into panoramic views (e.g., see Figure 5 (a)), such that angles in the polar representation now correspond to column indices in the panoramic one. This transformation allows us to replace complicated image operations in the polar domain by easier and more robust ones. In the following, we describe several ways of extracting useful low-dimensional feature vectors $\mathbf{x}$ from the 420-dimensional image columns, which can then be directly used to index the training and test targets in the GP framework.

1) Unsupervised Dimensionality Reduction: As a classic way of reducing the complexity of a data set, we applied the principle component analysis (PCA) to the raw 420-dimensional pixel vectors that are contained in the columns of the panoramic images. The PCA is implemented using eigenvalue decomposition of the covariance matrix of the training vectors. It yields a linear transformation which brings the input vectors into a new basis such that their dimensions are now ordered by the amount of data-set variance they carry. In this way, we can truncate the vectors to a few components without losing a large amount of information. The diagram in Figure 3 depicts the relative amount of variance that is explained for two different data sets when truncating the transformed data vectors after a certain number of components. In the experiments reported below, we trained the PCA on 600 input images and retained the first six principle components. Our experiments revealed that the *value* channel of the visual input is more important than *hue* and *saturation* for our task. The GP model learned with these 6-dimensional features is termed *PCA-GP* in the experimental section.

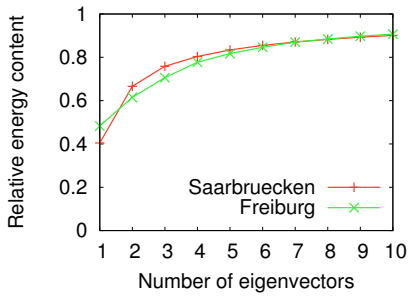


Fig. 3. The amount of variance explained by the the first principle components (eigenvectors) of the pixel columns in the two data sets.

2) Edge-based Features: The PCA is an unsupervised method that does not take into account prior information that might be available about the task to be solved – in this case, the fact that distances to the closest obstacles are to be predicted. Driven by the observation that, especially in indoor environments, there is a strong correlation between the extent of free space and the presence of horizontal edge features in the panoramic image, we also assessed the potential of edge-type features in our approach.

Laws' convolution masks [4] provide an easy way of constructing local feature extractors for discretized signals. The idea is to define three basic convolution masks

- $L_3 = (1, 2, 1)^T$ (Weighted Sum: Averaging),
- $E_3 = (-1, 0, 1)^T$ (First difference: Edges),
- $S_3 = (-1, 2, -1)^T$ (Second difference: Spots),

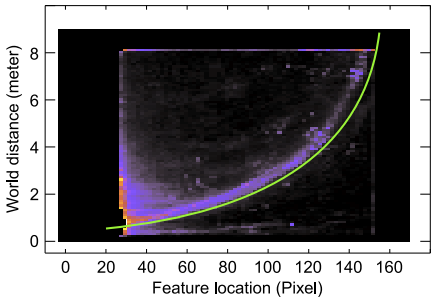


Fig. 4. Feature histogram for *Laws5+LMD* edge features. Each cell in the histogram is indexed by the pixel location of the edge feature (x-axis) and the length of the corresponding laser beam (y-axis). The optimized (parametric) mapping function that is used as a benchmark in our experiments is overlaid in green.

each having a different effect on (1-dimensional) patterns, and to construct more complex filters by a combination of the basic masks. In our application domain, we obtained good results with the (2-dimensional) directed edge filter $E_5 L_5^T$, which is the outer product of E_5 and L_5. Here, E_5 is a convolution of E_3 with L_3 and L_5 denotes L_3 convolved with itself. After filtering with this mask, we apply an optimized threshold to yield a binary response. This feature type is denoted as *Laws5* in the experimental section. As another well-known feature type, we applied the $E_3 L_3^T$ filter, i.e., the Sobel operator, in conjunction with Canny's algorithm [2]. This filter yields binary responses at the image locations with maximal grey-value gradients in gradient direction. We denote this feature type as *Laws3+Canny* in Section V. For both edge detectors, *Laws5* and *Laws3+Canny*, we search along each image column for the first detected edge. This pixel index then constitutes the feature value.

To increase the robustness of the edge detectors described above, we applied *lightmap damping* as an optional preprocessing step to the raw images. This means that, in a first step, a copy of the image is converted to gray scale and strongly smoothed with a Gaussian filter, such that every pixel represents the brightness of its local environment. This is referred to as the *lightmap*. The brightness of the original image is then scaled with respect to the lightmap, such that the *value* component of the color is increased in dark areas and decreased in bright areas. In the experimental section, this operation is marked by adding *+LMD* to the feature descriptions.

All parameters involved in the edge detection procedures described above were optimized to yield features that lie as close as possible to the laser end points projected onto the omnidirectional image using the acquired training set. For our regression model, we can now construct 4-dimensional feature vectors $\mathbf{x}$ consisting of the Canny-based feature, the Laws5-based feature, and both features with additional preprocessing using lightmap-damping. Since every of these individual features captures slightly different aspects of the visual input, the combination of all in what we call the *Feature-GP* yields more accurate predictions than any single one.

As a benchmark for predicting range information from edge features, we also evaluated the individual features directly. For doing so, we fitted a parametric function to training samples of feature-range pairs. This mapping function computes for each pixel location of an edge feature the length of the corresponding laser beam. The diagram in Figure 4 depicts the feature histogram for the *Laws5+LMD* features from one of our test runs that was used for the optimization. The color of a cell (x, y) in this diagram encodes the relative amount of features that were extracted at the pixel location x (measured from the center of the omnidirectional image) and that have a corresponding laser beam with a length of y in the training set. The optimized projection function is overlayed in green.

V. Experiments

The experiments presented in this section are designed to evaluate how well the proposed system is able to estimate range data from single monocular camera images. We document a series of different experiments: First, we evaluate the accuracy of the estimated range scans using the individual edge features directly, the *PCA-GP*, and the *Feature-GP*, which constitutes our regression model with the 4 edge-based vision features as input dimensions. Then, we illustrate how these estimates can be used to build grid maps of the environment. We also evaluated, whether applying the GBP model [15] as a post-processing step to the predicted range scans can further increase the prediction accuracy. The GBP model places a Gaussian process prior on the range function (rather than on the function that maps features to distances) and, thus, also models angular dependencies. We denote these models by *Feature-GP+GBP* and *PCA-GP+GBP*.

The two data sets used for the experiments have been recorded using a mobile robot equipped with a laser scanner, an omnidirectional camera, and odometry sensors at the AIS lab at the University of Freiburg (Figure 2 (b)) and at the DFKI lab in Saarbrücken (Figure 2 (c)). The two environments have quite different characteristics – especially in the visual aspects. While the environment in Saarbrücken mainly consists of solid, regular structures and a homogeneously colored floor, the lab in Freiburg exhibits many glass panes, an irregular, wooden floor, and challenging lighting conditions.

A. Accuracy of Range Predictions

We evaluated eight different system configurations, each on both test data sets. Table I summarizes the average RMSE (root mean squared error) obtained for the individual scenarios. The error is measured as the deviation of the range predictions using the visual input from the corresponding laser ranges recorded by the sensor. The first four configurations, referred to as C1 to C4, apply the optimized mapping functions for the different edge features (see Figure 4). Depending on the data, the features provide estimates with an RMSE of between 1.7 m and 3 m. We then evaluated the configurations C5 and C6 which use the four edge-based features as inputs to a Gaussian process model as described

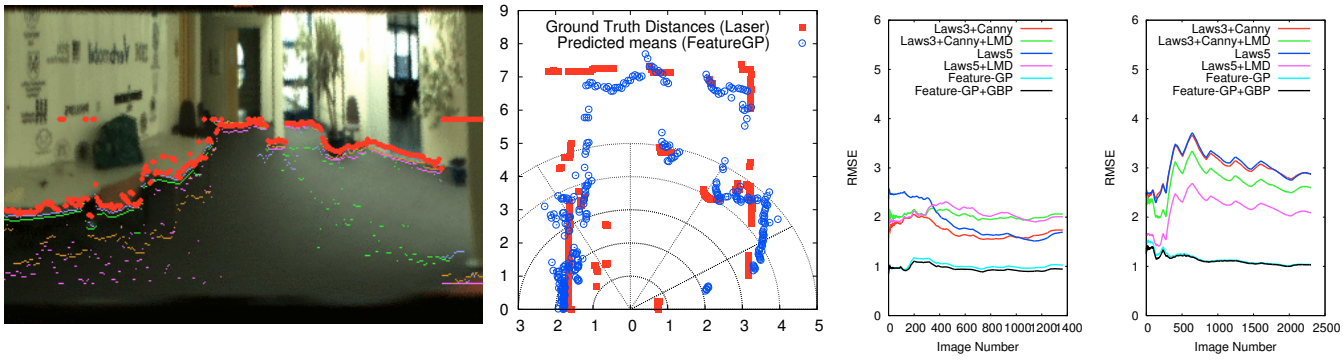


Fig. 5. (a) Estimated ranges projected back onto the camera image using the feature detectors directly (small dots) and using the *Feature-GP* model (red points). (b) Prediction results and the true laser scan at one of the test locations. The evolution of the root mean squared error (RSME) for the individual images of the Saarbrücken (c) and Freiburg (d) data sets.

in Section III to learn the mapping from the feature vectors to the distances. The learning algorithm was able to perform range estimation with an RMSE of around 1 m. Note that we measure the prediction error relative to the recorded laser beams rather than to the true geometry of the environment. Thus, we report a conservative error estimate that also includes errors due to reflected laser beams contained in the test set. To give a visual impression of the prediction accuracy of the *Feature-GP*, we give a typical laser scan and the mean predictions in diagram (b) of Figure 5.

TABLE I

AVERAGE ERRORS OBTAINED WITH THE DIFFERENT METHODS

Configuration	RMSE on test set Saarbrücken	Freiburg
C1: Laws5	1.70m	2.87m
C2: Laws5+LMD	2.01m	2.08m
C3: Laws3+Canny	1.74m	2.87m
C4: Laws3+Canny+LMD	2.06m	2.59m
C5: Feature-GP	1.04m	1.04m
C6: Feature-GP+GBP	**1.03m**	**0.94m**
C7: PCA-GP	1.24m	1.40m
C8: PCA-GP+GBP	1.22m	1.41m

As configuration C7, we evaluated the *PCA-GP* approach that does not require engineered features, but rather works on the low-dimensional representation of the raw visual input computed using the PCA. The resulting 6-dimensional feature vector is used as input to the Gaussian process model. With an RMSE of 1.2 m to 1.4 m, the *PCA-GP* outperforms all four engineered features, but is not as accurate as the *Feature-GP*. For configurations C6 and C8, we predicted the ranges per scan using the two different methods and additionally applied the GBP model [15] to incorporate angular dependencies between the predicted beams. This post-processing step yields slight improvements compared to the original variants C5 and C7.

Figure 5 (a) depicts an example images showing the predictions based on the individual vision features and the *Feature-GP*. It can be clearly seen from the image, that the different edge-based features model different parts of the range scan well. The *Feature-GP* fuses these unreliable estimates to achieve high accuracy on the whole scan. The result of the *Feature-GP+GBP* variant for the same situation is given in Figure 1. The evolution of the RMSE for the different methods over time is given in Figures 5 (c) and (d). As can be seen from the diagrams, the prediction using the *Feature-GP* model outperforms the other techniques and achieves a near-constant error rate.

B. Application to Mapping

Our approach can be applied to a variety of robotics tasks such as obstacle avoidance, localization, or mapping. To illustrate this, we show how to learn a grid map of the environment from the predictive range distributions. Compared to occupancy grid mapping where one estimates for each cell the probability of being occupied or free, we use the so called *reflection probability maps*. A cell of such a map models the probability that a laser beam passing this cell is reflected or not. Reflection probability maps, which are learned using the so called *counting model*, have the advantage of requiring no hand-tuned sensor model such as occupancy grid maps (see [1] for further details). The reflection probability m_i of a cell i is given by $m_i = \alpha_i/(\alpha_i + \beta_i)$ where α_i is the number of times an observation hits the cell, i.e., ends in it, and β_i is the number of misses, i.e., the number of times a beam has intercepted a cell without ending in it. Since our GP approach does not estimate a single laser end point, but rather a full (normal) distribution $p(z)$ of possible end points, we have to integrate over this distribution. More precisely, for each grid cell c_i, we update the cell's reflectance values according to the predictive distribution $p(z)$ according to the following formulas:

$$\alpha_i \leftarrow \alpha_i + \int_{z \in c_i} p(z)\, dz \tag{5}$$

$$\beta_j \leftarrow \beta_i + \int_{z > c_i} p(z)\, dz\ . \tag{6}$$

Note that for perfectly accurate predications, the extended update rule is equivalent to the standard formula stated above.

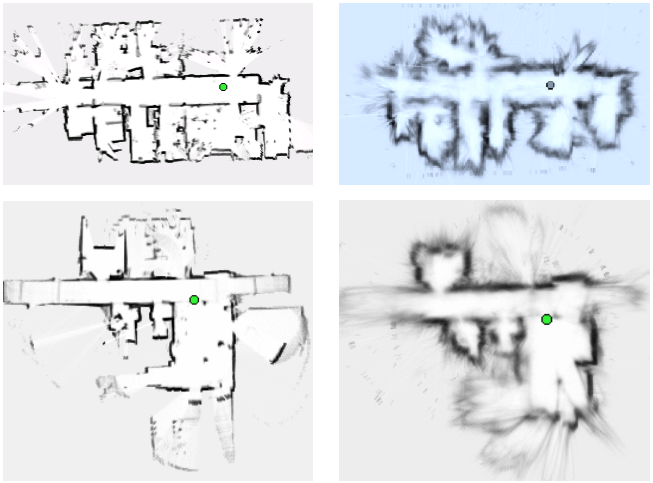


Fig. 6. Maps of the Freiburg AIS lab (top row) and DFKI Saarbrücken (bottom row) using real laser data (left) and the predictions of the *Feature-GP* (right).

We applied this extended reflection probability mapper to the trajectories and range predictions that resulted from the experiments reported on above. Figure 6 gives the laser-based maps using a standard mapper (left column) and the extended mapper using the predicted ranges (right column) for both environments (Freiburg on top and Saarbrücken below). In both cases, it is possible to build an accurate map, which is comparable to maps obtained with infrared proximity sensors [9] or sonars [23].

VI. Conclusions

We presented a novel approach for predicting range functions from single images recorded with a monocular camera. Our model is based on a Gaussian process model for regression, utilizing edge-based features extracted from the image or, alternatively, using the PCA to find a low-dimensional representation of the visual input in an unsupervised manner. Both models outperform the optimized individual features. We showed in experiments with a real robot that the range predictions are accurate enough to feed them into an extended mapping algorithm for predictive range distributions and that the resulting maps are comparable to maps obtained with infrared or sonar sensors.

In future research we would like to evaluate alternative techniques for dimensionality reduction, especially those taking the actual task into account (supervised PCA, LDA) or others that are directly integrated into the GP framework. Furthermore, we would like to evaluate our approach in other robotics tasks, such as exploration or place classification.

Acknowledgments

We would like to thank Kristian Kersting for the fruitful discussions and Andrzej Pronobis and Jie Luo for creating the data sets. This work has partly been supported by the EC under contract numbers FP6-004250-CoSy, FP6-IST-034120, and FP6-IST-045144, by the DFG under contract number SFB/TR-8, and by the German Ministry for Education and Research (BMBF) through the DESIRE project.

References

[1] W. Burgard, C. Stachniss, and D. Haehnel. *Autonomous Navigation in Dynamic Environments*, volume 35 of *STAR Springer tracts in advanced robotics*, chapter Mobile Robot Map Learning from Range Data in Dynamic Environments. Springer Verlag, 2007.

[2] F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 679–714, 1986.

[3] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G.R. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

[4] E. R. Davies. Laws texture energy in texture. In *Machine Vision: Theory, Algorithms, Practicalities*. Acedemic Press, 1997.

[5] A. Davision, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[6] E. Delage, H. Lee, and A.Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *Proceedings of the 12th International Symposium of Robotics Research (ISRR)*, 2005.

[7] P. Elinas, R. Sim, and J. J. Little. σSLAM: Stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. of ICRA*, 2006.

[8] P. Favaro and S. Soatto. A geometric approach to shape from defocus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):406–417, 2005.

[9] Y.S. Ha and H.H. Kim. Environmental map building for a mobile robot using infrared range-finder sensors. *Advanced Robotics*, 18(4):437–450, 2004.

[10] F. Han and S.-C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. In *IEEE Intern. Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis (HLK*, page 12, Washington, DC, USA, 2003.

[11] D. Hoiem, A.A. Efros, and M. Herbert. Recovering surface layout from an image. *IJCV*, 75(1), October 2007.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[13] J. Michels, A. Saxena, and A.Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*, pages 593–600, 2005.

[14] H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 116–121, St. Louis, MO, USA, 1985.

[15] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A nonparametric bayesian measurement model for range finders. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[16] C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[17] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, New Orleans, LA, USA, 2004.

[18] A. Saxena, S.H. Chung, and A.Y. Ng. 3-d depth reconstruction from a single still image. *Intern. Journal of Computer Vision (IJCV)*, 2007.

[19] R. Sim and J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2082–2089, 2006.

[20] F. Sinz, J. Quinonero-Candela, G. Bakir, C. Rasmussen, and M. Franz. Learning depth from stereo. In *26th DAGM Symposium*, 2004.

[21] H. Strasdat, C. Stachniss, M. Bennewitz, and W. Burgard. Visual bearing-only simultaneous localization and mapping with improved feature matching. In *Fachgespräche Autonome Mobile Systeme (AMS)*, 2007.

[22] G. Swaminathan and S. Grossberg. Laminar cortical mechanisms for the perception of slanted and curved 3-D surfaces and their 2-D pictorical projections. *J. Vis.*, 2(7):79–79, 11 2002.

[23] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

[24] A. Torralba and A. Oliva. Depth estimation from image structure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 2002.

[C17] G. Grisetti, D. Lordi Rizzini, C. Stachniss, E. Olson, and W. Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning

Giorgio Grisetti* Dario Lodi Rizzini‡ Cyrill Stachniss* Edwin Olson† Wolfram Burgard*

***Abstract*— In this paper, we address the problem of incrementally optimizing constraint networks for maximum likelihood map learning. Our approach allows a robot to efficiently compute configurations of the network with small errors while the robot moves through the environment. We apply a variant of stochastic gradient descent and use a tree-based parameterization of the nodes in the network. By integrating adaptive learning rates in the parameterization of the network, our algorithm can use previously computed solutions to determine the result of the next optimization run. Additionally, our approach updates only the parts of the network which are affected by the newly incorporated measurements and starts the optimization approach only if the new data reveals inconsistencies with the network constructed so far. These improvements yield an efficient solution for this class of online optimization problems.**

Our approach has been implemented and tested on simulated and on real data. We present comparisons to recently proposed online and offline methods that address the problem of optimizing constraint network. Experiments illustrate that our approach converges faster to a network configuration with small errors than the previous approaches.

I. Introduction

Maps of the environment are needed for a wide range of robotic applications such as search and rescue, automated vacuum cleaning, and many other service robotic tasks. Learning maps has therefore been a major research focus in the robotics community over the last decades. Learning maps under uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. The approaches mainly differ in the underlying estimation technique. Typical techniques are Kalman filters, information filters, particle filters, network based methods which rely on least-square error minimization techniques.

Solutions to the SLAM problem can be furthermore divided into online an offline methods. Offline methods are so-called batch algorithms that require all the data to be available right from the beginning [1], [2], [3]. In contrast to that, online methods can re-use an already computed solution and update or refine it. Online methods are needed for situations in which the robot has to make decisions based on the model of the environment during mapping. Exploring an unknown environment, for example, is a task of this category. Popular online SLAM approaches such as [4], [5] are based on the Bayes' filter. Recently, also incremental maximum-likelihood approaches have been presented as an effective alternative [6], [7], [8].

*Department of Computer Science, University of Freiburg, Germany. †MIT, 77 Massachusetts Ave., Cambridge, MA 02139-4307, USA. ‡Department of Information Engineering, University of Parma, Italy.

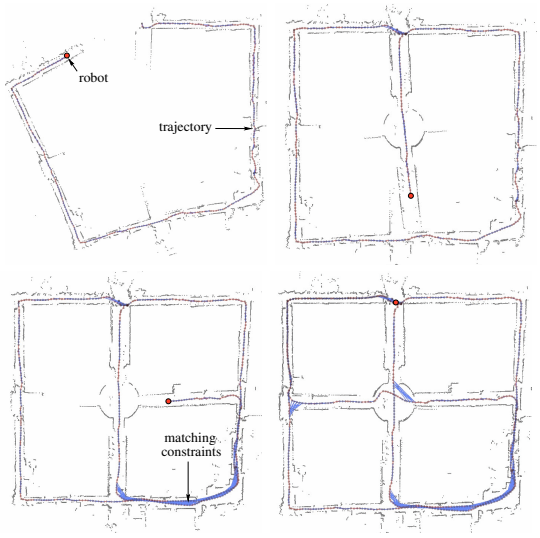


Fig. 1. Four snapshots created while incrementally learning a map.

In this paper, we present an efficient online optimization algorithm which can be used to solve the so-called "graph-based" or "network-based" formulation of the SLAM problem. Here, the poses of the robot are modeled by nodes in a graph and constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes. Our method belongs to the same class of techniques of Olson's algorithm or MLR [8]. It focuses on computing the best map and it assumes that the constraints are given. Techniques like the ATLAS framework [9] or hierarchical SLAM [10], for example, can be used to obtain the necessary data associations (constraints). They also apply a global optimization procedure to compute a consistent map. One can replace these optimization procedures by our algorithm and in this way make them more efficient.

Our approach combines the ideas of adaptive learning rates with a tree-based parameterization of the nodes when applying stochastic gradient descent. This yields an online algorithm that can efficiently compute network configurations with low errors. An application example is shown in Figure 1. It depicts four snapshots of our online approach during a process of building a map from the ACES dataset.

II. Related Work

A large number of mapping approaches has been presented in the past and a variety of different estimation techniques have been used to learn maps. One class of approaches uses constraint networks to represent the relations between poses and observations.

Lu and Milios [1] were the first who used constraint networks to address the SLAM problem. They proposed a brute force method that seeks to optimize the whole network at once. Gutmann and Konolige [11] presented an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Frese *et al.* [8] described a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions.

Olson *et al.* [2] were the first who applied a variant of stochastic gradient descent to compute solutions to this family of problems. They propose a representation of the nodes which enables the algorithm to perform efficient updates. Our previously presented method [3] introduced the tree parameterization that is also used in this paper. Subsequently, Olson *et al.* [6] presented an online variant of their method using adaptive learning rates. In this paper, we integrate such learning rates into the tree-based parameterization which yields a solution to the online SLAM problem that outperforms the individual methods.

Kaess *el al.* [7] proposed an on-line version of the smoothing and mapping algorithm for maximum likelihood map estimation. This approach relies on a QR factorization of the information matrix and integrates the new measurements as they are available. Using the QR factorization, the poses of the nodes in the network can be efficiently retrieved by back substitution. Additionally they keep the matrices sparse via occasional variable reordering. Frese [12] proposed the Treemap algorithm which is able to perform efficient updates of the estimate by ignoring the weak correlations between distant locations.

The contribution of this paper is an efficient online approach for learning maximum likelihood maps. It integrates adaptive learning rates into a tree-based network optimization technique using a variant of stochastic gradient descent. Our approach presents an efficient way of selecting only the part of the network which is affected by newly incorporated data. Furthermore, it allows to delay the optimization so that the network is only updated if needed.

III. Stochastic Gradient Descent for Maximum Likelihood Mapping

Approaches to graph-based SLAM focus on estimating the most likely configuration of the nodes and are therefore referred to as maximum-likelihood (ML) techniques [8], [1], [2]. The approach presented in this paper also belongs to this class of methods.

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. Let $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ be a vector of parameters which describes a configuration of the nodes. Let δ_{ji} and Ω_{ji} be respectively the mean and the information matrix of an observation of node j seen from node i. Let $f_{ji}(\mathbf{x})$ be a function that computes a zero noise observation according to the current configuration of the nodes j and i.

Given a constraint between node j and node i, we can define the *error* e_{ji} introduced by the constraint as

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

as well as the *residual* $r_{ji} = -e_{ji}(\mathbf{x})$. Let $\mathcal{C} = \{\langle j_1, i_1\rangle, \ldots, \langle j_M, i_M\rangle\}$ be the set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists. The goal of a ML approach is to find the configuration $\mathbf{x}^*$ of the nodes that minimized the negative log likelihood of the observations. Assuming the constraints to be independent, this can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i\rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{2}$$

In the remainder of this section we describe how the general framework of stochastic gradient descent can be used for minimizing Eq. (2) and how to construct a parameterization of the network which increases the convergence speed.

A. Network Optimization using Stochastic Gradient Descent

Olson *et al.* [2] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to address the compute the most likely configuration of the network's nodes. The approach minimizes Eq. (2) by iteratively selecting a constraint $\langle j, i\rangle$ and by moving the nodes of the network in order to decrease the error introduced by the selected constraint. Compared to the standard formulation of gradient descent, the constraints are not optimized as a whole but individually. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji} \tag{3}$$

Here $\mathbf{x}$ is the set of variables describing the locations of the poses in the network and $\mathbf{H}^{-1}$ is a preconditioning matrix. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, r_{ji} is the residual, and λ is the learning rate which decreases with the iteration. For a detailed explanation of Eq. (3), we refer the reader to our previous works [3], [2].

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing subsets of nodes, one subset for each constraint. Whenever time a solution for one of these subproblems is found, the network is updated accordingly. Obviously, updating the different constraints one after each other can have antagonistic effects on the corresponding subsets of variables. To avoid infinitive oscillations, one uses the learning rate λ to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different sub-problems to asymptotically converge towards an equilibrium point that is the solution reported by the algorithm.

B. Tree Parameterization

The poses $\mathbf{p} = \{p_1, \ldots, p_n\}$ of the nodes define the configuration of the network. The poses can be described by a vector of *parameters* $\mathbf{x}$ such that a bidirectional mapping between $\mathbf{p}$ and $\mathbf{x}$ exists. The parameterization defines the subset of variables that are modified when updating a constraint. An efficient way of parameterizing the node is to

use a tree. One can construct a spanning tree (not necessarily a minimum one) from the graph of poses. Given such a tree, we define the parameterization for a node as

$$x_i \quad = \quad p_i - p_{\text{parent}(i)}, \tag{4}$$

where $p_{\text{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (4), the tree stores the differences between poses. This is similar in the spirit to the incremental representation used in the Olson's original formulation, in that the difference in pose positions (in global coordinates) is used rather than pose-relative coordinates or rigid body transformations.

To obtain the difference between two arbitrary nodes based on the tree, one needs to traverse the tree from the first node upwards to the first common ancestor of both nodes and then downwards to the second node. The same holds for computing the error of a constraint. We refer to the nodes one needs to traverse on the tree as the path of a constraint. For example, $\mathcal{P}_{ji}$ is the path from node i to node j for the constraint $\langle j, i \rangle$. The path can be divided into an ascending part $\mathcal{P}_{ji}^{[-]}$ of the path starting from node i and a descending part $\mathcal{P}_{ji}^{[+]}$ to node j. We can then compute the residual in the global frame by

$$r'_{ji} \quad = \quad \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} + R_i \delta_{ji}. \tag{5}$$

Here R_i is the homogeneous rotation matrix of the pose p_i. It can be computed according to the structure of the tree as the product of the individual rotation matrices along the path to the root. Note that this tree does not replace the graph as an internal representation. The tree only defines the parameterization of the nodes.

Let $\Omega'_{ji} = R_i \Omega_{ji} R_i^T$ be the information matrix of a constraint in the global frame. According to [2], we compute an approximation of the Jacobian as

$$J'_{ji} \quad = \quad \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}}, \tag{6}$$

with $\mathcal{I}_k = (0 \ \cdots \ 0 \ \underbrace{I}_{k^{\text{th}} \text{ element}} \ 0 \ \cdots \ 0)$. Then, the update of a constraint turns into

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}, \tag{7}$$

where $|\mathcal{P}_{ji}|$ refers to the number of nodes in $\mathcal{P}_{ji}$. In Eq. (7), we replaced the preconditioning matrix $\mathbf{H}^{-1}$ with its scaled approximation $\mathbf{M}^{-1}$ as described in [2]. This prevents from a computationally expensive matrix inversion.

Let the *level* of a node be the distance in the tree between the node itself and the root. We define the *top node* of a constraint as the node on the path with the smallest level. Our parameterization implies that updating a constraint will never change the configuration of a node with a level smaller than the level of the top node of the constraint.

In principle, one could apply the technique described in this section as a batch algorithm to an arbitrarily constructed spanning tree of the graph. However, our proposed method uses a spanning tree which can be constructed incrementally, as described in the next section.

IV. Online Network Optimization

The algorithm presented in the previous section is a batch procedure. At every iteration, the poses of all nodes in the network are optimized. The fraction of the residual used in updating every constraint decreases over time with the learning rate λ, which evolves according to an harmonic progression. During online optimization, the network is dynamically updated to incorporate new movements and observations. In theory, one could also apply the batch version of our optimizer to correct the network. This, however, would require to compute a solution from scratch each time the robot moves or makes an observation which would obviously lead to an inefficient algorithm.

In this section we describe an incremental version of our optimization algorithm, which is suitable for solving online mapping problems. As pointed in [6] an incremental algorithm should have the following properties:

1) Every time a constraint is added to the network, only the part of the network which is affected by that constraint should be optimized. For example, when exploring new terrain, the effects of the optimization should not perturb distant parts of the graph.
2) When revisiting a known region of the environment it is common to re-localize the robot in the previously built map. One should use the information provided by the re-localization to compute a better initial guess for the position of the newly added nodes.
3) To have a consistent network, performing an optimization step after adding each constraint is often not needed. This happens when the newly added constraints are adequately satisfied by the current network configuration. Having a criterion for deciding when to perform unnecessary optimizations can save a substantial amount of computation.

In the remainder of this section, we present four improvements to the algorithm so that it satisfies the discussed properties.

A. Incremental Construction of the Tree

When constructing the parameterization tree online, we can assume that the input is a sequence of poses corresponding to a trajectory of the robot. In this case, subsequent poses are located closely together and there exist constraints between subsequent poses resulting from odometry or scan-matching. Further constraints between arbitrary nodes result from observations when revisiting a place in the environment.

We proceed as follows: the oldest node is the root of the tree. When adding a node i to the network, we choose as its parent the oldest node for which a constraint to the node i exists. Such a tree can be constructed incrementally since adding a new node does not require to change the existing parts of the tree.

The pose p_i and parameter x_i of a newly added node i is initialized according to the position of the parent node and

the connecting constraint as

$$p_i = p_{\text{parent}(i)} \oplus \delta_{i,\text{parent}(i)} \quad (8)$$

$$x_i = p_i - p_{\text{parent}(i)}. \quad (9)$$

The parent node represents an already explored part of the environment and the constraint between the new node and the parent can be regarded as a localization event in an already constructed map, thus satisfying Property 2. As shown in the experiments described below, this initialization appears to be a good heuristic for determining the initial guess of the pose of a newly added node.

B. Constraint Selection

When adding a constraint $\langle j, i \rangle$ to the graph, a subset of nodes needs to be updated. This set depends on the topology of the network and can be determined by a variant of breadth first visit. Let $\mathcal{G}_{j,i}$ be the minimal subgraph that contains the added constraint and has only one constraint to the rest of the graph. Then, the nodes that need to be updated are all nodes of the minimal subtree that contains $\mathcal{G}_{j,i}$. The precise formulation on how to efficiently determine this set is given by Algorithm 1.

Data: $\langle j, i \rangle$: the constraint, $\mathcal{G}$: the graph, $\mathcal{T}$: the tree.
Result: $\mathcal{N}_{ji}$: the set of affected nodes, $\mathcal{E}_{ji}$: the affected constraints.
Queue $f = \text{childrenOf}(\text{topNode}(\langle j, i \rangle))$;
$\mathcal{E}_{ji} := \text{edgesToChildren}(\text{topNode}(\langle j, i \rangle))$;
foreach $\langle a, b \rangle \in \mathcal{E}_{ji}$ **do**
  $\langle a, b \rangle .mark = true$;
end
while $f \neq \{\}$ **do**
  Node $n := \text{first}(f)$;
  $n.mark := true$
  foreach $\langle a, b \rangle \in \text{edgesOf}(n)$ **do**
    if $\langle a, b \rangle .mark = true$ **then**
      continue;
    end
    Node $m := (a = n)?b : a$;
    if $m = \text{parent}(n)$ or $m.mark = true$ **then**
      continue;
    end
    $\langle a, b \rangle .mark = true$;
    $\mathcal{E}_{ji} := \mathcal{E}_{ji} \cup \{\langle a, b \rangle\}$;
    if $\langle a, b \rangle \in \mathcal{T}$ **then**
      $f := f \cup \{m\}$;
    else
      $f := f \cup \text{childrenOf}(\text{topNode}(\langle a, b \rangle))$;
    end
  end
  $f := \text{removeFirst}(f)$;
  $\mathcal{N}_{ji} := \mathcal{N}_{ji} \cup \{n\}$;
end

Algorithm 1: Construction of the set of nodes affected by a constraint. For readability we assume that the frontier f can contain only the nodes which are not already marked.

Note that the number of nodes in $\mathcal{G}_{j,i}$ does depend only on the root of the tree and on the overall graph. It contains all variables which are affected by adding the new costraint $\langle i, j \rangle$.

C. Adaptive Learning Rates

Rather than using one learning rate λ for all nodes, the incremental version of the algorithm uses spatially adaptive learning rates introduced in [6]. The idea is to assign an individual learning rate to each node, allowing different parts of the network to be optimized at different rates. These learning rates are initialized when a new constraint is added to the network and they decrease with each iteration of the algorithm. In the following, we describe how to initialize and update the learning rates and how to adapt the update of the network specified in Eq. (7).

a) Initialization of the learning rates: When a new constraint $\langle j, i \rangle$ is added to the network, we need to update the learning rates for the nodes $\mathcal{N}_{ji}$ determined in the previous section. First, we compute the learning rate λ'_{ji} for the newly introduced information. Then, we propagate this learning rate to the nodes $\mathcal{N}_{ji}$.e

A proper learning rate is determined as follows. Let β_{ji} be the fraction of the residual that would appropriately fuse the previous estimate and the new constraint. Similar to a Kalman filter, β_{ji} is determined as

$$\beta_{ji} = \Omega_{ji}(\Omega_{ji} + \Omega_{ji}^{\text{graph}})^{-1}, \quad (10)$$

where Ω_{ji} is the information matrix of the new constraint, and $\Omega_{ji}^{\text{graph}}$ is an information matrix representing the uncertainty of the constraints in the network. Based on Eq. (10), we can compute the learning rate λ'_{ji} of the new constraint as

$$\lambda'_{ji} = \text{maxrow}\left(\frac{1}{|\mathcal{P}_{ji}|}(\beta_{ji} \oslash \mathbf{M}\Omega'_{ji})\right). \quad (11)$$

Here $\oslash$ represents the row by row division (see [6] for further details). The learning rate of the constraint is then propagated to all nodes $k \in \mathcal{N}_{ji}$ as

$$\lambda_k \leftarrow \max(\lambda_k, \lambda'_{ji}), \quad (12)$$

where λ_k is the learning rate of the node k. According to Eq. (11) constraints with large residuals result in larger learning rate increases than constraints with small residuals.

b) Update of the network: When updating the network, one has to consider the newly introduced learning rates. During an iteration, we decrease the individual learning rates of the nodes according to a generalized harmonic progression [13]:

$$\lambda_k \leftarrow \frac{\lambda_k}{1 + \lambda_k} \quad (13)$$

In this way, one guarantees the strong monotonicity of λ_k and thus the convergence of the algorithm to an equilibrium point.

The learning rates of the nodes cannot be directly used for updating the poses since Eq. (7) requires a learning rate for each constraint and not for each node. When updating the network given the constraint $\langle j, i \rangle$, we obtain an average learning rate $\tilde{\lambda}_{ji}$ from the nodes on $\mathcal{P}_{ji}$ as

$$\tilde{\lambda}_{ji} = \frac{1}{|\mathcal{P}_{ji}|} \sum_{k \in \mathcal{P}_{ji}} \lambda_k. \quad (14)$$

Then, the constraint update turns into

$$\Delta \mathbf{x}_k \quad = \quad \tilde{\lambda}_{ji} |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}. \tag{15}$$

D. Scheduling the Network Optimization

When adding a set of constraints $\langle j, i \rangle \in \mathcal{C}_{\text{new}}$ to a network without performing an optimization, we can incrementally compute the error of the network as

$$e_{\text{new}} = \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji} + \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{new}}} r_{ji}^T \Omega_{ji} r_{ji}. \tag{16}$$

Here e_{new} is the new error and $\mathcal{C}_{\text{old}}$ refers to the set of constraints before the modification.

To avoid unnecessary computation, we perform the optimization only if needed. This is the case when the newly incorporated information introduced a significant error compared to the error of the network before. We perform an optimization step if

$$\frac{e_{\text{new}}}{|\mathcal{C}_{\text{new}}| + |\mathcal{C}_{\text{old}}|} > \alpha \max_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji}. \tag{17}$$

Here α is a user-defined factor that allows the designer of a mapping system to adapt the quality of the incremental solutions to the needs of the specific application.

If we assume that the network in $\mathcal{C}_{\text{old}}$ has already converged, this heuristic triggers an optimization only if a significant inconsistency is revealed. Furthermore, the optimization only needs to be performed for a subset of the network and not for the whole network. The subset is given by

$$\mathcal{E} \quad = \quad \bigcup_{\langle j,i \rangle \in \mathcal{C}_{\text{new}}} \mathcal{E}_{ji}. \tag{18}$$

Here $\mathcal{E}_{ji}$ is the set of constraints to be updated given a new constraint $\langle j, i \rangle \in \mathcal{C}_{\text{new}}$. The sets $\mathcal{E}_{ji}$ are computed according to Algorithm 1. This criterion satisfies Property 3 and leads to an efficient algorithm for incrementally optimizing the network of constraints.

V. Experiments

This section is designed to evaluate the effectiveness of the proposed methods to incrementally learn maximum likelihood maps. We first show that such a technique is well suited to generate accurate grid maps given laser range data and odometry from a real robot. Second, we provide simulation experiments to evaluate the evolution of the error and provide comparisons to our previously proposed techniques [3], [2], [6]. Finally, we illustrate the computational advantages resulting from our algorithm.

A. Real World Experiments

To illustrate that our technique can be used to learn maps from real robot data, we used the freely available ACES dataset. The motivating example shown in Figure 1 depicts four different maps computed online by our incremental mapping technique. During this experiment, we extracted constraints between consecutive poses by means of pairwise scan matching. Loop closures were determined by localizing the robot in the previously built map by means of a particle filter.

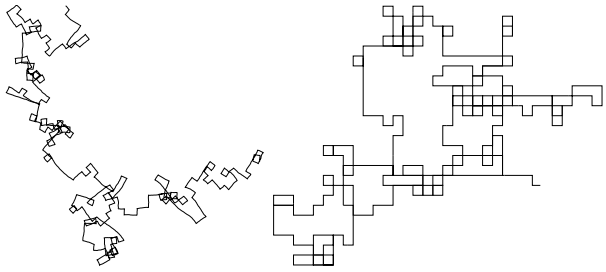


Fig. 2. Network used in the simulated experiments. Left: initial guess. Right: ground truth.

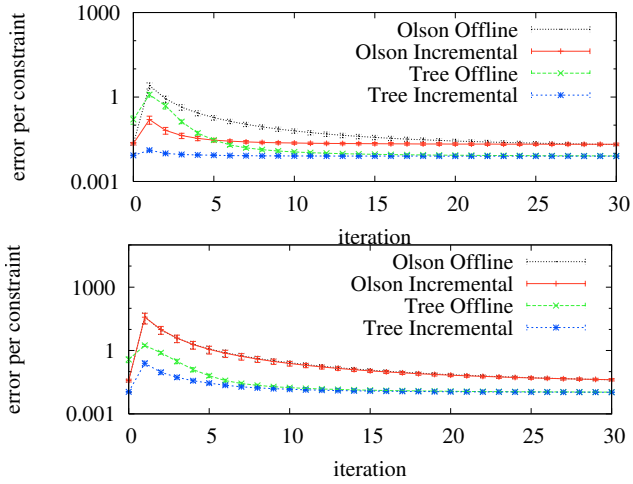


Fig. 3. Statistical experiments showing the evolution of the error per iteration of the algorithm. Top: situation in which the robot is closes a small loop. Bottom: closure of a large loop. The statistics have been generated by considering 10 different realizations of the observation noise along the same path.

As can be seen, our approach leads to accurate maps for real robot data. Similar results were obtained with all datasets we found online or recorded on our own.

B. Statistical Experiments on the Evolution of the Error

In the these experiments, we moved a virtual robot on a grid world. An observation is generated each time the current position of the robot was close to a previously visited location. The observations are corrupted by a given amount of Gaussian noise. The network used in this experiment is depicted in Figure 2.

We compare our approach named *Tree Incremental* with its offline variant [3] called *Tree Offline* which solves the overall problem from scratch. In addition to that, we compare it to the offline version without the tree optimization [2] called *Olson Offline* as well as its incremental variant [6] referred to as *Olson Incremental*. For space reasons, we omit comparisons to LU decomposition, EKF, and Gauss-Seidel. The advantages of our method over these other methods is similar to those previously reported [2].

To allow a fair comparison, we disabled the scheduling of the optimization of Eq. (17) and we performed 30 iterations every time 16 constraints were added to the network. During the very first iterations, the error of all approaches may show

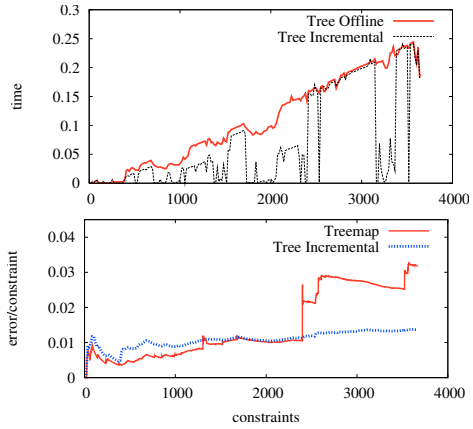


Fig. 4. Top: runtime comparison of the offline and the incremental approaches using a tree parameterization. The optimization is performed only when the error condition specified by Eq. (17) was verified. Bottom: Comparison of the evolution of the global error between Treemap[12] and the online version of our approach.

an increase, due to the bigger correction steps which result from increasing the learning rates.

Figure 3 depicts the evolution of the error for all four techniques during a mapping experiment. We depicted two situations. In the first one, the robot closed a small loop. As can be seen, the introduced error is small and thus our approach corrects the error within 2 iterations. Both incremental techniques perform better than their offline variants. The approach proposed in this paper outperforms the other techniques. The same holds for the second situation in which the robot was closing a large loop. Note that in most cases, one iteration of the incremental approach can be carried out faster, since only a subpart of the network needs to be updated.

C. Runtime Comparison

Finally, we evaluated our incremental version and its offline variant with respect to the execution time. Both methods where executed only when needed according to our criterion specified by Eq. (17). We measured the time needed to run the individual approach until convergence to the same low error configuration, or until a maximum number of iterations (30) was reached. As can be seen in Figure 4(top), the incremental technique requires significantly less operations and thus runtime to provide equivalent results in terms of error. Figure 4(bottom) shows the error plot of a comparison of our approach and Treemap [12] proposed by Frese. As shown in the error-plot, in the beginning Treemap performs slightly better than our algorithm, due to the exact calculation of the Jacobians. However, when closing large loops Treemap is more sensitive to angular wraparounds (see increase of the error at constraint 2400 in Figure 4). This issue is typically better handled by our iterative procedure. Overall, we observed that for datasets having a small noise Treemap provides slightly better estimates, while our approach is generally more robust to extreme conditions.

VI. Conclusion

In this paper, we presented an efficient online solution to the optimization of constraint networks. It can incrementally learn maps while the robot moves through the environment. Our approach optimizes a network of constraints that represents the spatial relations between the poses of the robot. It uses a tree-parameterization of the nodes and applies a variant of gradient descent to compute network configurations with low errors.

A per-node adaptive learning rate allows the robot to reuse already computed solutions from previous steps, to update only the parts of the network, which are affected by the newly incorporated information, and to start the optimization approach only if the new data causes inconsistencies with the already computed solution. We tested our approach on real robot data as well as with simulated datasets. We compared it to recently presented online and offline methods that also address the network-based SLAM problem. As we showed in practical experiments, our approach converges faster to a configuration with small errors.

Acknowledgment

The authors gratefully thank Udo Frese for providing us his Treemap implementation. This work has partly been supported by the DFG under contract number SFB/TR-8 (A3), by the EC under contract number FP6-IST-34120-muFly, and FP6-2005-IST-6-RAWSEEDS.

References

[1] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Journal of Autonomous Robots*, vol. 4, 1997.

[2] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006, pp. 2262–2269.

[3] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007. [Online]. Available: http://www.informatik.uni-freiburg.de/ stachnis/pdf/grisetti07rss.pdf

[4] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial realtionships in robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer Verlag, 1990, pp. 167–193.

[5] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[6] E. Olson, J. Leonard, and S. Teller, "Spatially-adaptive learning rates for online incremental slam," in *Robotics: Science and Systems*, Atlanta, GA, USA, 2007.

[7] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[8] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 1–12, 2005.

[9] M. Bosse, P. Newman, J. Leonard, and S. Teller, "An ALTAS framework for scalable mapping," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[10] C. Estrada, J. Neira, and J. Tardós, "Hierachical slam: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[11] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, Monterey, CA, USA, 1999, pp. 318–325.

[12] U. Frese, "Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping," *Journal of Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[13] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[C18] C. Stachniss, G. Grisetti, N. Roy, and W. Burgard. Evaluation of gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

Analyzing Gaussian Proposal Distributions for Mapping with Rao-Blackwellized Particle Filters

Cyrill Stachniss* Giorgio Grisetti* Wolfram Burgard* Nicholas Roy†

Abstract— **Particle filters are a frequently used filtering technique in the robotics community. They have been successfully applied to problems such as localization, mapping, or tracking. The particle filter framework allows the designer to freely choose the proposal distribution which is used to obtain the next generation of particles in estimating dynamical processes. This choice greatly influences the performance of the filter. Many approaches have achieved good performance through informed proposals which explicitly take into account the current observation. A popular approach is to approximate the desired proposal distribution by a Gaussian. This paper presents a statistical analysis of the quality of such Gaussian approximations. We also propose a way to obtain the optimal proposal in a non-parametric way and then identify the error introduced by the Gaussian approximation. Furthermore, we present an alternative sampling strategy that better deals with situations in which the target distribution is multi-modal. Experimental results indicate that our alternative sampling strategy leads to accurate maps more frequently that the Gaussian approach while requiring only minimal additional computational overhead.**

I. Introduction

Particle filters are a frequently used technique in robotics for dynamical system estimation. They have been used to localize robots [4], to build both feature-maps [12], [13] and grid-maps [7], [8], [9], and to track objects based on vision data [10]. A particle filter approximates the posterior by a set of random samples and updates it in a recursive way. The particle filter framework specifies how to update the sample set but leaves open how to choose the so-called proposal distribution. The proposal is used to draw the next generation of samples at the subsequent time step in the dynamical process. For example, in the context of localizing a robot, the odometry motion model is a good choice for the proposal in that it can be easily sampled and then easily transformed into the target distribution by such techniques as weighted importance sampling. In practice, the design of the proposal has a major influence on the performance and robustness of the filtering process. On the one hand, the closer the proposal is to the target distribution, the better is the estimation performance of the filter. On the other hand, the computational complexity of the calculation of the proposal distribution should be small in order to run the filter online. For this reason, the majority of particle filter applications restrict the proposal distribution to a Gaussian since one can efficiently draw samples from such a distribution.

Murphy, Doucet, and colleagues [6], [14] introduced factored particle filters, known as "Rao-Blackwellization", as an effective means to solve the simultaneous localization and mapping (SLAM) problem. By applying this factorization, several efficient mapping algorithms have been presented [7], [8], [9], [12] and we can note that all of these algorithms have used Gaussians to obtain the next generation of particles.

In this paper, we analyze how well such Gaussian proposal distributions approximate the optimal proposal in the context of mapping. We apply well-founded statistical measures to carry out the comparisons. To the best of our knowledge, this question has not been addressed in the context of particle filter applications in robotics so far. It turns out that Gaussians are often an appropriate choice but there exist situations in which multi-modal distributions are needed to appropriately sample the next generation of particles. Based on this insight, we present an alternative sampling technique that has the same complexity as the Gaussian approximation but can appropriately capture distributions with multiple modes, resulting in more robust mapping systems.

This paper is organized as follows. After a discussion of related approaches, we briefly introduce in Section III the ideas of mapping with Rao-Blackwellized filters. In Section IV, we explain how to actually represent and sample from the optimal proposal. We then present an efficient variant that allows us to deal with multi-modal proposals in an efficient way. In Section VI, we introduce the statistical tests that are used in the experimental section for evaluation.

II. Related Work

Particle filters have been applied to various kinds of robotic state estimation problems such as localization [4], mapping [7], [8], [9], [12], visual tracking [10], or data association problems [20]. Murphy, Doucet, and colleagues were the first that presented an approach based on a Rao-Blackwellized particle filter that learns grid maps [6], [14]. The first efficient approach for mapping with Rao-Blackwellized particle filters was the FastSLAM algorithm by Montemerlo *et al.* [13]. It uses a set of Kalman filters to represent the map features conditioned on a sampled robot pose. A Gaussian process model is used to sample the odometry motion model and generate the proposal distribution on the next step. The grid-based variant presented by Haehnel *et al.* [9] performs scan-matching as a preprocessing step. In this way, they are able to draw samples from Gaussians with lower variances compared to proposals computed based on the odometry only. This reduces the number of required particles and allows a robot to maintain a map estimate online. In contrast to that, Eliazar *et al.* [7] focus on an efficient grid map representation which allows the particles

*University of Freiburg, Department of Computer Science, D-79110 Freiburg. †MIT, 77 Massachusetts Ave., Cambridge, MA 02139-4307

to share a map. Subsequently, Montemerlo *et al.* published FastSLAM2 [12] that uses an informed proposal based on the most recent sensor observation to restrict the space for sampling. Again, to efficiently draw the next generation of particles, the distribution is assumed to be Gaussian. Grisetti *et al.* [8] extended FastSLAM2 to deal with large-scale occupancy grid maps. This technique combines scan-matching on a per particle basis with informed Gaussian proposal distributions.

To the best of our knowledge, there exists no evaluation of how well the Gaussian proposal distributions approximate the optimal proposal which in general is non-Gaussian in the context of mapping. There exist approaches that show that the uncertainty of certain SLAM techniques monotonically decreases over time. For example, Newman proved this property for the relative map filter and also showed that "in the limit, as the number of observations increases, the relative map becomes perfectly known" [15]. In the context of particle filters for SLAM, Montemerlo *et al.* [12] showed that FastSLAM2 "converges [...] for a restricted class of linear Gaussian problems". It, however, makes no statement about the validity of Gaussian approximations in real world settings.

III. Learning Maps with Rao-Blackwellized Particle Filters

A particle filter requires three sequential steps to update its estimate. Firstly, one draws the next generation of samples from the so-called proposal distribution π. Secondly, one assigns a weight to each sample. The weights account for the fact that the proposal distribution is in general not equal to the target distribution. The third step is the resampling step in which the target distribution is obtained from the weighted proposal by drawing particles according to their weight.

In the context of the SLAM problem, one aims to estimate the trajectory of the robot as well as a map of the environment. The key idea of a Rao-Blackwellized particle filter for SLAM is to separate the estimate of the trajectory $x_{1:t}$ of the robot from the map m of the environment. This is done by the following factorization

$$\begin{aligned} &p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = \\ &\quad p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}), \end{aligned} \tag{1}$$

where $z_{1:t}$ is the observation sequence and $u_{1:t-1}$ the odometry information. In practice, the first term of Eq. (1) is estimated using a particle filter and the second term turns into "mapping with known poses".

One of the main challenges in particle filtering is to choose an appropriate proposal distribution. The closer the proposal is to the true target distribution, the more precise is the estimate represented by the sample set. Typically, one requires the proposal π to fulfill the assumption

$$\begin{aligned} \pi(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \;=\; &\pi(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t-1}) \\ &\cdot \pi(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-2}). \end{aligned} \tag{2}$$

According to Doucet [5], the distribution

$$\begin{aligned} &p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \\ &\qquad \frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \end{aligned} \tag{3}$$

is the optimal proposal for particle i with respect to the *variance of the particle weights* that satisfies Eq. (2). This proposal minimizes the degeneracy of the algorithm (Proposition 4 in [5]). As a result, the computation of the weights turn into

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{\eta p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} \tag{4}$$

$$\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} \tag{5}$$

$$= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) \tag{6}$$

$$= w_{t-1}^{(i)} \cdot \int p(z_t \mid x') p(x' \mid x_{t-1}^{(i)}, u_{t-1}) \, dx'. \tag{7}$$

Unfortunately, the optimal proposal distribution is in general not available in closed form or in a suitable form for efficient sampling. As a result, most efficient mapping techniques use a Gaussian approximation of the optimal proposal. This approximation is easy to compute and allows the robot to sample efficiently. As we will show in this paper, the Gaussian assumption is not always justified. To provide examples for this statement, we first compute the optimal proposal explicitly and then compare it to the Gaussian approximation. Using the optimal proposal in a mapping system leads to computationally expensive operations which are explained in the next section in more detail.

IV. Computing and Sampling from the Optimal Proposal

This section explains how to compute the optimal proposal and how to sample from that distribution. In mapping as well as in many other problems, there is no closed form solution available but we can arrive at a high-fidelity numerical solution for the likelihood function. In our case, the numerator of Eq. (3) is the product of the observation likelihood and the odometry motion model. When using laser range finders, the dominating factor is the observation likelihood. To point-wise evaluate the observation likelihood, we use the so called "beam endpoint model" [19]. In this model, the individual beams within a scan are considered to be independent. Furthermore, the likelihood of a beam is computed based on the distance between the endpoint of the beam and the closest obstacle from that point. Using this point-wise evaluation of the observation likelihood, we can compute a three-dimensional histogram providing the observation likelihood for the different poses.

The second term in Eq. (3) is the robot motion model. In this paper, we consider the "banana-shaped" distribution known from most approaches to Monte-Carlo localization [4]. The likelihood for the individual poses is computed

point-wise and is stored in a histogram. This histogram describes the likelihood function in a non-parametric form. Histograms, however, are affected by discretization errors. To smooth this effect, we furthermore apply the Parzen window/kernel estimator [1] based on the evaluated data points. Let x^j be the evaluated poses, then this estimator is defined as

$$\hat{p}(x) \quad = \quad \frac{p(x^j)}{h} \sum_{j=1}^{n} K\left(\frac{x - x^j}{h}\right) \tag{8}$$

where h is called Parzen window. We chose the kernel $K(u)$ as

$$K(u) \quad = \quad \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right). \tag{9}$$

This technique allows us to smooth the histogram data and in this way avoid the discontinuities which are inherent in the histogram representation itself. Furthermore, we can make the likelihood of the smoothed histogram arbitrarily close to the optimal distribution of Eq. (3) by increasing the resolution of the local grid map and reducing the size of the histogram bins.

Given this non-parametric estimator, we can perform rejection sampling to draw the next generation of particles. Obviously, this results in a highly inefficient mapping system with respect to the computation time. However, it allows us to sample from an arbitrarily close approximation to the optimal proposal distribution and to compare it to its Gaussian approximation.

As we will illustrate in the experiments, in most cases the proposal can be safely approximated by a Gaussian. This explains why existing methods based on this particular approximation have been so successful. In certain situations, however, the distribution is highly non-Gaussian and often multi-modal so that the Gaussian does not properly approximate the true distribution which in turn can lead to the divergence of the filter. To overcome this problem, we present an alternative sampling method in the following section. This sampling strategy is able to handle multiple modes in the likelihood functions used as the proposal distribution. Note that our approach does not require any significant computational overhead compared to existing mapping systems that apply scan-matching in combination with a Gaussian proposal [8].

V. Efficient Mapping with Multi-Modal Proposal Distributions

In this section, we present our alternative sampling strategy that can handle multiple modes in the distributions while at the same time keeping the efficiency of a Gaussian proposal distribution. Our approach is equivalent to computing a sum of weighted Gaussians to model the proposal but does not require the explicit computation of a sum of Gaussians. Note that an open source implementation of our mapping system using this technique is available online [18].

Our previous method [8] first applies scan-matching on a per-particle basis. It then computes a Gaussian proposal *for each sample* by evaluating poses around the pose reported by the scan-matcher. This technique yields accurate results in case of a uni-modal distribution, but encounters problems in that it focuses only on the dominant mode to which the scan-matching process converges. The left image in Figure 1 illustrates an example in which the scan-matching process converges to the dominant peak denoted as "mode 1". As a result, the Gaussian proposal samples only from this mode and at most a few particles cover "mode 2" (and only if the modes are spatially close). Even if such situations are rarely encountered in practice, we found in our experiments that they are one of the major reasons for filter divergence.

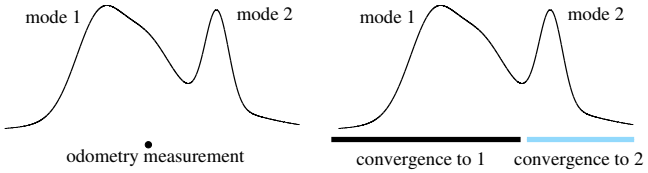


Fig. 1. The left image illustrates a 1D likelihood function and an odometry measurement. Conventional informed sampling first performs scan-matching starting from the odometry measurement. In this situation, the scan-matcher will find a local peak in the likelihood function (most likely mode 1) and the future sample will be drawn from a Gaussian centered at this single mode. The right image illustrates the new approach. It draws the sample first from the odometry model and applies scan-matching afterwards. When a drawn sample falls into the area colored black, the scan-matcher will converge to mode 1, otherwise, it will converge to mode 2. By sampling first from the odometry, then applying scan-matching, and finally computing local Gaussian approximations, multiple modes in the likelihood function are likely to be covered by the overall sample set.

One of the key ideas of our approach is to adapt the scan-matching/sampling procedure to better deal with multiple modes. It consists of a two step sampling. First, only the odometry motion model is used to propagate the samples. This technique is known from standard Monte-Carlo localization approaches (c.f. [4]) and allows the particles to cover possible movements of the robot. In a second step, gradient descent scan-matching is applied based on the observation likelihood and the denominator of Eq. (3). As a result, each sample converges to the mode in the likelihood function that is closest to its own starting position. Since the individual particles start from different locations, they are likely to cover the different modes in their corresponding likelihood functions as illustrated in the right image of Figure 1. Our approach leads to sample sets distributed according to a Gaussian *around the modes* in the observation likelihood functions. As we will demonstrate in the experimental results, this technique leads to proposal distributions which are closer to the optimal proposal given in Eq. (3) than the Gaussian approximations; when the distribution has only a single mode, the solution is equivalent to previous approaches [8].

VI. Statistical Tests

To analyze how close the Gaussian proposal as well as our new proposal are to the optimal proposal distribution, we make use of three statistical measures. First, we apply the Anderson-Darling test on normality [2]. This test is reported to be one of the most powerful tests in statistics for detecting most departures from normality. This test is

superior to the Kolmogorov-Smirnov test and has a similar performance than the Shapiro-Wilk test [16]. Second, we use the Kullback-Leibler divergence [11] to measure the distance between distributions. Third, we make use of a measure taken from the Cramér-von-Mises test [3], [21] to identify differences between distribution.

Given a set of n samples $\{y^1 < \ldots < y^n\}$ in ascending order of magnitude, the Anderson-Darling (AD) test computes the A statistic as

$$A = -n - \sum_{k=1}^{n} \frac{2k-1}{n} \left[\ln F(y^k) + \ln(1 - F(y^{n+1-k})) \right], \quad (10)$$

where F is the cumulated density function (CDF) of the distribution that is assumed to have generated the samples. In our case, F is the CDF of the normal distribution.

To determine if the samples are generated by a Gaussian or not, one needs to test if

$$A \cdot \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2} \right) \quad \leq \quad c, \quad (11)$$

where c is the Anderson-Darling test value for normal distributions corresponding to a desired level of significance. For example, for a 95% confidence test of normality, the corresponding c is 0.752.

This test allows us to check if the optimal proposal is in fact a Gaussian distribution. An interesting property of the AD test is that it also provides a confidence level for its result. To apply this test, we only need to draw a sample set from the optimal proposal and compute Eq. (10) and Eq. (11). Performing this test for all proposals generated during a mapping experiment provides a measure of how often a sample set is generated from a wrong distribution.

Besides the Anderson-Darling test, we apply the Kullback-Leibler divergence (KLD) which is a frequently used technique to measure the distance between two arbitrary distributions. This allows us to also compare our proposal given in the previous section to the optimal proposal distribution. A KLD value of zero indicates that the distributions are equal and the higher the KLD, the bigger is the difference between them. The KLD between p and f is defined as

$$KLD(p, f) \quad = \quad \int p(x) \cdot \log \left(\frac{p(x)}{f(x)} \right) dx. \quad (12)$$

The KLD takes into account a quotient between two distributions. This can give a high weight to differences in the tails of the distributions (see Eq. (12), where $f(x)$ is small).

An alternative measure for comparison is used in the Cramér-von-Mises test [3], [21]. It measures the disparity of two distributions by taking into account their cumulative density functions (CDF). Since it does not use a quotient as the KLD does, it gives less weight to the tails of the distribution. It computes the integral over the squared distances between the CDFs. Let p and f be the distributions to compare and P and F the corresponding CDFs. Then,

$$d(p, f) \quad = \quad \int [P(x) - F(x)]^2 \, dP(x) \quad (13)$$

provides a measure about the similarity of both distributions which is zero if both are equal.

The three techniques presented here are used in our experiments to identify the differences between the individual proposals and to illustrate potential weaknesses of the Gaussian proposals.

TABLE I

PROPOSAL DISTRIBUTIONS WHICH ARE REGARDED AS GAUSSIANS ACCORDING TO THE ANDERSON-DARLING TEST (95% CONFIDENCE).

Dataset	Gaussian proposal	Non-Gauss (unimodal)	Multi-modal proposal
Intel Research Lab	89.2%	7.2%	3.6%
FHW Museum	84.5%	10.4%	5.1%
Belgioioso	84.0%	10.4%	5.6%
MIT CSAIL	78.1%	15.9%	6.0%
MIT Killian Court	75.1%	19.1%	5.8%
Freiburg Bldg. 79	74.0%	19.4%	6.6%

VII. EXPERIMENTS

The experiments presented in this paper are all based on real world data. We furthermore used freely available datasets to perform our analysis. The learned maps and the datasets used here are available online [17].

A. Quality of Gaussian Proposals

In the first experiment, we carried out the Anderson-Darling (AD) test with a confidence of 95% to determine if the optimal proposal can be considered as Gaussian. The results of the test are described in Table I. As can be seen, depending on the dataset, in the optimal proposal was non-Gaussian in 10% to 26% of all cases.

By visually inspecting the datasets and resulting maps, we observed two different scenarios in which non-Gaussian situations occurred. Firstly, we often observed non-Gaussian observation likelihood functions in highly cluttered environments where small changes in the position led to substantial changes of the likelihood. Multi-modal distributions are likely to occur and Gaussians are not well suited to serve as a proposal in these cases. Secondly, non-Gaussian proposals occurred when the robot was moving in environments with long corridors, a fact that surprised us. At first sight, this may appear counterintuitive since corridors are well-structured environments. However, in positions where the robot cannot observe the end of the corridor with its sensor, the likelihood along the main axis of the corridor is almost constant which is highly non-Gaussian and can lead to a negative result of the AD test. One example is MIT Killian Court, consisting mainly of long corridors. Note that even if the AD test fails in such situations, Gaussians can be still good proposals.

In addition to testing acceptance as a Gaussian distribution, we analyzed the distance between the optimal proposal and its Gaussian approximation based on the KLD and the measure from the Cramér-von-Mises test (which is referred to as CvM in the remainder of this paper). Figure 2 plots the frequencies of the individual KLD and CvM values for the Intel and FHW datasets. As can be seen, the approximation error was small (values close to zero) in 94% to 97% of

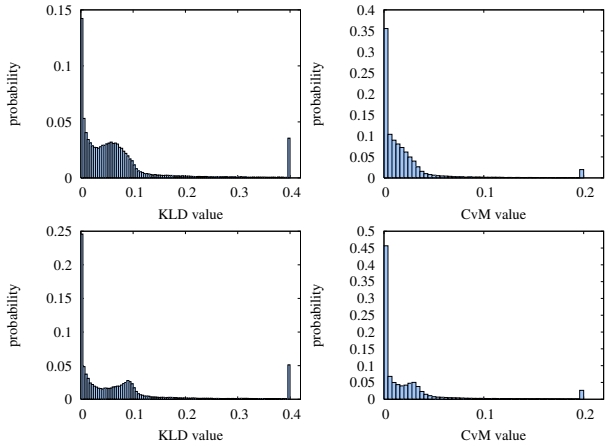


Fig. 2. Difference between the optimal proposal and the Gaussian approximation based on the Intel Research Lab (first row) and the FHW dataset (second row). The images on the left depict the frequencies of the individual Kullback-Leibler divergence values and the images on the right show the frequencies of the distance measure based on the Cramér-von-Mises test (see Eq. (13)). The right-most bin contains also all values larger or equal 0.4 (KLD) and 0.2 (CvM).

all cases. In all other cases, however, the distributions were substantially different. This fact is represented by the peak in the right-most bin of the histograms which contains all values larger or equal than 0.4 (KLD) and 0.2 (CvM). This peak corresponds to situations with multi-modal distributions which can only be badly approximated by a Gaussian. Note that similar results were obtained for the other datasets (see first row of Figure 3).

B. Multi-Modal Proposal Distribution

In the next experiment, we evaluated the alternative sampling strategy proposed in this paper. We used the KLD to compare our new proposal to the optimal proposal distribution. To actually perform the comparison, we computed all modes of the distribution explicitly, which is not required in the mapping system itself as described in Section V. To do so, we drew a set of samples and performed a gradient ascent in the likelihood function to find the individual modes. The modes were then approximated by Gaussians according to the sampled points.

The results of the comparison are shown in Figure 3 for different datasets. The plots in the first row show the KLD distance between the optimal proposal and its Gaussian approximation. The plots in the second row depict the corresponding comparison of our new proposal to the optimal one.

As can be seen, we obtained distributions that no longer approximated a significant fraction of the proposal distributions with large error (i.e., the right-most bin of the distance histograms). In contrast to this, the Gaussian approach approximates the optimal proposal inappropriately in 3% to 6% of all cases. The comparisons using the CvM value showed similar results and are omitted due to reasons of space.

Approaches using the Gaussian proposal have shown to build highly accurate maps of most datasets (compare the experiments in [8]) but there exist situations in which such

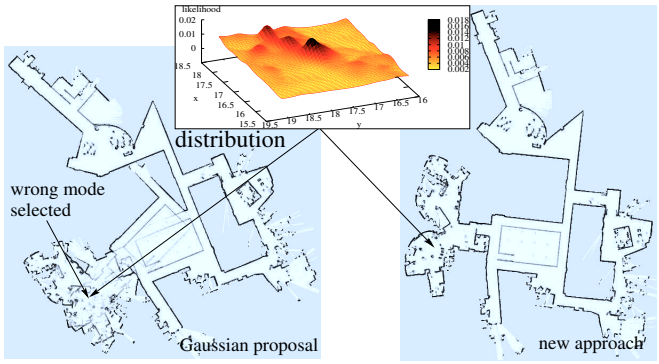


Fig. 4. Resulting map of the MIT CSAIL dataset using a Gaussian proposal (left) and our new approach (right). The Gaussian approach fails due to highly non-Gaussian likelihood functions in the cluttered room (illustrated for a given orientation θ in the top image). Trajectory length: 385m, recording time: 7 min, average speed: 0.9m/s.

TABLE II
EXECUTION TIME ON A 2.8 GHZ PC WITH A P4 SINGLE CORE CPU.

Dataset	N	Execution time		
		optimal	[8]	new method
MIT Killian Court	80	155 h	112 min	113 min
Freiburg Bldg. 79	30	84 h	62 min	62 min
Intel Research Lab	30	40 h	29 min	29 min
FHW Museum	30	38 h	27 min	27 min
Belgioioso	30	18 h	13 min	13 min
MIT CSAIL	30	10 h	7 min	7 min

techniques are likely to fail. This is especially the case if the dominant mode in the likelihood function is not the correct one. Such a situation occurs, for example, in the CSAIL dataset [17] recorded at MIT. Our expectation is that modeling multiple modes in the proposal distribution leads to more robust filters. We carried out 10 experiments with different random seeds and evaluated the success rate of the approach using the Gaussian proposal and our new method. Using the Gaussian approximation for the proposal distribution, the final map had the correct topology (all loops closed, etc.) in only 20% of trials whereas our new approach generated a correct map every time. Figure 4 shows example maps using the Gaussian proposal (left) and our new approach (right).

C. Runtime

In principle, it is possible to avoid Gaussian approximations in the proposal distribution. The main disadvantage when sampling from the optimal proposal is the high computational overhead. To illustrate this overhead, Table II shows the execution time for the individual approaches as well as the number of samples used (N). As can be seen, sampling from the optimal proposal is not suitable for practical applications since it took up to one week to correct a single dataset. In contrast to this, the computational overhead of our new approach is negligible. It allows a robot to learn an accurate map online while moving through the environment.

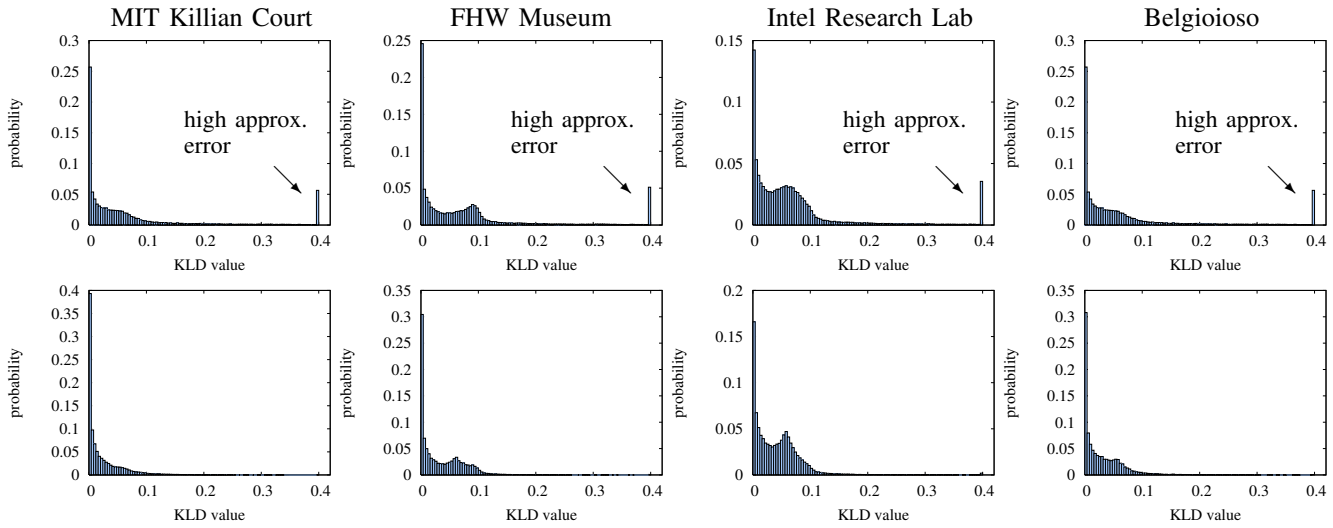


Fig. 3. The plots in the first row show the KLD between optimal proposal and its Gaussian approximation for different datasets. The plots in the second row depict the corresponding KLD between the optimal proposal and the proposal proposed in this paper. The right-most bin contains also all values larger or equal to 0.4. The right-most bin illustrates the mayor drawback of the Gaussian approximation since it described the situations in which the optimal proposal is highly non-Gaussian (e.g., multi-modal). Our new approach, however, can better deal with such situations.

VIII. Conclusion

In this paper, we analyzed how well Gaussian proposal distributions approximate the optimal proposal in the context of the application of Rao-Blackwellized particle filters to the simultaneous localization and mapping problem. We demonstrated that in around 5% of all cases, the Gaussian approximation is not sufficient to model the likelihood function. As such situations are one of the sources for the divergence of the filter, we presented an alternative sampling technique that is able to deal with multi-modal distributions while maintaining the same efficiency as the Gaussian proposal. This resulted in a more robust approach to mapping with Rao-Blackwellized particle filters. In experiments carried out with real data, we showed the efficiency and robustness of our approach.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8, by the EC under contract number FP6-IST-34120-muFly (action line: 2.5.2.: micro/nano based subsystems) and FP6-2005-IST-6-RAWSEEDS, and by the NSF under CAREER grant 0546467. Thanks to Dirk Hähnel for providing the Intel and the Belgioioso dataset as well as to Mike Bosse for the Killian Court dataset.

References

[1] E. Alaydin. *Introduction to Machine Learning*, chapter Nonparametric Density Estimation, pages 157–161. MIT Press, 2004.

[2] T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness-of-fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952.

[3] H. Cramér. On the composition of elementary errors. ii: statistical applications. *Skandinavisk Aktuarietidskrift*, 11:141–180, 1928.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[5] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processung Group, Dept. of Engeneering, University of Cambridge, 1998.

[6] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[7] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, 2003.

[8] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[9] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[10] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. of the European Conference on Computer Vision*, pages 343–356, 1996.

[11] S. Kullback and R. A Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[12] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2002.

[14] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[15] P.M. Newman. *On the structure and solution of the simultaneous localization and mapping problem*. PhD thesis, University of Sydney, Australia, 1999.

[16] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1965.

[17] C. Stachniss. Robotic datasets. http://www.informatik.uni-freiburg.de/∼stachnis/datasets, 2007.

[18] C. Stachniss and G. Grisetti. GMapping project at OpenSLAM.org. http://openslam.org, 2007.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*, chapter Robot Perception, pages 171–172. MIT Press, 2005.

[20] G.D. Tipaldi, A. Farinelli, L. Iocchi, and D. Nardi. Heterogeneous feature state estimation with rao-blackwellized particle filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[21] R. von Mises. *Wahrscheinlichkeitsrechnung und Ihre Anwendung in der Statistik und Theoretischen Physik*. Deuticke, Leipzig, Germany, 1931. In German.

[C19] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

Efficient Estimation of Accurate Maximum Likelihood Maps in 3D

Giorgio Grisetti Slawomir Grzonka Cyrill Stachniss Patrick Pfaff Wolfram Burgard

Abstract— **Learning maps is one of the fundamental tasks of mobile robots. In the past, numerous efficient approaches to map learning have been proposed. Most of them, however, assume that the robot lives on a plane. In this paper, we consider the problem of learning maps with mobile robots that operate in non-flat environments and apply maximum likelihood techniques to solve the graph-based SLAM problem. Due to the non-commutativity of the rotational angles in 3D, major problems arise when applying approaches designed for the two-dimensional world. The non-commutativity introduces serious difficulties when distributing a rotational error over a sequence of poses. In this paper, we present an efficient solution to the SLAM problem that is able to distribute a rotational error over a sequence of nodes. Our approach applies a variant of gradient descent to solve the error minimization problem. We implemented our technique and tested it on large simulated and real world datasets. We furthermore compared our approach to solving the problem by LU-decomposition. As the experiments illustrate, our technique converges significantly faster to an accurate map with low error and is able to correct maps with bigger noise than existing methods.**

I. Introduction

Learning maps has been a major research focus in the robotics community over the last decades and is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. In this paper, we consider the popular and so-called "graph-based" or "network-based" formulation of the SLAM problem in which the poses of the robot are modeled by nodes in a graph. Constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes. The goal of algorithms to solve this problem is to find a configuration of the nodes that maximizes the observation likelihood encoded in the constraints.

In the past, this concept has been successfully applied [3], [4], [7], [8], [9], [10], [12], [13], [15]. Such solutions apply an iterative error minimization techniques. They correct either all poses simultaneously [7], [9], [10], [15] or perform local updates [3], [4], [8], [13]. Most approaches have been designed for the two-dimensional space where the robot is assumed to operate on a plane [3], [4], [7], [10], [13]. Among all these approaches, multi-level relaxation [4] or Olson's algorithm [13] belong to the most efficient ones.

In the three-dimensional space, however, distributing an error between different nodes of a network is not straightforward. One reason for that is the non-commutativity of the three rotational angles. As a result, most approaches that provide good results in 2D are not directly applicable in 3D. One way is to ignore the non-commutativity of the rotational angles. In this case, however, the algorithm works only in case of small noise and in small environments. A few maximum likelihood mapping techniques have been proposed for the three-dimensional space [9], [12], [15]. Some approaches ignore the error in pitch and roll [9] whereas others detect loops and divide the error by the number of poses along the loop (weighted with path length, as in [12]). An alternative solution is to apply variants of the approach of Lu and Milios [10] and to correct the whole network at once [15].

All authors are members of the University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany

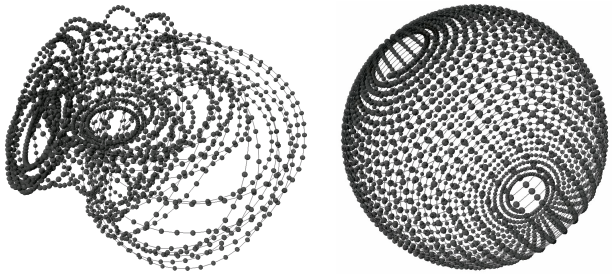


Fig. 1. A simulated trajectory of a robot moving on the surface of a sphere. The left image shows an uncorrected trajectory and the right image depicts the corrected one (approx. 8,600 constraints, 100 iterations, 21s).

The contribution of this paper is a technique to efficiently distribute the error over a sequence of nodes in all six dimensions (x, y, z, and the three rotational angles ϕ, θ, ψ). This enables us to apply a variant of gradient descent in order to reduce the error in the network. As a result, our approach converges by orders of magnitudes faster than the approaches mentioned above to low error configurations. As a motivating example, consider Figure 1. It depicts a trajectory of a simulated robot moving on the surface of a sphere. The left image depicts the input data and the right one the result of the technique presented in this paper.

The remainder of this paper is organized as follows. After discussing related work, we explain in Section III the graph-based formulation of the mapping problem as well as the key ideas of gradient descent in Section IV. Section V explains why the standard 2D approach cannot be used in 3D and introduces our technique to correct the poses given a network of constraints. Section VI analyzes the complexity of our approach. We finally present our experimental results in Section VII.

II. Related Work

A popular approach to find maximum likelihood (ML) maps is to apply least square error minimization techniques based on a network of relations. In this paper, we also

follow this way of describing the SLAM problem. Lu and Milios [10] first applied this approach in robotics to address the SLAM problem using a kind of brute force method. Their approach seeks to optimize the whole network at once. Gutmann and Konolige [7] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Howard *et al.* [8] apply relaxation to localize the robot and to build a map. Duckett *et al.* [3] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of constraints. In order to make the problem linear, they assume knowledge about the orientation of the robot. Frese *et al.* [4] propose a variant called multi-level relaxation (MLR). It applies relaxation based on different resolutions. Recently, Olson *et al.* [13] presented a novel method for correction two-dimensional networks using (stochastic) gradient descent. Olson's algorithm and MLR are currently the most efficient techniques available in 2D. All techniques discussed so far have been presented as solutions to the SLAM problem in the two-dimensional space. As we will illustrate in this paper, they typically fail to correct a network in 3D.

Dellaert proposed a smoothing method called square root smoothing and mapping [2]. It applies smoothing to correct the poses of the robot and feature locations. It is one of the few techniques that can be applied in 2D as well as in 3D. A technique that combines 2D pose estimates with 3D data has been proposed by Howard *et al.* [9] to build maps of urban environments. They avoid the problem of distributing the error in all three dimensions by correcting only the orientation in the x, y-plane of the vehicle. The roll and pitch is assumed to be measured accurately enough using an IMU.

In the context of three-dimensional maximum likelihood mapping, only a few approaches have been presented so far [11], [12], [15]. The approach of Nüchter *et al.* [12] describes a mobile robot that builds accurate three-dimensional models. In their approach, loop closing is achieved by uniformly distributing the error resulting from odometry over the poses in a loop. This technique provides good estimates but typically requires a small error in the roll and pitch estimate. Newman *et al.* [11] presented a sophisticated approach for detecting loop closures using laser and vision. Such an approach can be used to find the constraints which are the input to our algorithm.

Recently, Triebel *et al.* [15] described an approach that aims to globally correct the poses given the network of constraints in all three dimensions. At each iteration the problem is linearized and solved using LU decomposition. This yields accurate results for small and medium size networks especially when the error in the rotational component is small. We use this approach as a benchmark for our technique presented in this paper.

The contribution of this paper is a highly efficient technique to compute maximum likelihood maps in 3D. We present a way of distributing an error in all three rotational angles that accounts for the non-commutativity of these angles. This technique in combination with a variant of gradient descent allows us to correct larger networks than most state-of-the-art approaches.

III. On Graph-based SLAM

The goal of graph-based maximum-likelihood mapping algorithms is to find the configuration of nodes that maximizes the likelihood of the observations. For a more precise formulation consider the following definitions:

- $\mathbf{x}$ is a vector of parameters $(x_1 \ \cdots \ x_n)^T$ which describes a configuration of the nodes.
- δ_{ji} represents a constraint between the nodes i and j based on measurements. These constraints are the edges in the graph structure.
- Ω_{ji} is the information matrix capturing the uncertainty of δ_{ji}.
- $f_{ji}(\mathbf{x})$ is a function that computes a zero noise observation according to the current configuration of nodes. It returns an observation of node j from node i.

Given a constraint between node i and node j, we can define the error e_{ji} introduced by the constraint and residual r_{ji} as

$$e_{ji}(\mathbf{x}) = f_{ji}(\mathbf{x}) - \delta_{ji} = -r_{ji}(\mathbf{x}). \tag{1}$$

At the equilibrium point, e_{ji} is equal to 0 since $f_{ji}(\mathbf{x}) = \delta_{ji}$. In this case, an observation perfectly matches the current configuration of the nodes. Assuming a Gaussian observation error, the negative log likelihood of an observation f_{ji} is

$$F_{ji}(\mathbf{x}) = \frac{1}{2} \left(f_{ji}(\mathbf{x}) - \delta_{ji}\right)^T \Omega_{ji} \left(f_{ji}(\mathbf{x}) - \delta_{ji}\right) \tag{2}$$

$$\propto r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{3}$$

Under the assumption that the observations are independent, the overall negative log likelihood of a configuration $\mathbf{x}$ is

$$F(\mathbf{x}) = \frac{1}{2} \sum_{<j,i> \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}) \tag{4}$$

Here $\mathcal{C} = \{< j_1, i_1 >, \ldots, < j_M, i_M >\}$ is set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists.

A maximum likelihood map learning approach seeks to find the configuration $\mathbf{x}^*$ of the nodes that maximizes the likelihood of the observations which is equivalent to minimizing the negative log likelihood written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{5}$$

IV. Gradient Descent for Maximum Likelihood Mapping

Gradient descent (GD) is an iterative technique to find the minimum of a function. Olson *et al.* [13] were the first who applied it in the context of the SLAM problem in the two-dimensional space. GD seeks for a solution of Eq. (5) by iteratively selecting a constraint $< j, i >$ and by moving a set of nodes of the network in order to decrease the error introduced by the selected constraint. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \underbrace{\lambda \cdot J_{ji}^T \Omega_{ji} r_{ji}}_{\Delta \mathbf{x}} \tag{6}$$

Here $\mathbf{x}$ is the set of variables describing the locations of the poses in the network. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, and r_{ji} is the residual.

Reading the term Δx of Eq. (6) from right to left gives an intuition about the iterative procedure used in GD:

- r_{ji} is the residual which is the opposite of the error vector. Changing the network configuration in the direction of the residual r_{ji} will decrease the error e_{ji}.
- Ω_{ji} represents the information matrix of a constraint. Multiplying it with r_{ji} scales the residual components according to the information encoded in the constraint.
- J_{ji}^T: The role of the Jacobian is to map the residual term into a set of variations in the parameter space.
- λ is the learning rate which decreases with the iteration of GD and which makes the system to converge to an equilibrium point.

In practice, GD decomposes the overall problem into many smaller problems by optimizing the constraints individually. The difference between GD and stochastic GD is that the stochastic variant selects the constraints in a random order. Obviously, updating the different constraints one after each other can have opposite effects on a subset of variables. To avoid infinitive oscillations, one uses the learning rate to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different subproblems to asymptotically converge towards an equilibrium point that is the solution found by the algorithm. This equilibrium point is then reported as the maximum liklihood solution to the mapping problem.

V. 3D Graph Optimization

The graph-based formulation of the SLAM problem does not specify how the poses are presented in the nodes of the graph. In theory, one can choose an arbitrary parameterization. Our algorithm uses a tree based parameterization for describing the configuration of the nodes in the graph. To obtain such a tree from an arbitrary graph, one can compute a spanning tree. The root of the spanning tree is the node at the origin p_0. Another possibility is to construct a graph based on the trajectory of the robot in case this is available. In this setting, we build our parameterization tree as follows:

1) We assign a unique id to each node based on the timestamps and process the nodes accordingly.
2) The first node is the root of the tree.
3) As the parent of a node, we choose the node with the smallest id for which a constraint to the current node exists.

This tree can be easily constructed on the fly.

In the following, we describe how to use this tree to define the parameterization of the nodes in the network. Each node i in the tree is related to a pose p_i in the network and maintains a parameter x_i which is a 6D vector that describes its configuration. Note that the parameter x_i can be different from the pose p_i. In our approach, the parameter x_i is chosen as the relative movement from the parent of the node i in the tree to the node i itself

$$x_i = p_i \ominus p_{\text{parent}(i)}, \tag{7}$$

with $x_0 = p_0$. The operator $\ominus$ is the motion decomposition operator in 3D which is analogous to the one defined in 2D (see Lu and Milios [10]). A detailed discussion on tree parameterizations in combination with GD is out of the scope of this document and we refer the reader to [6].

Fig. 2. A simple example that illustrates the problem of distributing the error in 3D. The left image shows the input data which was obtained by moving a simulated robot over a hexagon twice with small Gaussian noise. The middle image show the result obtained if the non-commutativity of the rotation angles is ignored. The right images shows the result of our approach which is very close to the ground truth.

Before presenting our approach for correcting the poses in a network, we want to illustrate the problem of distributing an error over a sequence of nodes. Consider that we need to distribute an error e over a sequence of n nodes. In the two-dimensional space, this can be done in a straightforward manner as follows. Given the residual $r^{2D} = (r_x, r_y, r_\theta)$, we can simply change the pose of the i-th node in the chain by i/n times r^{2D}. This error propagation works well in 2D and is performed in most maximum-likelihood methods in the 2D space. In the three-dimensional space, however, such a technique is not applicable (with exception of very small errors). The reason for that is the non-commutativity of the three rotations

$$R(\phi, \theta, \psi) \quad \neq \quad \prod_1^n R(\frac{\phi}{n}, \frac{\theta}{n}, \frac{\psi}{n}), \tag{8}$$

where $R(\phi, \theta, \psi)$ is the three-dimensional rotation matrix. As illustrated in Figure 2, applying such an error propagation leads to divergence even for small and simple problems. Therefore, one has to find a different way of distributing the error over a chain of poses which is described in the following.

A. The Error Introduced by a Constraint

Let P_i be the homogenous transformation matrix corresponding to the pose p_i of the node i and X_i the transformation matrix corresponding to the parameter x_i. Let $\mathcal{P}_{i,0}$ be the ordered list of nodes describing a path in the tree from the root (here referred to as node 0) to the node i. We can express the pose of a node as

$$P_i = \prod_{k \in \mathcal{P}_{i,0}} X_k. \tag{9}$$

The homogenous transformation matrix X_i consists of a rotational matrix R and a translational component t. It has

the following form

$$X_i = \begin{pmatrix} R_k & t_k \\ 0 & 1 \end{pmatrix} \text{ with } X_i^{-1} = \begin{pmatrix} R_k^T & -R_k^T t_k \\ 0 & 1 \end{pmatrix} \tag{10}$$

In order to compute the transformation between two nodes i and j, one needs to consider the path $\mathcal{P}_{ji}$ from node i to node j. Since the nodes are arranged in a tree, this path consists of an ascending part and a descending part. Let $\mathcal{P}^a_{ji}$ be the ascending part of the path starting from node i and $\mathcal{P}^d_{ji}$ the descending part to node j. We can then compute the error e_{ji} in the reference frame of p_i as

$$e_{ji} = (p_j \ominus p_i) \ominus \delta_{ji}. \tag{11}$$

Using the matrix notation, the error is

$$E_{ji} = \Delta_{ji}^{-1} P_i^{-1} P_j \tag{12}$$

$$= \Delta_{ji}^{-1} \prod_{k^d \in \mathcal{P}^d_{ji}} X_{k^d}^{-1} \cdot \prod_{k^a \in \mathcal{P}^a_{ji}} X_{k^a}, \tag{13}$$

where Δ_{ji} is the matrix corresponding to δ_{ji}.

So far, we described the prerequisites for applying GD to correct the poses of a network. The goal of the update rule in GD is to iteratively update the configuration of a set of nodes in order to reduce the error introduced by a constraint. In Eq. (6), the term $J_{ji}^T \Omega_{ji}$ maps the variation of the error to a variation in the parameter space. This mapping is a linear function. As a result, the error might increase when applying GD in case of non-linear error surfaces. In the three-dimensional space, the rotational components often lead to highly non-linear error surfaces. Therefore, GD as well as similar minimization techniques cannot be applied directly to *large* mapping problems.

In our approach, we therefore chose a slightly different update rule. To overcome the problem explained above, we allow the usage of non-linear functions to describe the variation. The goal of this function is to compute a transformation of the nodes along the path in the tree so that the error introduced by the corresponding constraint is reduced. In detail, we design this function in a way so that it computes a new configuration of the variables $x_k \in \mathcal{P}_{ji}$ so that it corrects only a fraction λ of the error, where λ is the learning rate. In our experiments, we observed that such an update typically leads to a smooth deformation of the nodes along the path when reducing the error. In our approach, this deformation is done in two steps. First, we update the rotational components R_k of the variables x_k and second, we update the translational components t_k.

B. Update of the Rotational Component

This section explains how to deform a path in order to reduce the error introduced by a constraint. Without loss of generality, we consider the origin of the path p_i to be in the origin of our reference system. The orientation of p_j (in the reference frame of p_i) can be computed by multiplying the rotational matrices along the path $\mathcal{P}_{ji}$. To increase the readability of the document, we refer to the rotational matrices along the path as $\mathcal{R}_k$ neglecting the indices (compare Eq. (13)). The orientation of p_j is described by

$$\mathcal{R}_1 \mathcal{R}_2 \dots \mathcal{R}_n = \mathcal{R}_{1:n}, \tag{14}$$

where n is the length of the path $\mathcal{P}_{ji}$.

Distributing a given error over a sequence of 3D rotations, can be described in the following way: we need to determine a set of increments in the intermediate rotations of the chain so that the orientation of the last node (here node j) is $\mathcal{R}_{1:n}B$ where B the matrix that rotates x_j to the desired orientation based on the error. Formulated in a mathematical way, we need to compute a set of rotations A_k so that

$$\mathcal{R}_{1:n} B = \prod_{k=1}^{n} \mathcal{R}_k A_k. \tag{15}$$

Once the matrices A_k are known, the new rotational matrices of the parameters x_k are updated by

$$\mathcal{R}_k \leftarrow \mathcal{R}_k A_k. \tag{16}$$

We can decompose the matrix B into a set of incremental rotations $B = B_{1:n}$. In our current implementation, we compute the individual matrices B_k by using the spherical linear interpolation (slerp) [1]. We can decompose B using the slerp function with a parameter $u \in [0, 1]$ with $slerp(B, 0) = I$ and $slerp(B, 1) = B$. According to this framework, we can compute the rotation B_k as

$$B_k = [slerp(B, u_{k-1})]^T \, slerp(B, u_k). \tag{17}$$

To determine the values u_{k-1} and u_k, we consider the eigenvalues of the covariances of the constraints connecting the nodes $k-1$ and k. This is an approximation which works well in case of roughly spherical covariances. Note that the eigenvalues need to be computed only once in the beginning and are then stored in the tree.

Using this decomposition of B leads to Eq. (15) in which B is replaced by $B_{1:n}$. This equation admits an infinitive number of solutions. However, we are only interested in solutions which can be combined incrementally. Informally speaking, this means when truncating the path from n to $n-1$ nodes, the solution of the truncated path should be part of the solution of the full path. Formally, we can express this property by the following system of equations:

$$\forall_{k=1}^{n} : \quad R_1 A_1 \dots R_k A_k = R_{1:k} B_{1:k} \tag{18}$$

Given this set of equations, the solution for the matrices A_k can be computed as

$$A_k = R_k^T (B_{1:k-1})^T R_k B_{1:k}. \tag{19}$$

This is an exact solution that is always defined since A_k, R_k, and B_k are rotation matrices. The proof of Eq. (19) is given in the Section IX at the end of this document. Based on Eq. (16) and Eq. (19), we have a closed form solution for updating the rotational matrices of the parameters x_k along the path $\mathcal{P}_{ji}$ from the node i to the node j.

Note that we also use the slerp function to compute the fraction of the rotational component of the residual that is introduced by λ (see Section V-A).

For simplicity of presentation, we showed how to distribute the rotational error while keeping the node i fixed. In our implementation, however, we fix the position of the so-called "top node" in the path which is the node that is closest to the root of the tree (smallest level in the tree). As a result, the update of a constraint has less side-effects on other constraints in the network. Fixing the top node instead of node i can be obtained by simply saving the pose of the top node before updating the path. After the update, one transforms all nodes along path in way that the top node maintains its previous pose. Furthermore, we used the matrix notation in this paper to formulate the error distribution since it provides a clearer formulation of the problem. In our implementation, however, we use quaternions for representing rotations because they are numerically more stable. In theory, however, both formulations are equivalent. An open source implementation is available [5].

C. Update of the Translational Component

Compared to the update of the rotational component described above, the update of the translational component can be done in a straightforward manner. In our mapping system, we distribute the translational error over the nodes along the path without changing the previously computed rotational component.

We distribute the translational error by linearly moving the individual nodes along the path by a fraction of the error. This fraction depends in the uncertainty of the individual constraints encoded in the corresponding covariance matrices. Equivalent to the case when updating the rotational component, these fractions is also scaled with the learning rate.

VI. Computational Complexity

A single iteration of our algorithm requires to distribute the error introduced by the individual constraints over a set of nodes. Therefore, the complexity is proportional to the number of constraints times the number of operations needed to distribute the error of a single constraint.

In the remainder of this section, we analyze the number of operations needed to distribute the error of a single constraint. Once the poses of the nodes involved in an update step are known, the operations described in Section V-B and V-C can be carried out in a time proportional to the number of nodes $|\mathcal{P}|$ along the path $\mathcal{P}$. Computing the poses of the nodes along a path requires to traverse the tree up to the root according to Eq. (9). A naive implementation requires repeated traversals of the tree up to the root. This, however, can be avoided by choosing an intelligent order in which to process the constraints.

Let the "top node" of a path be the node with the smallest level in the tree. In our current implementation, we sort the constraints according to level of the corresponding top node. This can be done as a preprocessing step. We can process the constraints according to this order. The advantage of this order is that a constraint never modifies a node that has a smaller level in the tree. By storing the pose for each node in the tree, we therefore do not have to traverse the tree up to the root anymore. It is sufficient to access the parent of the top node in order to compute the poses for all nodes along a path $\mathcal{P}$. As a result, updating a constraint requires a time proportional to $|\mathcal{P}|$ and the overall complexity per iteration turns into $\mathcal{O}(M \cdot \mathbb{E}(|\mathcal{P}|))$. Here M is the number of constraints, and $\mathbb{E}(|\mathcal{P}|)$ is the average path length. In all our experiments, we experienced that the average path length grows more or less logarithmically with the number of nodes in the graph. This explains the fast pose updates of our approach shown in the experimental section.

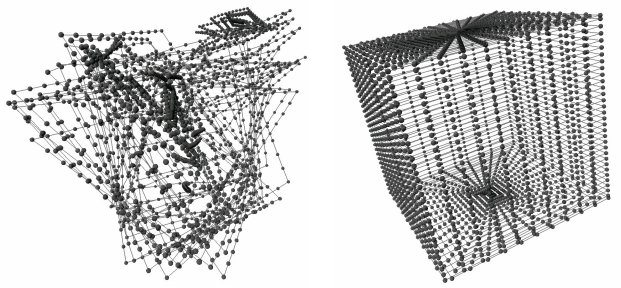


Fig. 3. A simulated trajectory of a robot moving on the surface of a cube. The left image shows an uncorrected trajectory and the right image depicts the corrected one (approx. 4,700 constraints, 100 iterations, 11s).

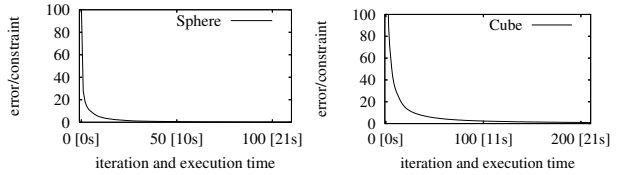


Fig. 4. The evolution of the error for the sphere and cube experiment.

VII. Experiments

The experiments are designed to show the properties of our technique. We first present results obtained in simulated experiments and then show results using real robot data.

A. Experiments with Simulated Data

In order to give the reader an intuition about the accuracy of our approach, we generated two datasets in which the virtual robot moved on the surfaces of easy to visualize geometric objects. In particular, we used a sphere and a cube. The nodes of the network as well as the constraints between the nodes were distorted with Gaussian noise. The left images of Figure 1 and Figure 3 depict the distorted input data whereas the images on the right illustrate the results obtained by our approach. As the figures indicate, the pose correction nicely recovers the original geometric structure.

To provide more quantitative results, Figure 4 depicts the evolution of the average error per link versus the iteration number as well as execution time for the sphere and the cube experiment. As can be seen, our approach converges to a configuration with small errors in less than 100 iterations.

We also applied the approach of Triebel *et al.* to both datasets. As mentioned above, this approach linearizes the

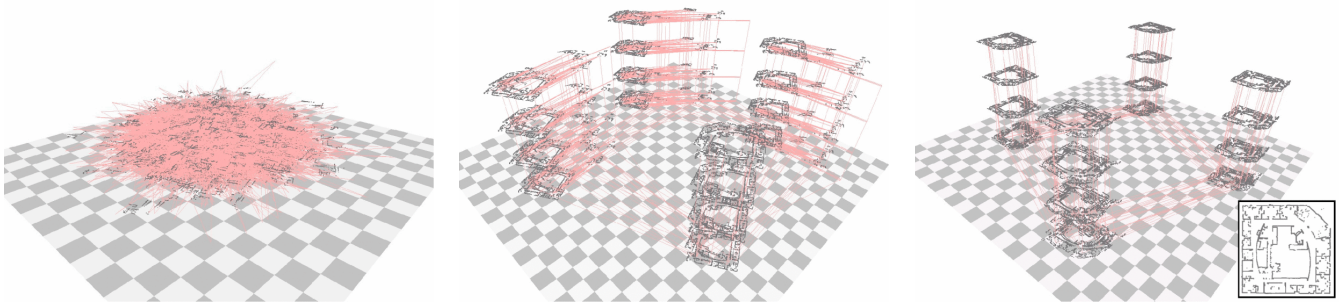


Fig. 5. The real world dataset of the Intel Research Lab recorded in 2D is used to generate a large 3D dataset. Each of the four virtual buildings consist of four identical floors. The left image depicts the starting configuration. The image in the middle depicts an intermediate result and the right one the corrected map after 50 iterations of our approach. We plotted in the images constraints between buildings and floors. For a better visibility, we furthermore plotted the constraints between individual nodes which introduce a high error and not all constraints. Constraints are plotted in light gray (red) and the laser data in black. The small image on the right shows a (corrected) map of the two-dimensional laser range data.

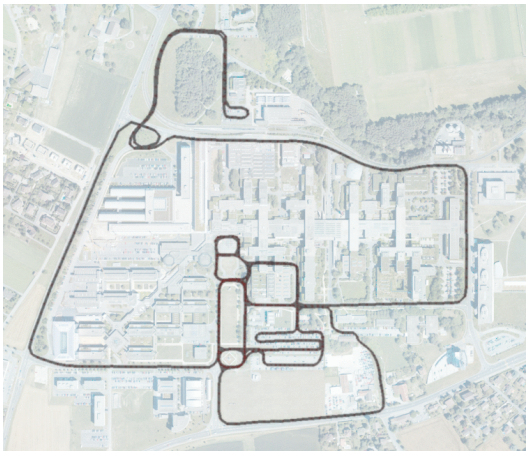


Fig. 6. The corrected trajectory plotted on top of an aerial image of the EPFL campus.

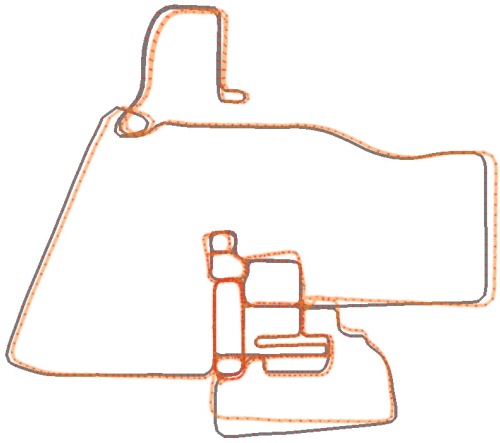


Fig. 7. The trajectory corrected by our approach is shown in black and the trajectory of the (D)GPS and IMU-based localization system is shown in orange/gray. By considering Figure 6 one can see that the black one covers the streets accurately.

problem and solves the resulting equation system using LU decomposition. Due to the comparably high noise in the simulated experiments, the linearization errors prevented this approach to find an appropriate configuration of the nodes.

B. Experiments with Partially Real Robot Data

The next experiment is obtained by extending data obtained from a 2D laser range finder into three dimensions. We used the 2D real world dataset of the Intel Research Lab in Seattle and constructed virtual buildings with multiple floors. The constraints between buildings and floors are manually added but all other data is real robot data. The dataset consists of 15.000 nodes and 72.000 constraints. We introduced a high error in the initial configuration of the poses in all dimensions. This initial configuration is shown in the left image of Figure 5. As can be seen, no structure is recognizable. When we apply our mapping approach, we get an accurate map of the environment. The image in the middle depicts an intermediate result and the right image show the resulting map after 50 iterations. To compute this result, it took around 3 minutes on a dual core Pentium 4 processor with 2.4 GHz.

C. Mapping with a Car-like Robot

Finally, we applied our method to a real world three-dimensional dataset. We used a Smart car equipped with 5 SICK laser range finders and various pose estimation sensors to record the data. The robot constructs local three-dimensional maps, so-called multi-level surface maps [15], and builds a network of constrains where each node represents such a local map. Constraints between the maps are obtained by matching the individual local maps.

We recorded a large-scale dataset at the EPFL campus in which the robot moved on a 10 km long trajectory. Figure 6 depicts an overlay of the corrected trajectory on an aerial image. As can be seen from the trajectory, several loops have been closed. Furthermore, it includes multiple levels such as an underground parking garage and a bridge with an underpass. The localization system of the car which is based on (D)GPS and IMU data is used to compute the incremental constraints. Additional constraints are obtained by matching local maps. This is achieved by first classifying cells of the local maps into different classes and then applying a variant of the ICP algorithm that considers these classes. More details on this matching can be found in our previous

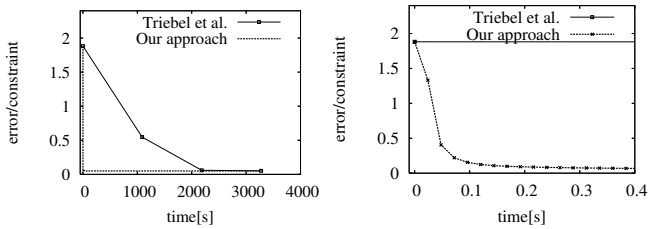


Fig. 8. The evolution of the average error per constraint of the approach of Triebel *et al.* [15] and our approach for the dataset recorded with the autonomous car. The right image shows a magnified view to the first 400 ms.

work [14]. Figure 7 plots the trajectory corrected by our approach and the one of the (D)GPS/IMU-based localization system.

We used the dataset from the EPFL campus to compare our new algorithm to the approach of Triebel *et al*. In this experiment, both approaches converge to more or less the same solution. The time needed to achieve this correction, however, is by orders of magnitudes smaller when applying our new technique. This fact is illustrated in Figure 8 which plots the average error per constraints versus the execution time required by both techniques.

We also applied our 3D optimizer to pure 2D problems and compared its performance to our 2D method [6]. Both techniques lead to similar results, the 2D version, however, is around 3 times faster that the 3D version. This results from the additional DOF in the state space.

VIII. Conclusion

In this paper, we presented a highly efficient solution to the problem of learning three-dimensional maximum likelihood maps for mobile robots. Our technique is based on the graph-formulation of the simultaneous localization and mapping problem and applies a variant of gradient descent to minimize the error in the network of relations.

Our method has been implemented and exhaustively tested in simulation experiments as well as with real robot data. We furthermore compared our method to a common existing approach to learn such models in the three-dimensional space. As shown in the experiments, our approach converges significantly faster and yields accurate maps with low errors.

IX. Appendix: Proof of Eq. (19)

We can proof by induction that the equation system in Eq. (18) always has a solution which is given by

$$A_k \quad = \quad R_k^T (B_{1:k-1})^T R_k B_{1:k} \qquad k = 1, \ldots, n. \tag{20}$$

- **Basis** $(n = 1)$:
  Based on Eq. (18) with $n = 1$ and by knowing that R_1 is a rotation matrix, a solution always exists and is given by
  $$A_1 = R_1^{-1} R_1 B_1 = B_1. \tag{21}$$
- **Inductive Step**:
  Assuming that Eq. (19) holds for $k = 1, \ldots, n-1$, we show that it holds also for $k = n$. We use Eq. (18) with $k = n - 1$ to substitute the term $R_1 A_1 \ldots R_{n-1} A_{n-1}$ in the equation for $k = n$. This leads to
  $$(R_{1:n-1} B_{1:n-1}) R_n A_n \quad = \quad R_{1:n-1} R_n B_{1:n}. \tag{22}$$
  By multiplying $(R_{1:n-1} B_{1:n-1} R_n)^{-1}$ from the left hand side, this turns into
  $$A_n \quad = \quad R_n^{-1} (B_{1:n-1})^{-1} (R_{1:n-1})^{-1} R_{1:n-1} R_n B_{1:n}$$
  Since R_k and B_k are rotation matrices, the inverse is always defined and given by the transposed matrix:
  $$A_n \quad = \quad R_n^T (B_{1:n-1})^T R_n B_{1:n} \qquad \text{q.e.d.} \tag{23}$$

Acknowledgment

This work has been supported by the DFG within the Research Training Group 1103 and under contract number SFB/TR-8 and by the EC under contract number FP6-2005-IST-5-muFly, FP6-2005-IST-6-RAWSEEDS, and FP6-004250-CoSy. Thanks to Udo Frese for his insightful comments and to Rudolph Triebel for providing his mapping system. Further thanks to Pierre Lamon for the joint effort in recording the EPFL dataset.

References

[1] T. Barrera, A. Hast, and E. Bengtsson. Incremental spherical linear interpolation. In *SIGRAD*, volume 13, pages 7–13, 2004.
[2] F. Dellaert. Square Root SAM. In *Proc. of Robotics: Science and Systems (RSS)*, pages 177–184, Cambridge, MA, USA, 2005.
[3] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287 – 300, 2002.
[4] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.
[5] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. TORO project at OpenSLAM.org. http://www.openslam.org/toro.html, 2007.
[6] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.
[7] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, CA, USA, 1999.
[8] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.
[9] A. Howard, D.F. Wolf, and G.S. Sukhatme. Towards 3d mapping in large urban environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 419–424, 2004.
[10] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.
[11] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2006.
[12] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.
[13] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.
[14] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, W. Burgard, and R. Siegwart. Towards mapping of cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.
[15] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[C20] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3d using attitude and noisy vision sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

Learning Maps in 3D using Attitude and Noisy Vision Sensors

Bastian Steder Giorgio Grisetti Slawomir Grzonka Cyrill Stachniss Axel Rottmann Wolfram Burgard

***Abstract*— In this paper, we address the problem of learning 3D maps of the environment using a cheap sensor setup which consists of two standard web cams and a low cost inertial measurement unit. This setup is designed for lightweight or flying robots. Our technique uses visual features extracted from the web cams and estimates the 3D location of the landmarks via stereo vision. Feature correspondences are estimated using a variant of the PROSAC algorithm. Our mapping technique constructs a graph of spatial constraints and applies an efficient gradient descent-based optimization approach to estimate the most likely map of the environment. Our approach has been evaluated in comparably large outdoor and indoor environments. We furthermore present experiments in which our technique is applied to build a map with a blimp.**

I. Introduction

In the last decades, the simultaneous localization and mapping (SLAM) problem has been an active field of research and effective solutions have been proposed. The majority of approaches is able to learn 2D maps of large-scale environments [13]. When moving from 2D to 3D map learning, the higher dimension of the search space prevents us to directly apply 2D algorithms to the 3D case. Different systems for building 3D maps have been proposed [5], [12], [14] but most of these approaches rely on bulky sensors having a high range and accuracy (e.g., SICK laser range finders) which cannot be used on small flying vehicles.

Cameras are an attractive alternative to laser range finders. Due to their limited weight and low power consumption, they can be incorporated into a wide class of devices. Existing approaches that address the vision-based SLAM problem mainly focus on scenarios in which a robot repeatedly observes a set of features [4], [11]. They have been shown to learn accurate feature maps of small-scale environments.

In this paper, we present a system that allows us to acquire elevation maps of large environments using two low quality web-cams and a low cost inertial measurement unit (IMU). Especially the cameras provide comparably low quality images which are affected by significant motion blur. Figure 1 illustrates this sensor setup.

Our approach integrates the data coming from the IMU and the cameras to obtain an estimate of the camera motion of the 3D position of the features extracted from the image data. We address the SLAM problem by constructing a graph of relations between poses. Each node in the graph represents a camera pose. An edge between two nodes is obtained from the sensor measurements and encodes the spatial constraints between two different camera poses. Our systems combines SURF features [2] with a PROSAC-based technique [3] to identify the correct correspondences between images. Loops are detected by matching features extracted from the images recorded from the different locations. The correction step is carried out using an optimization algorithm. The contribution of this paper is an approach that enables us to build highly accurate elevation maps of large environments using a comparably poor sensor setup. Our system is designed to work on lightweight flying vehicles.

All authors are members of the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

Fig. 1. Top Left: the sensors used for testing our approach. We assembled two cheap USB web-cams as a stereo pair and combined it with a XSens MTi inertial measurement unit. Bottom Left: a typical stereo image used for constructing the map. Note the significant motion blur affecting the image. Right: the procedure for acquiring the data. We mounted the sensors with the cameras looking downwards on a stick and we then walked around the campus.

II. Related Work

The effectiveness of vision-based approaches strongly depends on the feature extraction algorithms. To this end, SIFT features [10] represent a robust and popular option but they require significant computational resources. Compared to SIFT, SURF features [2] are significantly faster to compute while providing comparably stable feature descriptors. Therefore, we apply this technique in our work.

Jensfelt *et al.* [8] proposed an effective way of meeting the computational constraints imposed by online processing by combining a SIFT feature extractor and an interest points tracker. The interest points are obtained by using an Harris corner extractor. While the SIFT feature extraction can be performed at low frequency, the movement of the robot is constantly estimated by tracking the interest points at high frequency. Andreasson *et al.* [1] presented a technique that is based on a local similarity measure for images. They store reference images at different locations and use these references as a map. In this way, their approach is reported to scale well with the size of the environment.

Davison *et al.* [4] proposed a single camera SLAM algorithm. The system computes the map by means of a Kalman filter. A particle filter is applied to initialize the

3D landmarks. The particles estimate the depth information of the landmarks. The approach does not depend on an initial odometry estimate and is effective on small scale environments as well as in situations in which the robot repeatedly observes the same scene. However, it requires good quality images. Montiel *et al.* [11] extended this framework by proposing an inverse depth parameterization of the landmarks. Since this parameterization can be better approximated by a Gaussian, the use of the particle filter in the initial stage can be avoided.

Other approaches use a combination of inertial sensors and cameras. For example, Eustice *et. al* [5] rely on a combination of highly accurate gyroscopes, magnetometers, and pressure sensors to obtain a good estimate of orientation and altitude of an underwater vehicle. Based on these estimates, they construct an accurate global map using an information filter based on high resolution stereo images.

The work which is closest to our approach is a technique proposed by Jung *et al.* [9]. They use a high resolution stereo camera for building elevation maps with a blimp. The map consists of 3D landmarks extracted from interest points in the stereo image obtained by a Harris corner detector and the map is estimated using a Kalman filter. Due to the wide field of view and the high quality of the images the nonlinearities in the process were adequately solved by the Kalman filter. In contrast to this, our approach is able to deal with low resolution and low quality images. It is particularly suitable for mapping indoor environments and for being used on small size flying vehicles. We furthermore apply a more efficient error minimization approach [6].

III. Maximum Likelihood Elevation Map Estimation

The SLAM problem can be formulated as a graph: the nodes of the graph represent the poses of the robot along its trajectory and an edge between two nodes encodes the pairwise observations. Here, each node x_i of the graph represents a 6D camera pose. An edge between two nodes i and j is represented by the tuple $\langle \delta_{ji}, \Omega_{ji} \rangle$. δ_{ji} and Ω_{ji} are respectively the mean and the information matrix of a measurement made from the node i about the location of the node j expressed in the reference frame of the node i.

In our system, the information between two poses depends on the correspondence of the images acquired between the poses and on the IMU measurements. Once the graph is constructed, one has to compute the configuration of the nodes which best explains the observations. This results in deforming the robot trajectory based on the constraints to obtain a map.

Such a graph-based maximum likelihood SLAM approach requires to solve the following sub-problems:

- The construction of the graph based on the sensor input.
- The optimization of the graph so that the likelihood of the observations is maximized.

The first problem is addressed in this and the two subsequent sections. A solution to the second problem is then provided in Section VI.

Our approach relies on visual features extracted from the images obtained from two down-looking cameras. We use SURF features [2] instead of SIFT features [10] since they are significantly faster to compute while providing the same robustness. A SURF feature is rotation and scale invariant and is described by a descriptor vector and the position, orientation, and scale in the image.

In order to build consistent maps, we need to determine the camera position $(x\ y\ z\ \phi\ \theta\ \psi)^T$ given the features in the current image, a subset of spatially close features in the map, and the measurements obtained by the IMU.

The IMU provides the orientation of the system in terms of the Euler angles roll (ϕ), pitch (θ), and yaw (ψ). Due to the low quality IMU in combination with the presence of magnetic disturbances in indoor environments as well as on real robots, the heading information is highly affected by noise. In our experiments, we found that the roll and the pitch observations can directly be integrated into the estimate whereas the yaw information was too noisy to provide useful information. This reduces the dimensionality of each pose that needs to be estimated from $\mathbb{R}^6$ to $\mathbb{R}^4$.

Whenever a new image is acquired, a node x_{i+1} that models the new camera pose is added to the graph. The main challenge is to add the correct edges between x_{i+1} and other nodes $x_j, j \leq i$ of the graph. To do so, one has to solve the so-called data association problem. This means one has to determine which feature in the current image corresponds to which feature in the map. Let $S = \{s_1, \ldots, s_n\}$ refer to a local map of features that will be matched against the features $F = \{f_1, \ldots, f_m\}$ extracted from the current image. The result of such a matching is a transformation T which describes the spatial relations between the two sets of features. In the remainder of this section, we discuss how to compute the camera pose given the two sets S and F while the question of how to determine the set S is discussed in Section V.

IV. The Transformation Between Camera Poses

In this section, we describe how to compute the transformation of the camera based on the set of observed features F and the set of map features S.

Given the camera parameters, such a transformation can be determined by using two corresponding features in the two sets. This holds only since the attitude of the camera is known. In order to reduce the effects of outliers, we select the correspondences by using a consensus algorithm similar to PROSAC [3]. The main idea of PROSAC is to construct a prior for sampling the correspondences based on the distance of the descriptors. In this way, a smaller number of trials is required to find good candidate transformations than with the uninformed version of the RANSAC algorithm. We first determine the possible correspondences based on the feature descriptors. Subsequently, we select from this set the correspondences to compute the candidate transformations. We assign a score based on a fitness function to each candidate transformation and select the transformation with

the highest score. The next subsections explain our procedure in detail.

A. Potential Correspondences

For each feature f_i in the camera image and each feature s_j in the map, we compute the Euclidian distance $d^F(f_i, s_j)$ between their descriptor vectors. The distance is used to compute the set of potential correspondences $C = \{c_{ij}\}$ by considering only the feature pairs whose distance is below a given threshold D as

$$C = \{c_{ij} = \langle f_i, s_j \rangle \mid d^F(f_i, s_j) < D \wedge f_i \in F \wedge s_j \in S\}. \quad (1)$$

For simplicity of notation, we will refer to the elements of C as c, neglecting the indices i and j. The features in a correspondence can be retrieved by using the selector functions $\mathrm{f}(c)$ and $\mathrm{s}(c)$ so that

$$c = \langle f_i, s_j \rangle \leftrightarrow f_i = \mathrm{f}(c) \wedge s_j = \mathrm{s}(c). \quad (2)$$

A camera transformation is determined by two correspondences c_a and c_b. Accordingly, the number of possible transformations is proportional to $|C|^2$. We can limit the computational load of the approach by sorting the correspondences based on their distances and by considering only the best N correspondences, where $N = 250$ in our current system. Let C' be this reduced set. A candidate transformation T_{ab} is computed for each pair of correspondences $\langle c_a, c_b \rangle \in C' \times C'$.

We compute the transformation T_{ab} based on $\langle c_a, c_b \rangle$ as follows. Assuming the attitude and the internal parameters of the camera to be known, it is possible to project the segment connecting the two features on a plane parallel to the ground. The same is done with the two features in the map. The offset between the two camera poses along the z axis is determined using the pinhole camera model. Subsequently, the yaw between the images is computed as the angle between the two projections. Finally, x and y can be directly calculated by matching a pair of corresponding points in the translated image after applying the yaw correction.

B. Score

The previous step computes a set of candidate transformations $\{T_{ab}\}$. To select the best one, we need to rank them according to a score. The score of T_{ab} measures the quality of a matching by considering all potential correspondences between the feature sets that have not been used for determining T_{ab}. This set is given by

$$\tilde{C}_{ab} = C - \{c_a, c_b\}. \quad (3)$$

For each $c_k \in \tilde{C}_{ab}$, we project the associated features $\mathrm{f}(c_k)$ and $\mathrm{s}(c_k)$ in the image, according to T_{ab}. The score $\mathrm{v}(c_k)$ of the correspondence c_k is the following:

$$\mathrm{v}(c_k) = w \left(1 - \frac{d^I(\mathrm{f}(c_k), \mathrm{s}(c_k))}{\mathrm{d}^I_{max}}\right) + (1-w) d^F\left(\mathrm{f}(c_k), \mathrm{s}(c_k)\right). \quad (4)$$

Here w is a weighting factor, $d^I(\mathrm{f}(c_k), \mathrm{s}(c_k))$ is the distance between the features projected into the image space, d^I_{max} is the maximum value to accept as a match, and $d^F(\mathrm{f}(c_k), \mathrm{s}(c_k))$ is the distance between the feature descriptors. The overall score of the transformation T_{ab} is the sum of the individual scores of the correspondences in $\tilde{C}_{ab}$

$$\mathrm{score}(a, b) = \sum_{c_k \in \tilde{C}_{ab}} \mathrm{v}(c_k). \quad (5)$$

V. Extracting Constraints

The procedure described in the previous sections tells us how to compute the transformation of the camera given two sets of features. So far, we left open how the subsets of map features S is selected. In this section, we explain how to choose this subset to adequately keep track of the potential topologies of the environment. The selection of the subset of features in combination with the approach described in the previous section, defines the constraints represented by the edges in the graph.

While incrementally constructing a graph, one can distinguish three types of constraints: *visual odometry constraints*, *localization constraints*, and *loop closing constraints*. Visual odometry constraints are computed by considering the potential match between the features in the current image, and a limited subset of frames acquired from camera poses which are temporally close to the current one. Localization constraints occur when the camera is moving through an already visited region. In this case, the features in the current map are selected in a region around the pose estimate obtained from visual odometry. Finally, loop closing constraints model a spatial relation between the current frame and a region in the map which has been seen long time before. In our approach, we seek to find these different constraints in each step.

A. Visual Odometry

Each time a new image is acquired, we augment the graph with a new pose that represents the location of the most recent camera observation. This node is initialized according to the translation resulting from to the visual odometry.

The visual odometry estimate is obtained by first constructing the set S_o based on the features extracted from the last M frames and then extracting the best transformation according to Section IV. Let x_k be a node in the graph and let $S(x_k)$ be the set of features which have been observed by that node. If x_{i+1} is the current pose, we compute the set S_o for determining the visual odometry as

$$S_\mathrm{o} = \bigcup_{j=i-M}^{i} S(x_j). \quad (6)$$

An advantage of this procedure is that it in practice always finds a good incremental motion estimate. However, due to the error accumulation the estimate is affected by a drift which in general grows over time.

B. Localization

When the camera moves through known terrain, it is possible to determine the constraints by matching the current features with the ones in the map. This can be done by

localizing the robot in a region around the estimate provided by the visual odometry. This set of features is computed by considering all nodes in the graph that are close to the current node. Note that we ignore the features that are already used to compute the visual odometry. This procedure is effective for re-localizing the camera in a small region around the most recent position. The computational cost depends roughly on the area spanned by the search.

C. Loop Closing

As a third step, we seek for loop closures. In case the camera re-enters known terrain after having moved for a long time in an unknown region, the accumulated uncertainty can prevent the localization procedure for determining the right correspondences. Performing the localization procedure on the whole map is possible in theory. However, this operation is typically too expensive to be performed online.

Therefore, our algorithm reduces this cost by executing this search in two passes. At a first level only one feature in the current image is matched with all the features in the map, and the descriptors distances are computed. The reference feature is the one having the highest score when computing the visual odometry. Subsequently, a localization is performed around all features whose distance from the reference feature is below a given threshold. This is clearly a heuristic but it shows a robust matching behavior in real world situations. Note that it can happen that this approach does not find an existing correspondences but it is unlikely that this leads to a wrong constraint.

VI. Graph Optimization

Given a constraint between node i and node j, we can define the error e_{ji} introduced by the constraint as

$$e_{ji} = x_j - (x_i \oplus \delta_{ji}). \tag{7}$$

Here $\oplus$ represents the standard motion composition operator. At the equilibrium point, e_{ji} is equal to 0 since $x_j = x_i \oplus \delta_{ji}$.

In this case, an observation perfectly matches the current configuration of the nodes. Assuming a Gaussian observation error, the negative log-likelihood of an observation F_{ji} is

$$F_{ji}(\mathbf{x}) = \frac{1}{2} \left(x_j - (x_i \oplus \delta_{ji})\right)^T \Omega_{ji} \left(x_j - (x_i \oplus \delta_{ji})\right) \tag{8}$$

Under the assumption that the observations are independent, the overall negative log likelihood of a configuration $\mathbf{x}$ is

$$F(\mathbf{x}) = \frac{1}{2} \sum_{\langle j,i \rangle \in \mathcal{C}} e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}) \tag{9}$$

Here $\mathcal{C} = \{\langle j_1, i_1 \rangle, \ldots, \langle j_M, i_M \rangle\}$ is a set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists.

The goal of the optimization phase is to find the configuration $\mathbf{x}^*$ of the nodes that maximizes the likelihood of the observations. This can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{10}$$

To compute this quantity, we use a variant of the iterative 3D optimization approach presented by Grisetti *et al.* [6]. Since in our setting the yaw and the pitch of the camera are known from the IMU, we perform the search in the $(x\ y\ z\ \psi)$ space only.

During one iteration, the algorithm optimizes the individual constraints sequentially. It distributes the error e_{ji} introduced by the constraint over a set of nodes related to this constraint. Each time a constraint $\langle \delta_{ji}, \Omega_{ji} \rangle$ between the nodes i and j is optimized, we consider a path on the graph between the two nodes and modify the configuration of these nodes in order to reduce the error.

Let x_i and x_j be the poses of the nodes in the current configuration. We can compute the error between the two constraints in the global reference frame according to Eq. 7. Let $\mathcal{P}_{ji} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ be the a path in the graph connecting the nodes i and j. Given a node x_k, we consider the number $\mathrm{n}(k)$ of constraints affecting the update of the node. This number can be determined as:

$$\mathrm{n}(k) = \sum_{\langle j,i \rangle \in \mathcal{C}} \begin{cases} 1 & \text{if } x_k \in \mathcal{P}_{ji} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

In practice $\mathrm{n}(k)$ is the number of constraints whose paths constrain the node x_k. By assuming the Ω_{ji} to be spherical information matrices, this number represents an approximation of the stiffness of a node in the network.

We linearly distribute the error between the nodes in the path. Each of the nodes in the path will receive a contribution inversely proportional to its stiffness, according to the following rule:

$$\Delta x^{(k)} = -|\mathcal{P}_{ji}| \frac{\sum_{k=1}^{j-1} 1/\mathrm{n}(k)}{\sum_{k=1}^{N} 1/\mathrm{n}(k)} \cdot e_{ji}. \tag{12}$$

Here $|\mathcal{P}_{ji}|$ is the length of the path, $\mathrm{n}(k)$ is the stiffness of a node computed according to Eq. (11).

Updating a constraint, however, can increase the error introduced by other constraints. Therefore, we merge the effects of the individual updates according to a learning rate. The learning rate decreases each iteration. Accordingly, the fraction of the error used for updating a constraint decreases with each iteration. As a consequence, the modification of the overall network configuration introduced by the update will be smaller and the nodes of the graph will converge towards a common equilibrium point, close to a maximum likelihood configuration. More details can be found in [6], [7].

VII. Experiments

In this section, we present the experiments carried out to evaluate our approach. We used only real world data which we recorded with our sensor platform shown in Figure 1 as well as using a real blimp.

A. Outdoor Environments

In the first experiment, we measured the performance of our algorithm using data recorded in outdoor environments. For obtaining this dataset, we mounted our sensor platform on the tip of a rod to simulate a freely floating vehicle with the cameras pointing downwards (see Figure 1). We

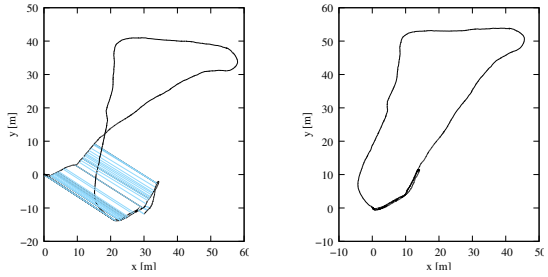


Fig. 2. The left image shows the path of the camera in black and the matching constraints in gray. The right image shows the corrected trajectory after applying the optimization technique.

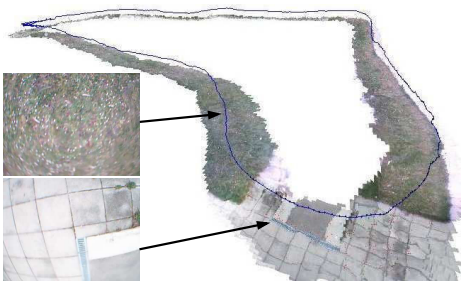


Fig. 3. Perspective view of the textured elevation map of the outdoor experiment together with two camera images recorded at the corresponding locations.

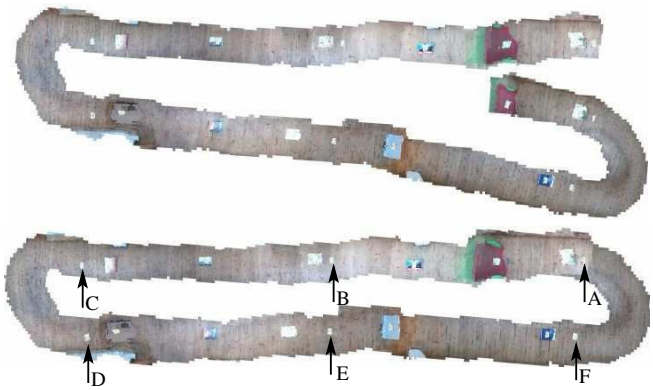


Fig. 4. Top view of the map in the indoor experiment. The top image show the map estimate based on the visual odometry (before the global correction) and the lower image depicts it after least square error minimization. The labels A to F present six landmarks for which we determined the ground truth location manually and which are used to evaluate the accuracy of our approach.

TABLE I

ACCURACY OF THE RELATIVE POSE ESTIMATE BETWEEN LANDMARKS

landmarks	A-B	B-C	C-D	D-E	E-F	F-A	loop
mean error [m]	0.19	0.27	0.1	0.23	0.2	0.13	1.11
sigma [m]	0.24	0.35	0.12	0.4	0.32	0.15	1.32
error [%]	4.2	6.1	8.1	5.7	4.5	8.6	5.2

walked on a long path around our building over different types of ground like grass and pavement. The real trajectory has a length of about 190 m (estimated via Google Earth). The final graph contains approximately 1400 nodes and 1600 constraints. The trajectory resulting from the visual odometry is illustrated in the left image of Figure 2. Our system autonomously extracted data association hypotheses and constructed the graph. These matching constraints are colored light blue/gray in the same image. After applying our optimization technique, we obtained a map in which the loop has correctly been closed. The corrected trajectory is shown in the right image of Figure 2. A perspective view of the textured elevation is depicted in Figure 3.

The length of the trajectory after correction was 208 m which is an overestimation of approximatively 9% (given the rough ground truth estimate obtained from Google Earth). Given that our low cost stereo system has an uncertainty of around 10 cm at an altitude of 1 m, this is in the bounds of a consistent map.

This experiment illustrates that our approach is able to build maps of comparably large environments and that it is able to find the correct correspondences between the observations. Note that this is done without real odometry information compared to wheeled robots. This is possible even if the camera images are blurry and mainly show grass and concrete surfaces.

B. Indoor Environments

The second experiment evaluates the performance of our approach quantitatively in an indoor environment. The data was acquired with the same sensor setup as in the previous experiment. We moved in the corridor of our building which has a wooden floor. For a better illustration, some objects were placed on the ground. Note that although the artificial objects on the ground act as reliable landmarks, they are not necessary for our algorithm as shown by the first experiment. Figure 4 depicts the result of the visual odometry (top image) and the final map after least square error minimization (lower image). We measured the location of six landmarks in the environment manually with a measurement tape (up to an accuracy of approx. 3 cm). The distance in the x coordinate between neighboring landmarks is 5 m and it is 1.5 m in the y direction. The six landmarks are labeled as A to F in the lower image. We used these six known locations to estimate the quality of our mapping technique. We repeated the experiment 10 times and measured the relative distance between them.

Table I summarizes this experiment. As can be seen, the error of the relative pose estimates is always below 10% compared to the true difference. The error results mainly from potential mismatches of features and from the error in our low quality stereo setup. Given this cheap setup, this is an accurate estimate for a system lacking sonar, laser range data, and real odometry information.

C. Experiment using a Blimp

The third experiment was performed using a real flying vehicle which is depicted in Figure 5. The problem with the blimp is its limited payload. Therefore, we were unable to mount the IMU and had only a single camera available which was pointing downwards. Since only one camera was

Fig. 5. The left image depicts our blimp and the right one example images received via the analog video transmission link.

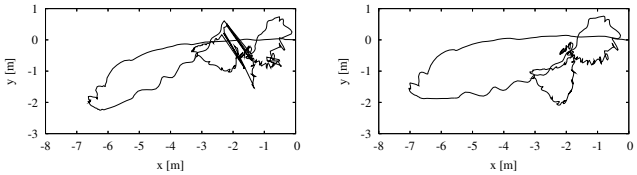


Fig. 6. The left image illustrates the trajectory recovered by our approach. Straight lines indicate that the robot re-localized in previously seen parts of the environment (loop closure). The small loops and the discontinuities in the trajectory result from assuming the attitude to be identically zero. In this way changes in tilt and roll were mapped by our algorithm in changes in x and y. The right image shows the trajectory obtained after applying our optimization algorithm.

available, the distance information estimated by the visual odometry can only be determined up to a scale factor. Furthermore, our system had no information about the attitude of the sensor platform due to the missing IMU. Therefore, we flew conservative maneuvers only and assumed that the blimp was flying parallel to the ground. The left image in Figure 5 shows our blimp in action.

The data from the camera was transmitted via an analog video link and all processing has been done off board. Interferences in the image frequently occurred due to the analog link as illustrated in the right image of Figure 5. In practice, such noise typically leads to outliers in the feature matching. The mapped environment is a factory floor of concrete that provides poor textures which makes it hard to distinguish the individual features.

Even under these hard conditions, our system worked satisfactory well. We obtained a comparably good visual odometry and could extract correspondences between the individual nodes on the graph. Figure 6 shows the uncorrected as well as the corrected graph from a top view.

D. Performance

All the experiments have been executed on a 1.8 GHz Pentium dual core laptop computer. The frame rate we typical obtain for computing the visual odometry and performing the local search for matching constraints is between 5 and 10 fps. We use an image resolution of 320 by 240 pixel and we typically obtain between 50 and 100 features per image. The exact value, however, depends on the quality of the images.

The time to carry out the global search for matching constraints increases linearly with the size of the map. In the first experiment presented in this paper, the frequency with which the global search for loop closures could be executed was 1 Hz.

VIII. Conclusions

In this paper, we presented a mapping system that is able to build consistent maps of the environment using a cheap vision-based sensor setup. Our approach integrates state-of-the-art techniques to extract features, to estimate correspondences between landmarks, and to perform least square error minimization. Our system is robust enough to handle low textured surfaces like large areas of concrete or lawn. We are furthermore able to deploy our system on a flying vehicle and to obtain consistent elevation maps of the ground.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8, within the Research Training Group 1103 and by the EC under contract number FP6-IST-34120-muFly, action line: 2.5.2.: micro/nano based subsystems, and FP6-004250-CoSy.

References

[1] H. Andreasson, T. Duckett, and A. Lilienthal. Mini-slam: Minimalistic visual slam in large-scale environments based on a new interpretation of image similarity. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, Graz, Austria, 2006.

[3] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, USA, 2005.

[4] A. Davison, I. Reid, , N. Molton, and O. Stasse. Monoslam:real time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[5] R.M. Eustice, H. Singh, J.J. Leonard, and M.R. Walter. Visually mapping the RMS Titanic: conservative covariance estimates for SLAM information filters. *Int. Journal of Robotics Research*, 25(12), 2006.

[6] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[7] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[8] P. Jensfelt, D. Kragic, and M. Folkesson, J. B jörkman. A framework for vision based bearing only 3d slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, CA, 2006.

[9] I. Jung and S. Lacroix. High resolution terrain mapping using low altitude stereo imagery. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Nice, France, 2003.

[10] D.G. Lowe. Distinctibe image features from scale invariant keypoints. *Int. Journal on Computer Vision*, 2004.

[11] J.M. Montiel, J. Civera, and A.J. Davison. Unified inverse depth parameterization for monocular slam. In *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, Massatchuttes, USA, 2006.

[12] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[13] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[14] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[C21] K.M. Wurm, C. Stachniss, G. Grisetti, and W. Burgard. Improved simultaneous localization and mapping using a dual representation of the environment. In *Proc. of the European Conference on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

Improved Simultaneous Localization and Mapping using a Dual Representation of the Environment

Kai M. Wurm Cyrill Stachniss Giorgio Grisetti Wolfram Burgard

University of Freiburg, Dept. of Computer Science, Georges-Köhler-Allee 79, 79110 Freiburg, Germany

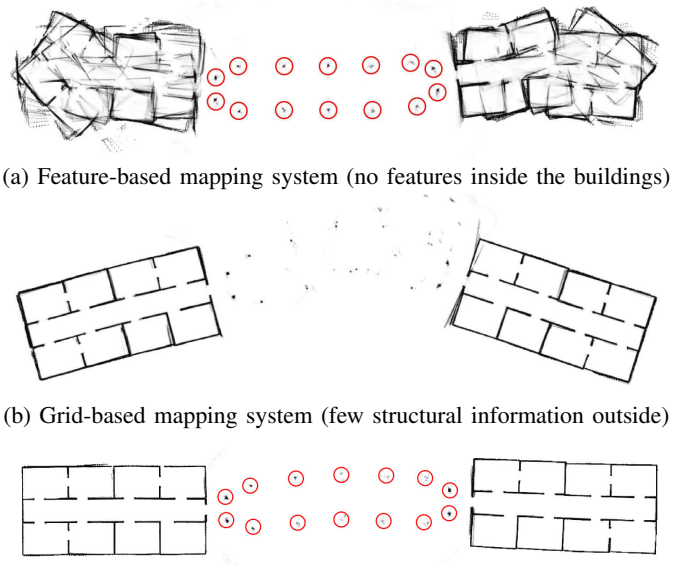
***Abstract*— The designer of a mapping system for mobile robots has to choose how to model the environment of the robot. Popular models are feature maps and grid maps. Depending on the structure of the environment, each representation has certain advantages. In this paper, we present an approach that maintains feature maps as well as grid maps of the environment. This allows a robot to update its pose and map estimate based on the representation that models the surrounding of the robot in the best way. The model selection procedure is obtained by reinforcement learning and takes a decision based on the current observation. As we will illustrate in simulation as well as in real world experiments, this allows a robot to learn accurate maps in a more robust way than approaches using only feature or only grid maps.**

I. Introduction

Building maps is one of the fundamental tasks of mobile robots. In the literature, the mobile robot mapping problem is often referred to as the *simultaneous localization and mapping (SLAM)* problem. It is considered to be a complex problem, because for localization a robot needs a consistent map and for acquiring a map a robot requires a good estimate of its location. This mutual dependency between the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space.

A large variety of different estimation techniques has been proposed. Extended Kalman filter, sparse extended information filters, maximum likelihood methods, particle filter, and several other techniques have been applied to estimate the pose of the robot as well as a map of the environment. Most approaches to mapping use sets of features to model the environment, grid maps, or topological maps. Each representation has its own advantages. The environment the robot is deployed in mainly influences the decision which model to chose. For example, in large open spaces with predefined landmarks, feature-based approaches are likely to outperform mapping techniques based on grid maps. In dense and cluttered environment, however, grids offer substantial advantages.

In our system, we maintain the joint posterior about the trajectory of the robot and the map of the environment using a Rao-Blackwellized particle filter. The contribution of this paper is a novel approach which combines feature-based models with occupancy grid maps. Our approach allows a robot to perform its corrections based on both representations. It selects the model that is currently the best one to map the surroundings of the robot. The model selection process is obtained using reinforcement learning. It makes a decision based on the current sensor observations and the state of the filter. As we will demonstrate in the experiments, our approach outperforms pure grid and pure feature-based approaches. A motivating example is shown in Figure 1.



(a) Feature-based mapping system (no features inside the buildings)

(b) Grid-based mapping system (few structural information outside)

(c) Combining features and grid maps

Fig. 1. When mapping environments that contain large open spaces with few landmarks as well as dense structures, a combination of feature maps and grids maps outperforms the individual techniques.

This paper is organized as follows. After a discussion of related work, we briefly introduce mapping with Rao-Blackwellized particle filters in Section III. Section IV presents our filter for mapping which maintains the dual model of the environment. Section V explains our model selection process based on reinforcement learning. Experiments carried out in simulation and on real robots are presented in Section VI.

II. Related Work

Mapping techniques for mobile robots can be roughly classified according to the map representation and the underlying estimation technique. One popular map representation is the occupancy grid. Whereas such grid-based approaches are computationally expensive and typically require a huge amount of memory, they are able to represent arbitrary objects. Feature-based representations are attractive because of their compactness. However, they rely on predefined feature extractors, which assumes that some structures in the environments are known in advance.

The model of the environment and the applied state estimation technique are often coupled. One of the most popular approaches are extended Kalman filters (EKFs) in combination

with landmarks. The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [10, 15]. Their weakness lies in the strong assumptions that have to be made on both the robot motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely identifiable. There exist techniques [14] to deal with unknown data association in the SLAM context, however, if these assumptions are violated, the filter is likely to diverge [5, 9, 19].

Thrun *et al.* [18] proposed a method that uses the inverse of the covariance matrix. The advantage of the sparse extended information filters (SEIFs) is that they make use of the approximative sparsity of the information matrix and in this way can perform predictions and updates in constant time. Eustice *et al.* [4] presented a technique to make use of exactly sparse information matrices in a delayed-state framework.

In a work by Murphy, Doucet, and colleagues [2, 13], Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a possible robot trajectory and a map. The framework has been subsequently extended by Montemerlo *et al.* [11, 12] for approaching the SLAM problem with landmark maps. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [3] and Hähnel *et al.* [7]. Whereas the first work describes an efficient map representation, the second presents an improved motion model that reduces the number of required particles. The work of Grisetti *et al.* [6] describes an improved variant of the algorithm proposed by Hähnel *et al.* [7] combined with the ideas of FastSLAM2 [11]. Instead of using a fixed proposal distribution, the algorithm computes an improved proposal distribution on a per-particle basis on the fly.

So far, there exist only very few methods that try to combine feature-based models with grid maps. One is the hybrid metric map (HYMM) approach [8]. It estimates the location of features and performs a triangulation between them. In this triangulation, a so called dense map is maintained which can be transformed according to the update of the corresponding landmarks. This allows the robot to obtain a dense map by using a feature-based mapping approach. However, it is still required that the robot is able to reliably extract landmarks.

III. Mapping with Rao-Blackwellized Particle Filters

According to Murphy [13], the key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the map m and the trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot. This estimation is performed given the observations $z_{1:t} = z_1, \ldots, z_t$ and the odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$ obtained by the mobile robot. The Rao-Blackwellized particle filter for SLAM makes use of the following factorization

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}). \quad (1)$$

This factorization allows us to first estimate only the trajectory of the robot and then to compute the map given that trajectory. Since the map strongly depends on the pose estimate of the robot, this approach offers an efficient computation. This technique is often referred to as Rao-Blackwellization.

Typically, Eq. (1) can be calculated efficiently since the posterior about maps $p(m \mid x_{1:t}, z_{1:t})$ can be computed analytically using "mapping with known poses" since $x_{1:t}$ and $z_{1:t}$ are known.

To estimate the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ about the potential trajectories, one can apply a particle filter. Each particle represents a potential trajectory of the robot. Furthermore, an individual map is associated with each sample. The maps are built from the observations and the trajectory represented by the corresponding particle.

This framework allows a robot to learn models of the environment and estimate its trajectory but it leaves open how the environment is represented. So far, this approach has been applied using feature-based models [11, 12] or grid maps [3, 6, 7, 13]. Each representation has its own advantages and one typically needs some prior information about the environment to select the appropriate model. In this paper, we combine both types of maps to represent the environment. This allows us to combine the advantages of both worlds. Depending on the most recent observation, the robot selects that model which is likely to be the best model in the current situation.

IV. Dual Model of the Environment

Our mapping system applies such a Rao-Blackwellized particle filter to maintain the joint posterior about the trajectory of the robot and the map of the environment. In contrast to previous algorithms, each particle carries a grid map as well as a map of features. The key idea is to maintain both representations simultaneously and to select in each step the model that is best suited to update the pose and map estimate of the robot. Our approach is independent of the actual features that are used. In our current system, we use a laser range finder and extract clusters of beam end points which are surrounded by free space. In this way, we obtain features from trees, street lamps, etc. Note that other feature detectors can be directly integrated into our approach. The detector itself is completely transparent to the algorithm.

In each step, our algorithm considers the current estimate as well as the current sensor and odometry observation to select either the grid or the feature model to perform the next update step. This decision affects the proposal distribution in the particle filter used for mapping. The proposal distribution is used to obtain the next generation of particles as well as to compute the importance weights of the samples.

In the remainder of this section, we first introduce the particularities of our particle filter. We then explain in the subsequent section how to actually select the model for the current step.

If the grid map is to be used, we draw the new particle poses from an improved proposal distribution as introduced by Grisetti *et al.* [6]. This proposal performs scan-matching on a per particle basis and then approximates the likelihood function by a Gaussian. This technique has been shown to yield accurate grid maps of the environment, given that there

is enough structure to perform scan-matching for an initial estimate.

When using feature maps, we apply the proposal distribution as done by Montemerlo *et al.* [12] in the FastSLAM algorithm.

After the proposal is used to obtain the next generation of samples, the importance weights are computed according to Grisetti *et al.* [6] and Montemerlo *et al.* [12] respectively. Note that we compute for each sample i two weights $w_g^{(i)}$ (based on the grid map) and $w_f^{(i)}$ (based on the feature map). For resampling, one weight is required but we need both values in our decision process as explained in the remainder of this paper.

To carrying out the resampling step, we apply the adaptive resampling strategy originally proposed by Doucet [1]. It computes the so-called effective sample size or effective number of particles (N_{eff}) to decide whether to resample or not. This is done based on the weights resulting from the proposal used to obtain this generation of samples.

V. Model Selection

The probably most important aspect of our proposed algorithm is to decide which representation to choose given the current sensor readings and the filter. In the following, we describe different strategies we investigated and which are evaluated in the experimental section of this paper.

A. Observation Likelihood Criterion

A mapping approach that relies on scan-matching is most likely to fail if laser readings cannot be aligned to the map generated so far. This is likely to be the case in large open space with sparse observations. In such a situation it is often better to use a pre-defined feature extractor (in case there are feature) to estimate the pose of the robot.

A measure that can be used to detect such a situation is the likelihood $l(z_t, x_t, m_{g,t})$ that the scan-matching seeks to maximize. To point-wise evaluate the observation likelihood of a laser observation, we use the so called "beam endpoint model" [17]. In this model, the individual beams within a scan are considered to be independent. The likelihood of a beam is computed based on the distance between the endpoint of the beam and the closest obstacle from that point.

Calculating the average likelihood for all particles results in a value that can be used as a heuristic to decide which map representation to use in a given situation:

$$\bar{l} \quad = \quad \frac{1}{N} \sum_i l(z_t, x_t^{(i)}, m_{g,t}^{(i)}) \tag{2}$$

A heuristic for selecting the feature-based representation instead of the grid map can be obtained based on a threshold ($\bar{l} \leq c_1$).

B. N_{eff} Criterion

As described above, each particle i carries two weights $w_g^{(i)}$ and $w_f^{(i)}$, one for the grid-map and one for the feature-map. These weights can be seen as an indicator of how well a particle explains the data and therefore can be used as a heuristic for model selection. Since the weights of a particle are based on different types of measurement, they cannot be compared directly. What can be compared, however, is the weight distribution over the filter.

One way to measure this difference in the individual weights is to compute the variance of the weights. Intuitively a set of weights with low variance does not strongly favor any of the hypothesis represented by the particles, while a high variance indicates that some hypotheses are more likely than others.

This suggests that a strategy based on the N_{eff} value, which is strongly related to the variance of the weights, can be a good heuristic. N_{eff} is computed for both sets of weights as

$$N_{\mathit{eff}}^g = \frac{1}{\sum_{i=1}^N (w_g^{(i)})^2} \quad \text{and} \quad N_{\mathit{eff}}^f = \frac{1}{\sum_{i=1}^N (w_f^{(i)})^2}. \tag{3}$$

It can be easily seen, that a higher variance in the weights yields a lower N_{eff} value. Assuming that a set of particles with a higher variance in the weights is usually more discriminative, it seems reasonable to switch to the feature-based model whenever $N_{\mathit{eff}}^f < N_{\mathit{eff}}^g$.

C. Reinforcement Learning for Model Selection

Both approaches described above are clearly heuristics. In this section, we describe how to use reinforcement learning to combine the heuristics while avoiding their pitfalls. The basic idea of reinforcement learning is to find a mapping from states S to actions A which maximizes a numerical reward signal r (see [16] for an introduction). Such a mapping is called a policy and can be learned by interacting with the environment. Inspired by the human learning method of trial and error, this class of learning algorithms perform a series of actions and analyze the obtained reward.

There exist a number of algorithms for reinforcement learning that differ most notably in the knowledge available about the environment. If it can be modeled as an Markov decision process for example, technics such as policy iteration can be applied. If no model of the environment is available, Monte Carlo methods or Temporal-Difference Learning (TD learning) should be applied. For our approach, we use the Sarsa algorithm [16] which is a popular algorithm among the TD methods and does not require a model of the environment. It learns an action-value function $Q(s, a)$ which assigns a value to state-action pairs. Those values can then be used to generate a policy (e.g., choose the action that has the highest value in a given state).

To apply this method to our model selection problem, we have to define the states S, the actions A, and the reward $r : S \rightarrow R$. Defining the actions is straight forward as $A = \{a_g, a_f\}$, where a_g defines the use of the grid map and a_f the use of the feature map.

The state set has to be defined in a way that it represents all necessary information about the sensor input and the filter to make a decision. To achieve this, our state consists of the average scan matching likelihood $\bar{l}$, a boolean variable given by $N_{\mathit{eff}}^f < N_{\mathit{eff}}^g$, and a boolean variable if a known feature has currently been detected or not. This results in

$$S := \{\bar{l}\} \times \{1_{N_{\mathit{eff}}^f < N_{\mathit{eff}}^g}\} \times \{1_{\text{ feature detected}}\}. \tag{4}$$

The value of $\bar{l}$ is divided into (here seven) discrete intervals $(0.0-0.15,\ 0.16-0.3,\ 0.31-0.45,\ 0.46-0.6,\ 0.61-0.75,\ 0.76-0.9,\ 0.91-1.0)$, resulting in $7\times 2\times 2 = 28$ states. It is important to keep the number of states small since learning the policy otherwise may require too many computation resources (even as a preprocessing step which needs to be executed only once).

The policy is learned on simulated data where the true robot pose x_t^* is available in every time step t. We use the weighted average deviation from the true pose to define our reward-function. To avoid a punishment that result from wrong decisions in the past (e.g. a wrong rotation), we only use the deviation accumulated since the last evaluation step $t-1$:

$$r(s_t) \quad = \quad r(s_{t-1}) \ - \ \sum_{i=1}^{N} w^{(i)} \ ||x^{(i)} - x_t^*|| \qquad (5)$$

Deviations from the simulated path result in negative rewards. As mentioned in the previous section, each particle stores two weights. For calculating the weighted average, we use $w_g^{(i)}$ if the last action taken was a_g and $w_f^{(i)}$ if a_f was taken.

The environment for learning consists of building-like structures with hallways and an outdoor part that models a set of trees. We recorded a simulated path and executed the learning algorithm for 1000 times. During learning, we us an ϵ-greedy policy. In state s, a greedy policy chooses the action a which has the highest value $Q(s,a)$. In contrast to this, an ϵ-greedy policy allows exploratory actions by choosing a random action with likelihood ϵ.

This technique results in a policy that tells the robot when to select the feature-based representation and when to choose the grid map. Note that our approach to learn a strategy for making decisions is independent of the actual feature detector used. One could even use this approach to choose among multiple feature detectors. The overall mapping algorithm is depicted in Algorithm 1.

Algorithm 1 Our combined approach

Require:
$\mathcal{S}_{t-1}$, the sample set of the previous time step
$z_{l,t}$, the most recent laser scan
$z_{f,t}$, the most recent feature measurement
u_{t-1}, the most recent odometry measurement
Ensure:
$\mathcal{S}_t$, the new sample set

$\text{maptype} = \text{decide}(\mathcal{S}_{t-1}, z_{l,t}, z_{f,t}, u_{t-1})$

$\mathcal{S}_t = \{\}$
for all $s_{t-1}^{(i)} \in \mathcal{S}_{t-1}$ **do**
$< x_{t-1}^{(i)}, w_{g,t-1}^{(i)}, w_{f,t-1}^{(i)} m_{g,t-1}^{(i)}, m_{f,t-1}^{(i)} >= s_{t-1}^{(i)}$

// compute proposal
if $(\mathbf{maptype} = \mathbf{grid})$ **then**
$x_t^{(i)} \sim P(x_t \mid x_{t-1}^{(i)}, u_{t-1}, z_{l,t})$
else
$x_t^{(i)} \sim P(x_t \mid x_{t-1}^{(i)}, u_{t-1})$
end if

// update importance weights
$w_{g,t}^{(i)} = \text{updateGridWeight}(w_{g,t-1}^{(i)}, m_{g,t-1}^{(i)}, z_{l,t})$
$w_{f,t}^{(i)} = \text{updateFeatureWeight}(w_{f,t-1}^{(i)}, m_{f,t-1}^{(i)}, z_{f,t})$

// update maps
$m_{g,t}^{(i)} = \text{integrateScan}(m_{g,t-1}^{(i)}, x_t^{(i)}, z_{l,t})$
$m_{f,t}^{(i)} = \text{integrateFeatures}(m_{f,t-1}^{(i)}, x_t^{(i)}, z_{f,t})$

// update sample set
$\mathcal{S}_t = \mathcal{S}_t \cup \{< x_t^{(i)}, w_{g,t}^{(i)}, w_{f,t}^{(i)}, m_{g,t}^{(i)}, m_{f,t}^{(i)} >\}$
end for

for $i = 0$ to N **do**
if $(\mathbf{maptype} = \mathbf{grid})$ **then**
$w^{(i)} = w_g^{(i)}$
else
$w^{(i)} = w_f^{(i)}$
end if
end for

$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2}$
if $N_{\text{eff}} < T$ **then**
$\mathcal{S}_t = \text{resample}(\mathcal{S}_t, \{w^{(i)}\})$
end if

VI. Experiments

Our approach has been evaluated using simulated and real robot data. Real world experiments have been conducted using an ActivMedia Pioneer 2-AT robot equipped with a SICK LMS laser range finder. For generating the simulated data, we used the Carnegie Mellon Robot Navigation Toolkit.

The experiments have been designed to verify that our mapping approach is able to reduce the error compared to the purely feature-based technique (FastSLAM [12]) and to the purely grid-based approach [6]. In case the environment suggests the use of one single model, the result is obviously the same as using the original approach.

A. Simulation Experiments

The simulated environment used to test our approach is shown in Figure 2. It shows two symmetric buildings connected by an alley spanning 70 m in total. We simulated a laser range finder with a maximum range of 4m which is less than the distance between the trees in the alley (5m). The motivating example in the introduction of this paper shows example results obtained with the different approaches. Figure 1 (a) is the result of the purely feature-based FastSLAM approach. Since no features are found inside the building structures, the robot cannot correct its trajectory inside the buildings. In contrast, the path through the alley is well corrected.

The purely grid-based approach [6] is able to correctly map the buildings but introduces large errors in the alley (see Figure 1 (b)). Due to the limited range of the sensor, too few obstacles are observed and therefore no scan registration is possible and thus the grid-based approach fails to map the alley appropriately.

In contrast to this, our combined approach using the learned policy is able to correct the trajectory of the robot all the time by selecting the appropriate model. It uses the grid maps inside

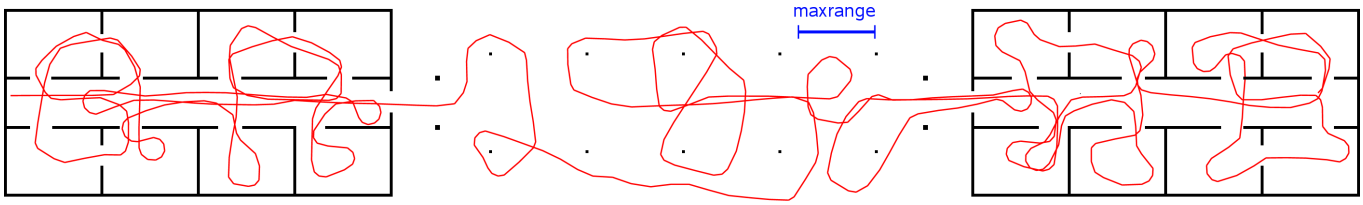


Fig. 2. Simulated environment used test our approach. This shows the ground truth map and trajectory of the robot.

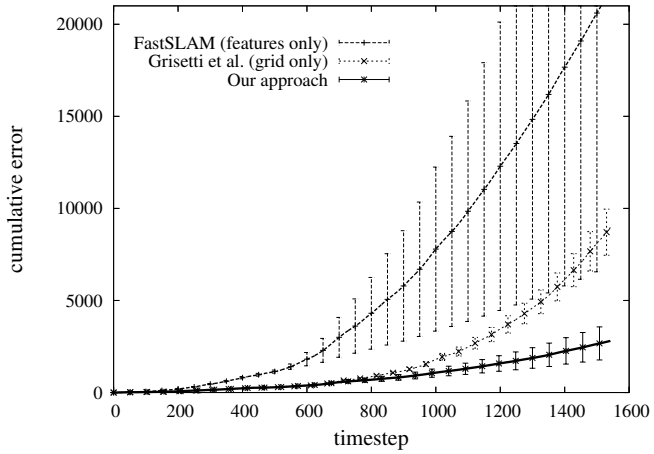


Fig. 3. Deviation of the weighted mean of the samples from ground truth using grid- and feature-model on their own and using the combined approach. The error bars illustrate the 0.05 confidence level.

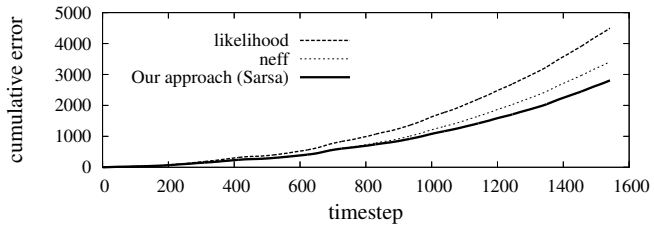


Fig. 4. Deviation of the weighted mean of the samples from ground truth using the scan-match likelihood heuristic, the N_{eff} heuristic and our approach.

the buildings and the features outside. The resulting map is shown in Figure 1 (c).

To evaluate our approach more quantitatively, we repeated this experiment for 20 times with different random seeds. We compared our approach to the pure feature-based approach and the pure grid-based approach. The results in Figure 3 show, that the combined approach is significantly better than both pure approaches (0.05 significance).

In addition, we compared the solution obtained by Sarsa with those of the scan-matching heuristic and the N_{eff} heuristic. We measured the absolute deviation from ground truth in every time step. Figure 4 illustrates that the average error of the learned model selection policy is lower than when using the heuristics. However, we could not show that this improvement is significant.

One interesting fact can be observed when comparing the results of these three technique by manual inspection. Even if the error measured as the deviation from the ground truth is not significantly smaller for the learned policy, the maps typically

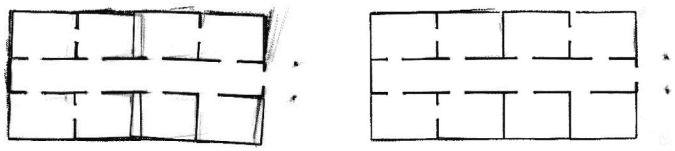


Fig. 5. Typical mapping results when using the likelihood-heuristic (left) and our Sarsa-based approach (right).

look nicer. The scan-match heuristic for example relies on a fixed threshold c_1. If the threshold is not optimally tuned, in can happen that the grid approach is not selected eventhough it would be better. This leads to walls which are more blurred or slightly missaligned. Figure 5 depicts a magnified view of two maps illustrating the difference. Unfortunately, it is hard to design a measure that is able to take this blurriness into account. A similar effect can be observed when using the N_{eff} criterion.

B. Real World Experiments

Real robot data has been recorded at Freiburg University. The computer science campus includes a parking space of about 50m by 120m (see Figure 6). Lamps are set in two rows at a distance of 16m. The dataset was recorded at a time when no cars were present and therefore only the lamps caused reflections of the laser beam. The robot was steered manually through a building, around the neighboring parking space, and back into the building. The trajectory is plotted in Figure 7. To evaluate our approach, we limited the maximum laser range to 12m, which is less than the distance between two lamps.

Since no ground truth was available, we measured the error to an approximated robot path which was generated using the grid-based approach of Grisetti *et al.* [6] with the full 80m sensor range (shown in red/dark gray in Figure 7). Due to the 80m range, the robot always observed enough obstacles to build an accurate map. Figure 8 shows the error of the weighted mean trajectory over time. In summary, the real robot experiment leads to similar results as simulated experiments. The combined approach performed better compared to both traditional SLAM techniques with 12m sensor range.

The computational requirements of the presented approach are approximatively the sum of the individual techniques. On a standard PC, our implementation runs online.

VII. Conclusions

In this paper, we presented an improved approach to learning models of the environment with a Rao-Blackwellized particle filters. Our approach maintains feature maps as well

Fig. 6. Parking space at Freiburg campus.

Fig. 7. Grid map of parking space and neighboring building 078 at Freiburg campus. The approximated robot trajectory is shown in red/dark gray, the result of our combined mapping approach is shown in green/light gray.

as grid maps to represent spatial structures. This allows the robot to select the model which provides the best expected map estimate. The model selection procedure is obtained by a reinforcement learning approach. The robot considers the previous estimate as well as the current observations to chose the model that will be used in the upcoming correction step. The process itself is independent of the actual feature detector. Our approach has been implemented and evaluated on real robot data as well as in simulation experiments. We showed that the presented technique allows a robot to more robustly learn maps of different types of environments. It ourperforms traditional approach that use only features or only grid maps.

ACKNOWLEDGMENT

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A3), and by the EC under contract number FP6-2005-IST-6-RAWSEEDS, FP6-2005-IST-5-muFly, and FP6-004250-CoSy.

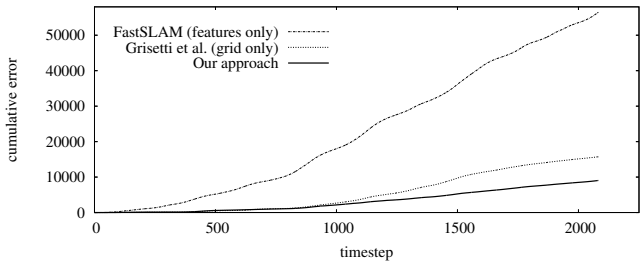


Fig. 8. Deviation of the weighted mean of the samples from the estimated trajectory (using the 80m range scanner).

REFERENCES

[1] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge, 1998.

[2] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 176–183, Stanford, CA, USA, 2000.

[3] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[4] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[5] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17–26, Seattle, WA, USA, 2001.

[6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[7] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[8] E.M. Nebot J.I. Nieto, J.E. Guivant. The hybrid metric maps (HYMMs): A novel map representation for denseslam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[9] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[11] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[12] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[13] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[14] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[15] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[17] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*, chapter Robot Perception, pages 171–172. MIT Press, 2005.

[18] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *J. of Robotics Research*, 23(7/8):693–716, 2004.

[19] J. Uhlmann. *Dynamic Map Building and Localization: New Theoretical Foundations*. PhD thesis, University of Oxford, 1995.

[C22] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent

Giorgio Grisetti Cyrill Stachniss Slawomir Grzonka Wolfram Burgard
University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany

***Abstract*— In 2006, Olson *et al.* presented a novel approach to address the graph-based simultaneous localization and mapping problem by applying stochastic gradient descent to minimize the error introduced by constraints. Together with multi-level relaxation, this is one of the most robust and efficient maximum likelihood techniques published so far. In this paper, we present an extension of Olson's algorithm. It applies a novel parameterization of the nodes in the graph that significantly improves the performance and enables us to cope with arbitrary network topologies. The latter allows us to bound the complexity of the algorithm to the size of the mapped area and not to the length of the trajectory as it is the case with both previous approaches. We implemented our technique and compared it to multi-level relaxation and Olson's algorithm. As we demonstrate in simulated and in real world experiments, our approach converges faster than the other approaches and yields accurate maps of the environment.**

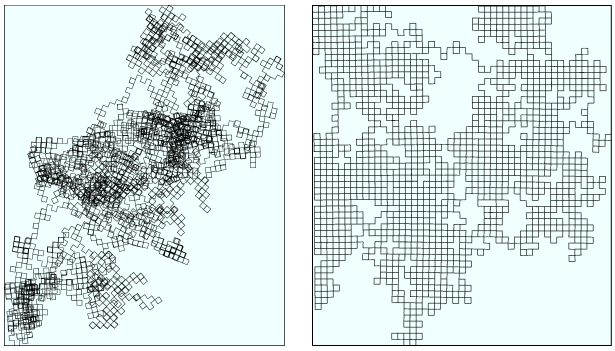


Fig. 1. The left image shows an uncorrected network with around 100k poses and 450k constraints. The right image depicts the network after applying our error minimization approach (100 iterations, 17s on a P4 CPU with 1.8GHz).

I. Introduction

Models of the environment are needed for a wide range of robotic applications, including search and rescue, automated vacuum cleaning, and many others. Learning maps has therefore been a major research focus in the robotics community over the last decades. Learning maps under uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. The approaches mainly differ due to the underlying estimation technique such as extended Kalman filters, information filters, particle filters, or least-square error minimization techniques.

In this paper, we consider the so-called "graph-based" or "network-based" formulation of the SLAM problem in which the poses of the robot are modeled by nodes in a graph [2, 5, 6, 7, 11, 13]. Constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes.

The goal of an algorithm designed to solve this problem is to find the configuration of the nodes that maximizes the observation likelihood encoded in the constraints. Often, one refers to the negative observation likelihood as the error or the energy in the network. An alternative view to the problem is given by the spring-mass model in physics. In this view, the nodes are regarded as masses and the constraints as springs connected to the masses. The minimal energy configuration of the springs and masses describes a solution to the mapping problem. Figure 1 depicts such a constraint network as a motivating example.

A popular solution to this class of problems are iterative approaches. They can be used to either correct all poses simultaneously [6, 9, 11] or to locally update parts of the network [2, 5, 7, 13]. Depending on the used technique, different parts of the network are updated in each iteration. The strategy for defining and performing these local updates has a significant impact on the convergence speed.

Our approach uses a tree structure to define and efficiently update local regions in each iteration. The poses of the individual nodes are represented in an incremental fashion which allows the algorithm to automatically update successor nodes. Our approach extends Olson's algorithm [13] and converges significantly faster to a network configuration with a low error. Additionally, we are able to bound the complexity to the size of the environment and not to the length of the trajectory.

The remainder of this paper is organized as follows. After discussing the related work, Section III explains the graph-based formulation of the mapping problem. Subsequently, we explain the usage of stochastic gradient descent to find network configurations with small errors. Section V introduces our tree parameterization and in Section VI we explain how to obtain such a parameterization tree from robot data. We finally present our experimental results in Section VII.

II. Related Work

Mapping techniques for mobile robots can be classified according to the underlying estimation technique. The most popular approaches are extended Kalman filters (EKFs), sparse extended information filters, particle filters, and least square error minimization approaches. The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [10, 14]. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely

identifiable. There exist techniques [12] to deal with unknown data association in the SLAM context, however, if certain assumptions are violated the filter is likely to diverge [8].

Frese's TreeMap algorithm [4] can be applied to compute nonlinear map estimates. It relies on a strong topological assumption on the map to perform sparsification of the information matrix. This approximation ignores small entries in the information matrix. In this way, Frese is able to perform an update in $\mathcal{O}(\log n)$ where n is the number of features.

An alternative approach is to find maximum likelihood maps by least square error minimization. The idea is to compute a network of relations given the sequence of sensor readings. These relations represent the spatial constraints between the poses of the robot. In this paper, we also follow this way of formulating the SLAM problem. Lu and Milios [11] first applied this approach in robotics to address the SLAM problem using a kind of brute force method. Their approach seeks to optimize the whole network at once. Gutmann and Konolige [6] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Howard *et al.* [7] apply relaxation to localize the robot and build a map. Duckett *et al.* [2] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of constraints. In order to make the problem linear, they assume knowledge about the orientation of the robot. Frese *et al.* [5] propose a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions. MLR is reported to provide very good results and is probably the best relaxation technique in the SLAM context at the moment.

Note that such maximum likelihood techniques as well as our method focus on computing the best map and assume that the data association is given. The ATLAS framework [1] or hierarchical SLAM [3], for example, can be used to obtain such data associations (constraints). They also apply a global optimization procedure to compute a consistent map. One can replace such optimization procedures by our algorithm and in this way make ATLAS or hierarchical SLAM more efficient.

The approach closest to the work presented here is the work of Olson *et al.* [13]. They apply stochastic gradient descent to reduce the error in the network. They also propose a representation of the nodes which enables the algorithm to perform efficient updates. The approach of Olson *et al.* is one of the current state-of-the-art approaches for optimizing networks of constraints. In contrast to their technique, our approach uses a different parameterization of the nodes in the network that better takes into account the topology of the environment. This results in a faster convergence of our algorithm.

Highly sophisticated optimization techniques such as MLR or Olson's algorithm are restricted to networks that are built in an incremental way. They require as input a sequence of robot poses according to the traveled path. First, this makes it difficult to use these techniques in the context of multi-robot SLAM. Second, the complexity of the algorithm depends on the length of the trajectory traveled by the robot and not on the size of the environment. This dependency prevents to use these approaches in the context of lifelong map learning.

One motivation of our approach is to build a system that depends on the size of the environment and not explicitely on the length of the trajectory. We designed our approach in a way that it can be applied to arbitrary networks. As we will show in the remainder of this paper, the ability to use arbitrary networks allows us to prune the trajectory so that the complexity of our approach depends only on the size of the environment. Furthermore, our approach proposes a more efficient parameterization of the network when applying gradient descent.

III. On Graph-based SLAM

Most approaches to graph-based SLAM focus on estimating the most-likely configuration of the nodes and are therefore referred to as maximum-likelihood (ML) techniques [2, 5, 6, 11, 13]. They do not consider to compute the full posterior about the map and the poses of the robot. The approach presented in this paper also belongs to this class of methods.

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. For a more precise formulation consider the following definitions:

- $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ is a vector of parameters which describes a configuration of the nodes. Note that the parameters x_i do not need to be the absolute poses of the nodes. They are arbitrary variables which can be mapped to the poses of the nodes in real world coordinates.
- δ_{ji} describes a constraint between the nodes j and i. It refers to an observation of node j seen from node i. These constraints are the edges in the graph structure.
- Ω_{ji} is the information matrix modeling the uncertainty of δ_{ji}.
- $f_{ji}(\mathbf{x})$ is a function that computes a zero noise observation according to the current configuration of the nodes j and i. It returns an observation of node j seen from node i.

Given a constraint between node j and node i, we can define the *error* e_{ji} introduced by the constraint as

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

as well as the *residual* r_{ji}

$$r_{ji}(\mathbf{x}) \quad = \quad -e_{ji}(\mathbf{x}). \tag{2}$$

Note that at the equilibrium point, e_{ji} is equal to 0 since $f_{ji}(\mathbf{x}) = \delta_{ji}$. In this case, an observation perfectly matches the current configuration of the nodes. Assuming a Gaussian observation error, the negative log likelihood of an observation f_{ji} is

$$\begin{aligned} F_{ji}(\mathbf{x}) \quad &\propto \quad (f_{ji}(\mathbf{x}) - \delta_{ji})^T \, \Omega_{ji} \, (f_{ji}(\mathbf{x}) - \delta_{ji}) && (3)\\ &= \quad e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}) && (4)\\ &= \quad r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (5) \end{aligned}$$

Under the assumption that the observations are independent, the overall negative log likelihood of a configuration $\mathbf{x}$ is

$$\begin{aligned} F(\mathbf{x}) \quad &= \quad \sum_{\langle j,i \rangle \in \mathcal{C}} F_{ji}(\mathbf{x}) && (6)\\ &= \quad \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (7) \end{aligned}$$

Here $\mathcal{C} = \{\langle j_1, i_1\rangle, \ldots, \langle j_M, i_M\rangle\}$ is set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists.

The goal of a ML approach is to find the configuration $\mathbf{x}^*$ of the nodes that maximizes the likelihood of the observations. This can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{8}$$

IV. Stochastic Gradient Descent for Maximum Likelihood Mapping

Olson *et al.* [13] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to address the SLAM problem. The approach minimizes Eq. (8) by iteratively selecting a constraint $\langle j, i\rangle$ and by moving the nodes of the network in order to decrease the error introduced by the selected constraint. Compared to the standard formulation of gradient descent, the constraints are not optimized as a whole but individually. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \underbrace{\lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji}}_{\Delta \mathbf{x}_{ji}} \tag{9}$$

Here $\mathbf{x}$ is the set of variables describing the locations of the poses in the network and $\mathbf{H}^{-1}$ is a preconditioning matrix. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, and r_{ji} is the residual.

Reading the term $\Delta \mathbf{x}_{ji}$ of Eq. (9) from right to left gives an intuition about the sequential procedure used in SGD:

- r_{ji} is the residual which is the opposite of the error vector. Changing the network configuration in the direction of the residual r_{ji} will decrease the error e_{ji}.
- Ω_{ji} represents the information matrix of a constraint. Multiplying it with r_{ji} scales the residual components according to the information encoded in the constraint.
- J_{ji}^T: The role of the Jacobian is to map the residual term into a set of variations in the parameter space.
- $\mathbf{H}$ is the Hessian of the system and it represents the curvature of the error function. This allows us to scale the variations resulting from the Jacobian depending on the curvature of the error surface. We actually use an approximation of $\mathbf{H}$ which is computed as

  $$\mathbf{H} \quad \simeq \quad \sum_{\langle j,i\rangle} J_{ji} \Omega_{ji} J_{ji}^T. \tag{10}$$

  Rather than inverting the full Hessian which is computationally expensive, we approximate it by

  $$\mathbf{H}^{-1} \quad \simeq \quad [\mathrm{diag}(\mathbf{H})]^{-1}. \tag{11}$$

- λ is a learning rate which decreases with the iteration of SGD and which makes the system to converge to an equilibrium point.

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing the constraints individually. Each time a solution for one of these subproblems is found, the network is updated accordingly. Obviously, updating the different constraints one after each other can have opposite effects on a subset of variables. To avoid infinitive oscillations, one uses the learning rate to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different sub-problems to asymptotically converge towards an equilibrium point that is the solution reported by the algorithm.

This framework allows us to iteratively reduce the error given the network of constraints. The optimization approach, however, leaves open how the nodes are represented (parameterized). Since the parameterization defines also the structure of the Jacobians, it has a strong influence on the performance of the algorithm.

The next section addresses the problem of how to parameterize a graph in order to efficiently carry out the optimization approach.

V. Network Parameterizations

The poses $\mathbf{p} = \{p_1, \ldots, p_n\}$ of the nodes define the configuration of the network. The poses can be described by a vector of parameters $\mathbf{x}$ such that a bijective mapping g between $\mathbf{p}$ and $\mathbf{x}$ exists

$$\mathbf{x} = g(\mathbf{p}) \qquad\qquad \mathbf{p} = g^{-1}(\mathbf{x}). \tag{12}$$

As previously explained, in each iteration SGD decomposes the problem into a set of subproblems and solves them successively. In this work, a subproblem is defined as the optimization of a single constraint. Different solutions to the individual subproblems can have antagonistic effects when combining them.

The parameterization g defines also the subset of variables that are modified by a single constraint update. A good parameterization defines the subproblems in a way that the combination step leads only to small changes of the individual solutions.

A. Incremental Pose Parameterization

Olson *et al.* propose the so-called incremental pose parameterization. Given a set of node locations p_i and given a fixed order on the nodes, the incremental parameters x_i can be computed as follows

$$x_i \quad = \quad p_i - p_{i-1}. \tag{13}$$

Note that x_i is computed as the difference between two subsequent nodes and not by motion composition. Under this parameterization, the error in the global reference frame (indicated by primed variables) has the following form

$$e'_{ji} \quad = \quad p_j - (p_i \oplus \delta_{ji}) \tag{14}$$

$$\quad = \quad \left(\sum_{k=i+1}^{j} x_k\right) + \underbrace{\left(\prod_{k=1}^{i} \tilde{R}_k\right)}_{R_i} \delta_{ji}, \tag{15}$$

where $\oplus$ is the motion composition operator according to Lu and Milios [11] and $\tilde{R}_k$ the homogenous rotation matrix of the *parameter* x_k. The term R_k is defined as the rotation matrix of the *pose* p_k. The information matrix in the global reference frame can be computed as

$$\Omega'_{ji} \quad = \quad R_i \Omega_{ji} R_i^T. \tag{16}$$

According to Olson *et al.* [13], neglecting the contribution of the angular terms of $x_0, \ldots, x_i$ to the Jacobian results in the following simplified form

$$J'_{ji} = \sum_{k=i+1}^{j} \mathcal{I}_k \quad \text{with} \quad \mathcal{I}_k = (0 \cdots 0 \underbrace{I}_{k} 0 \cdots 0). \quad (17)$$

Here 0 is the 3 by 3 zero matrix and I is the 3 by 3 identity.

Updating the network based on the constraint $\langle j, i \rangle$ with such an Jacobian results in keeping the node i fixed and in distributing the residual along all nodes between j and i.

Olson *et al.* weight the residual proportional to $j-i$ which is the number of nodes involved in the constraint. The parameter x_k of the node k with $k = i+1, \ldots, j$ is updated as follows

$$\Delta x_k \quad = \quad \lambda w_k \Omega'_{ji} r'_{ji}, \quad (18)$$

where the weight w_k is computed as

$$w_k = (j-i) \left[\sum_{m=i+1}^{j} D_m^{-1} \right]^{-1} D_k^{-1}. \quad (19)$$

In Eq. (19), D_k are the matrices containing the diagonal elements of the k^{th} block of the Hessian $\mathbf{H}$. Intuitively, each variable is updated proportional to the uncertainty about that variable. Note that the simple form of the Jacobians allows us to update the parameter vector for each node individually as expressed by Eq. (18).

The approach presented in this section is currently one of the best solutions to ML mapping. However, it has the following drawbacks:

- In practice, the incremental parameterization cannot deal with arbitrarily connected networks. This results from the approximation made in Eq. (17), in which the angular components are ignored when computing the Jacobian. This approximation is only valid if the subsequent nodes in Eq. (13) are spatially close. Furthermore, the way the error is distributed over the network assumes that the nodes are ordered according to poses along the trajectory. This results in adding a large number of nodes to the network whenever the robot travels for a long time in the same region. This requirement prevents an approach from merging multiple nodes into a single one. Merging or pruning nodes, however, is a necessary precondition to allow the robot lifelong map learning.
- When updating a constraint between the nodes j and i, the parameterization requires to change the j-i nodes. As a result, each node is likely to be updated by several constraints. This leads to a high interaction between constraints and will typically reduce the convergence speed of SGD. For example, the node k will be updated by all constraints $\langle j', i' \rangle$ with $i' < k \leq j'$. Note that using an intelligent lookup structure, this operation can be carried out in $\mathcal{O}(\log n)$ time where n is the number of nodes in the network [13]. Therefore, this is a problem of convergence speed of SGD and not a computational problem.

B. Tree Parameterization

Investigating a different parameterization which preserves the advantages of the incremental one but overcomes its drawbacks is the main motivation for our approach. First, our method should be able to deal with arbitrary network topologies. This would enable us to compress the graph whenever robot revisits a place. As a result, the size of the network would be proportional to the visited area and not to the length of the trajectory. Second, the number of nodes in the graph updated by each constraint should mainly depend on the topology of the environment. For example, in case of a loop-closure a large number of nodes need to be updated but in all other situations the update is limited to a small number of nodes in order to keep the interactions between constraints small.

Our idea is to first construct a spanning tree from the (arbitrary) graph. Given such a tree, we define the parameterization for a node as

$$x_i \quad = \quad p_i - p_{\text{parent}(i)}, \quad (20)$$

where $p_{\text{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (20), the tree stores the differences between poses. As a consequence, one needs to process the tree up to the root to compute the actual pose of a node in the global reference frame.

However, to obtain only the difference between two arbitrary nodes, one needs to traverse the tree from the first node upwards to the first common ancestor of both nodes and then downwards to the second node. The same holds for computing the error of a constraint. We refer to the nodes one needs to traverse on the tree as the path of a constraint. For example, $\mathcal{P}_{ji}$ is the path from node i to node j for the constraint $\langle j, i \rangle$. The path can be divided into an ascending part $\mathcal{P}_{ji}^{[-]}$ of the path starting from node i and a descending part $\mathcal{P}_{ji}^{[+]}$ to node j. We can then compute the error in the global frame by

$$\begin{aligned} e'_{ji} &= p_j - (p_i \oplus \delta_{ji}) && (21) \\ &= p_j - (p_i + R_i \delta_{ji}) && (22) \\ &= \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - R_i \delta_{ji}. && (23) \end{aligned}$$

Here R_i is the rotation matrix of the pose p_i. It can be computed according to the structure of the tree as the product of the individual rotation matrices along the path to the root.

Note that this tree does not replace the graph as an internal representation. The tree only defines the parameterization of the nodes. It can furthermore be used to define an order in which the optimization algorithm can efficiently process the constraints as we will explain in the remainder of this section. For illustration, Figure 2 (a) and (b) depict two graphs and possible parameterization trees.

Similar to Eq. (16), we can express the information matrix associated to a constraint in the global frame by

$$\Omega'_{ji} \quad = \quad R_i \Omega_{ji} R_i^T. \quad (24)$$

As proposed in [13], we neglect the contribution of the rotation matrix R_i in the computation of the Jacobian. This approximation speeds up the computation significantly. Without

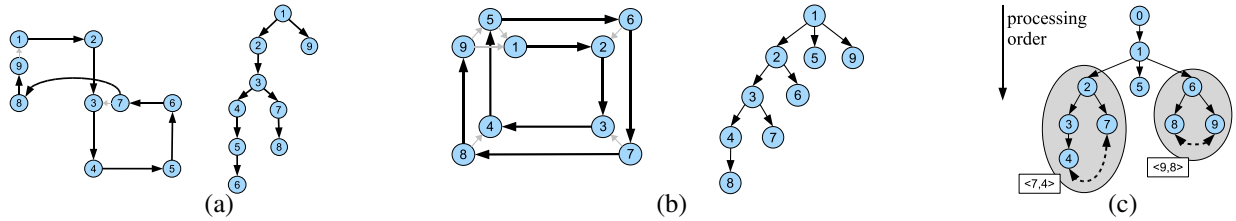


Fig. 2. (a) and (b): Two small example graphs and the trees used to determine the parameterizations. The small grey connections are constraints introduced by observations where black ones result from odometry. (c) Processing the constraints ordered according to the node with the smallest level in the path avoids the recomputation of rotational component of all parents. The same holds for subtrees with different root nodes on the same level.

this approximation the update of a single constraint influences the poses of all nodes up to the root.

The approximation leads to the following Jacobian:

$$J'_{ji} = \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}} \quad (25)$$

Compared to the approach described in the previous section, the number of updated variables per constraint is in practice smaller when using the tree. Our approach updates $|\mathcal{P}_{ji}|$ variables rather than $j - i$. The weights w_k are computed as

$$w_k = |\mathcal{P}_{ji}| \left[\sum_{m \in \mathcal{P}_{ji}}^{j} D_m^{-1} \right]^{-1} D_k^{-1}, \quad (26)$$

where D_k is the k-th diagonal block element of $\mathbf{H}$. This results in the following update rule for the variable x_k

$$\Delta x_k = \lambda w_k \cdot \mathrm{s}(x_k, i, j) \cdot \Omega'_{ji} r'_{ji}, \quad (27)$$

where the value of $\mathrm{s}(x_k, j, i)$ is $+1$ or -1 depending on where the parameter x_k is located on the path $\mathcal{P}_{ji}$:

$$\mathrm{s}(x_k, j, i) = \begin{cases} +1 & \text{if } x_k \in \mathcal{P}_{ji}^{[+]} \\ -1 & \text{if } x_k \in \mathcal{P}_{ji}^{[-]} \end{cases} \quad (28)$$

Our parameterization maintains the simple form of the Jacobians which enables us to perform the update of each parameter variable individually (as can be seen in Eq. (27)). Note that in case one uses a tree that is degenerated to a list, this parameterization is equal to the one proposed by Olson *et al.* [13]. In case of a non-degenerated tree, our approach offers several advantages as we will show in the experimental section of this paper.

The optimization algorithm specifies how to update the nodes but does not specify the order in which to process the constraints. We can use our tree parameterization to sort the constraints which allows us to reduce the computational complexity of our approach.

To compute the residual of a constraint $\langle j, i \rangle$, we need to know the rotational component of the node i. This requires to traverse the tree up to the first node for which the rotational component is known. In the worst case, this is the root of the tree.

Let the *level* of a node be the distance in the tree between the node itself and the root. Let z_{ji} be the node in the path of the constraint $\langle j, i \rangle$ with the smallest level. The level of the constraint is then defined as the level of z_{ji}.

Our parameterization implies that updating a constraint will never change the configuration of a node with a level smaller than the level of the constraint. Based on this knowledge, we can sort the constraints according to their level and process them in that order. As a result, it is sufficient to access the parent of z_{ji} to compute the rotational component of the node i since all nodes with a smaller level than z_{ji} have already been corrected.

Figure 2 (c) illustrates such a situation. The constraint $\langle 7, 4 \rangle$ with the path $4, 3, 2, 7$ does not change any node with a smaller level than the one of node 2. It also does not influence other subtrees on the same level such as the nodes involved in the constraint $\langle 9, 8 \rangle$.

In the following section, we describe how we actually build the tree given the trajectory of a robot or an arbitrary network as input.

VI. Construction of the Spanning Tree

When constructing the parameterization tree, we distinguish two different situations. First, we assume that the input is a sequence of positions belonging to a trajectory traveled by the robot. Second, we explain how to build the tree given an arbitrary graph of relations.

In the first case, the subsequent poses are located closely together and there exist constraints between subsequent poses resulting from odometry or scan-matching. Further constraints between arbitrary nodes result from observations when revisiting a place in the environment. In this setting, we build our parameterization tree as follows:

1) We assign a unique id to each node based on the timestamps and process the nodes accordingly.
2) The first node is the root of the tree (and therefore has no parent).
3) As the parent of a node, we choose the node with the smallest id for which a constraint to the current node exists.

This tree can be easily constructed on the fly. The Figures 2 (a) and (b) illustrates graphs and the corresponding trees. This tree has a series of nice properties when applying our optimization algorithm to find a minimal error configuration of the nodes. These properties are:

- The tree can be constructed incrementally: when adding a new node it is not required to change the existing tree.
- In case the robot moves through nested loops, the interaction between the updates of the nodes belonging to the individual loops depends on the number of nodes the loops have in common.

- When retraversing an already mapped area and adding constraints between new and previously added nodes, the length of the path in the tree between these nodes is small. This means that only a small number of nodes need to be updated.

The second property is illustrated in Figure 2 (a). The two loops in that image are only connected via the constraint between the nodes 3 and 7. They are the only nodes that are updated by constraints of both loops.

The third property is illustrated in Figure 2 (b). Here, the robot revisits a loop. The nodes 1 to 4 are chosen as the parents for all further nodes. This results in short paths in the tree when updating the positions of the nodes while retraversing known areas.

The complexity of the approach presented so far depends on the length of the trajectory and not on the size of the environment. These two quantities are different in case the robot revisits already known areas. This becomes important whenever the robot is deployed in a bounded environment for a long time and has to update its map over time. This is also known as lifelong map learning. Since our parameterization is not restricted to a trajectory of sequential poses, we have the possibility of a further optimization. Whenever the robot revisits a known place, we do not need to add new nodes to the graph. We can assign the current pose of the robot to an already existing node in the graph.

Note that this can be seen as an approximation similar to adding a rigid constraint neglecting the uncertainty of the corresponding observation. However, in case local maps (e.g., grid maps) are used as nodes in the network, it makes sense to use such an approximation since one can localize a robot in an existing map quite accurately.

To also avoid adding new constraints to the network, we can refine an existing constraint between two nodes in case of a new observation. Given a constraint $\delta_{ji}^{(1)}$ between the nodes j and i in the graph and a new constraint $\delta_{ji}^{(2)}$ based on the current observation. Both constraints can be combined to a single constraint which has the following information matrix and mean:

$$\Omega_{ji} = \Omega_{ji}^{(1)} + \Omega_{ji}^{(2)} \tag{29}$$

$$\delta_{ji} = \Omega_{ji}^{-1}(\Omega_{ji}^{(1)} \cdot \delta_{ji}^{(1)} + \Omega_{ji}^{(2)} \cdot \delta_{ji}^{(2)}) \tag{30}$$

As a result, the size of the problem does not increase when revisiting known locations. As the experiments illustrate, this node reduction technique leads to an increased convergence speed.

In case the input to our algorithm is an arbitrary graph and no natural order of the nodes is provided, we compute a minimal spanning tree to define the parameterization. Since no additional information (like consecutive poses according to a trajectory) is available, we cannot directly infer which parts of the graph are well suited to form a subtree in the parameterization tree. The minimal spanning tree appears to yield comparable results with respect to the number of iterations needed for convergence in all our experiments.

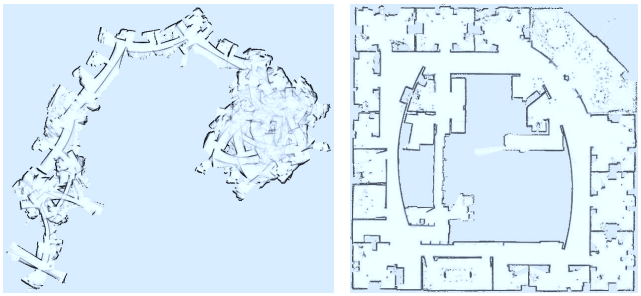


Fig. 3. The map of the Intel Research Lab before (left) and after (right) execution of our algorithm (1000 nodes, runtime $<$1s).

VII. Experiments

This section is designed to evaluate the properties of our tree parameterization for learning maximum likelihood maps. We first show that such a technique is well suited to generate accurate occupancy grid maps given laser range data and odometry from a real robot. Second, we provide simulation experiments on large-scale datasets. We furthermore provide a comparison between our approach, Olson's algorithm [13], and multi-level relaxation by Frese *et al.* [5]. Finally, we analyze our approach and investigate properties of the tree parameterization in order to explain why we obtain better results then the other methods.

A. Real World Experiments

The first experiment is designed to illustrate that our approach can be used to build maps from real robot data. The goal was to build an accurate occupancy grid map given the laser range data obtained by the robot. The nodes of our graph correspond to the individual poses of the robot during data acquisition. The constraints result from odometry and from the pair-wise matching of laser range scans. Figure 3 depicts two maps of the Intel Research Lab in Seattle. The left one is constructed from raw odometry and the right one is the result obtained by our algorithm. As can be seen, the corrected map shows no inconsistencies such as double corridors. Note that this dataset is freely available on the Internet.

B. Simulated Experiments

The second set of experiments is designed to measure the performance of our approach quantitatively. Furthermore, we compare our technique to two current state-of-the-art SLAM approaches that work on constraint networks, namely multi-level relaxation by Frese *et al.* [5] and Olson's algorithm [13]. In the experiments, we used the two variants of our method: the one that uses the node reduction technique described in Section VI and the one that maintains all the nodes in the graph.

In our simulation experiments, we moved a virtual robot on a grid world. An observation is generated each time the current position of the robot was close to a previously visited location. We corrupted the observations with a variable amount of noise for testing the robustness of the algorithms. We simulated different datasets resulting in graphs with a number of constraints between around 4,000 and 2 million.

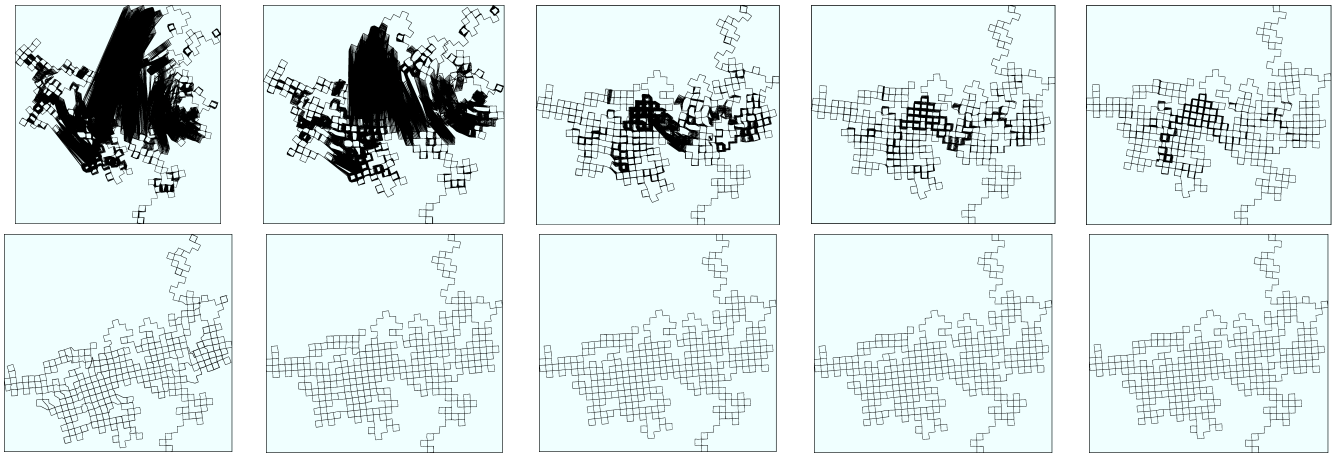


Fig. 4. Results of Olson's algorithm (first row) and our approach (second row) after 1, 10, 50, 100, 300 iterations for a network with 64k constraints. The black areas in the images result from constraints between nodes which are not perfectly corrected after the corresponding iteration (for timings see Figure 6).

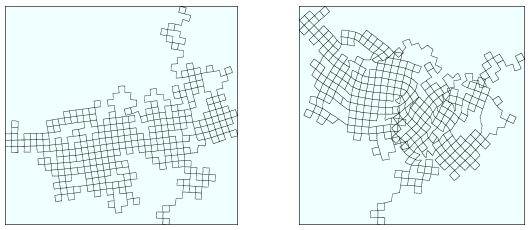


Fig. 5. The result of MLR strongly depends on the initial configuration of the network. Left: small initial pose error, right: large initial pose error.

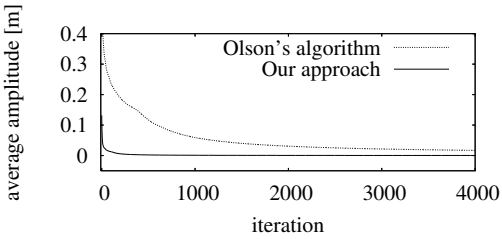


Fig. 7. The average amplitude of the oscillations of the nodes due to the antagonistic effects of different constraints.

Figure 4 depicts a series of graphs obtained by Olson's algorithm and our approach after different iterations. As can be seen, our approach converges faster. Asymptotically, both approaches converge to a similar solution.

In all our experiments, the results of MLR strongly depended on the initial positions of the nodes. In case of a good starting configuration, MLR converges to an accurate solution similar to our approach as shown in Figure 5 (left). Otherwise, it is likely to diverge (right). Olson's approach as well as our technique are more or less independent of the initial poses of the nodes.

To evaluate our technique quantitatively, we first measured the error in the network after each iteration. The left image of Figure 6 depicts a statistical experiments over 10 networks with the same topology but different noise realizations. As can be seen, our approach converges significantly faster than the approach of Olson *et al.* For medium size networks, both approaches converge asymptotically to approximatively the same error value (see middle image). For large networks, the high number of iterations needed for Olson's approach prevented us from showing this convergence experimentally. Due to the sake of brevity, we omitted comparisons to EKF and Gauss Seidel relaxation because Olson *et al.* already showed that their approach outperforms such techniques.

Additionally, we evaluated in Figure 6 (right) the average computation time per iteration of the different approaches. As a result of personal communication with Edwin Olson, we furthermore analyzed a variant of his approach which is restricted to spherical covariances. It yields similar execution times *per iteration* than our approach. However, this restricted variant has still the same converge speed with respect to the number of iterations than Olson's unrestricted technique. As can be seen from that picture, our node reduction technique speeds up the computations up to a factor of 20.

C. Analysis of the Algorithm

The experiments presented above illustrated that our algorithm offers significant improvements compared to both other techniques. The goal of this section is to experimentally point out the reasons for these improvements.

The presented tree parameterization allows us to decompose the optimization of the whole graph into a set of weakly interacting problems. A good measure for evaluating the interaction between the constraints is the average number l of updated nodes per constraint. For example, a network with a large value of l has typically a higher number of interacting constraints compared to networks with low values of l. In all experiments, our approach had a value between 3 and 7. In contrast to that, this values varies between 60 and 17,000 in Olson's approach on the same networks. Note that such a high average path length reduces the convergence speed of Olson's algorithm but does not introduce a higher complexity.

The optimization approach used in this paper as well as in Olson's algorithm updates for each constraint the involved nodes to minimize the error in the network. As a result, different constraints can update poses in an antagonistic way during one iteration. This leads to oscillations in the position

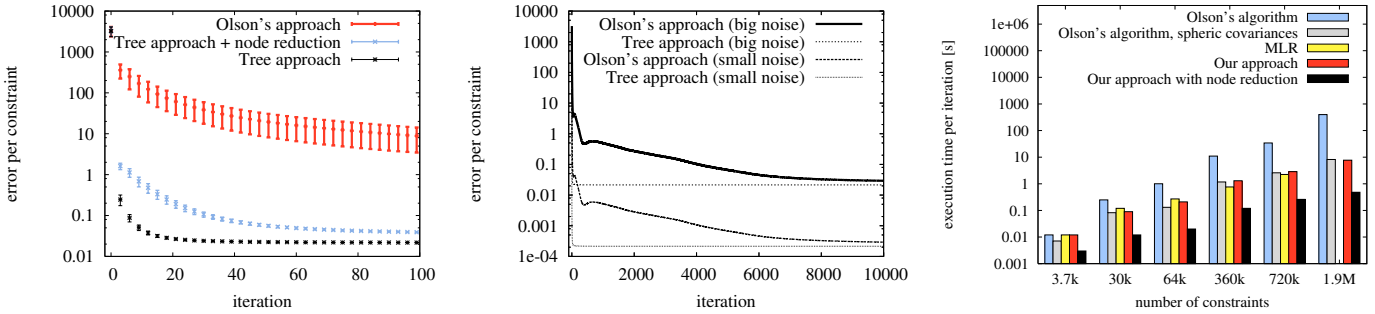


Fig. 6. The left image illustrates shows the error of our and Olson's approach in a statistical experiment ($\sigma = 0.05$ confidence). The image in the middle shows that both techniques converge asymptotically to the same error. The right image shows the average execution time *per iteration* for different networks. For the 1.9M constraints network, the executing of MLR required memory swapping and the result is therefore omitted.

of a node before convergence. Figure 7 illustrates the average amplitude of such an oscillations for Olson's algorithm as well as for our approach. As can be seen, our techniques converges faster to an equilibrium point. This a further reason for the higher convergence speed of our approach.

D. Complexity

Due to the nature of gradient descent, the complexity of our approach per iteration depends linearly on the number of constraints. For each constraint $\langle j, i \rangle$, our approach modifies exactly those nodes which belong to the path $\mathcal{P}_{ji}$ in the tree. Since each constraint has an individual path length, we consider the average path length l. This results in an complexity per iteration of $\mathcal{O}(M \cdot l)$, where M is the number of constraints. In all our experiments, l was approximatively $\log N$, where N is the number of nodes. Note that given our node reduction technique, M as well as N are bounded by the size of the environment and not by the length of the trajectory.

A further advantage of our technique compared to MLR is that it is easy to implement. The function that performs a single iteration requires less than 100 lines of C++ code. An open source implementation, image and video material, and the datasets are available at the authors' web-pages.

VIII. Conclusion

In this paper, we presented a highly efficient solution to the problem of learning maximum likelihood maps for mobile robots. Our technique is based on the graph-formulation of the simultaneous localization and mapping problem and applies a gradient descent based optimization scheme. Our approach extends Olson's algorithm by introducing a tree-based parameterization for the nodes in the graph. This has a significant influence on the convergence speed and execution time of the method. Furthermore, it enables us to correct arbitrary graphs and not only a list of sequential poses. In this way, the complexity of our method depends on the size of the environment and not directly on the length of the input trajectory. This is an important precondition to allow a robot lifelong map learning in its environment.

Our method has been implemented and exhaustively tested on simulation experiments as well as on real robot data. We furthermore compared our method to two existing, state-of-the-art solutions which are multi-level relaxation and Olson's algorithm. Our approach converges significantly faster than both approaches and yields accurate maps with low errors.

Acknowledgment

The authors would like to gratefully thank Udo Frese for his insightful comments and for providing us his MLR implementation for comparisons. Further thanks to Edwin Olson for his helpful comments on this paper. This work has partly been supported by the DFG under contract number SFB/TR-8 (A3) and by the EC under contract number FP6-2005-IST-5-muFly and FP6-2005-IST-6-RAWSEEDS.

References

[1] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 1899–1906, Taipei, Taiwan, 2003.

[2] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287 – 300, 2002.

[3] C. Estrada, J. Neira, and J.D. Tardós. Hierachical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[4] U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[5] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[6] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symp. on Comp. Intelligence in Robotics and Automation*, pages 318–325, Monterey, CA, USA, 1999.

[7] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1055–1060, 2001.

[8] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[9] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3232–3238, Las Vegas, NV, USA, 2003.

[10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[11] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[12] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[13] E. Olson, J.J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 2262–2269, 2006.

[14] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[C23] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, W. Burgard, and R. Siegwart. Towards mapping of cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

Towards Mapping of Cities

Patrick Pfaff* Rudolph Triebel*† Cyrill Stachniss†* Pierre Lamon† Wolfram Burgard* Roland Siegwart†

*University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany
†Eidgenössische Technische Hochschule (ETH), IRIS, ASL, 8092 Zurich, Switzerland

***Abstract*— Map learning is a fundamental task in mobile robotics because maps are required for a series of high level applications. In this paper, we address the problem of building maps of large-scale areas like villages or small cities. We present our modified car-like robot which we use to acquire the data about the environment. We introduce our localization system which is based on an information filter and is able to merge the information obtained by different sensors. We furthermore describe out mapping technique that is able to compactly model three-dimensional scenes and allows us efficient and accurate incremental map learning. We additionally apply a global optimization techniques in order to accurately close loops in the environment. Our approach has been implemented and deeply tested on a real car equipped with a series of sensors. Experiments described in this paper illustrate the accuracy and efficiency of the presented techniques.**

I. Introduction

Building models of the environment is a fundamental task of mobile robots since maps are needed for most high-level robotic applications. In the past, many researchers focused on the problem of learning maps and different techniques have been proposed [5], [10], [11], [18], [20]. Most of the proposed approaches focus on learning models for indoor environments like office spaces. Recently, several groups addressed the problem of learning two and three-dimensional models of outdoor scenes [6], [12], [13], [14], [22].

Since DARPA Grand Challenge [2], the usage of cars instead of classical mobile robots became popular in the research community [1], [23], [25]. Compared to standard robots, cars offer the possibility to travel longer distances, carry more sensors, and thus being more suitable for mapping large areas.

The contribution of this paper is an approach towards mapping of large-scale areas like villages or small cities. We describe our system to learn three-dimensional models of the environment. We apply probabilistic state estimation techniques as well as classification approaches to obtain these models. Our implementation uses a modified Smart car depicted in Figure 1 equipped with a series of sensors, ranging from proximity sensor, GPS, and an inertial measurement unit (IMU).

II. Related Work

The problem of learning models of the environment has been studied intensively in the past. In the literature, this problem is often referred to as simultaneous localization and mapping (SLAM). Most approaches to map learning generate two-dimensional models from range sensor data. A series of different approaches has been developed to address this problem [4], [5], [7], [9], [10], [11], [18]. Recently, several techniques for acquiring three-dimensional data with rotating 2d range scanners installed on a mobile robot have been developed [8], [26], [27]. Other authors have studied the acquisition of three-dimensional maps from vehicles that are assumed to operate on a flat surface. For example, Thrun *et al.* [21] present an approach that employs two 2d range scanners for constructing volumetric maps. Whereas the first is oriented horizontally and is used for localization, the second points towards the ceiling and is applied for acquiring 3d point clouds.

Fig. 1. The left image depicts the vertically mounted SICK LMS laser range finders which are rotated with constant speed by an electric step motor mounted under the lasers. The right image shows our robot. The robot is a standard Smart car. The model is a Smart fortwo coupé passion of the year 2005, which is equipped with a 45 kW engine.

A popular representation for $2\frac{1}{2}$-dimensional maps in robotics are elevation maps [17], [28]. In contrast to that, our approach learns a three dimensional model that can be regarded as an extention to elevation map which is able to store multiple layers for each grid cell [24]. This allows us to model structures like, e.g., bridges, underpasses, and trees in a more accurate way. We present our technique to match individual surface maps into a globally consistent model of the environment using a global error minimization approach. All techniques have been implemented and tested on a real car.

In the context of autonomous cars, a series of successful systems [1], [23], [25] have been developed due to DARPA Grand Challenge. As a result of this challenge, there exist autonomous cars that reliably avoid obstacles and navigate at comparably high speeds. The focus of the Grand Challenge

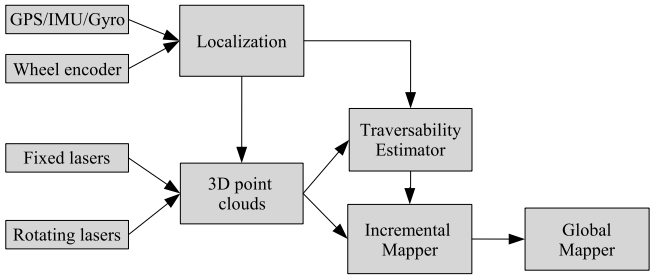


Fig. 2. The information flow between the individual modules.

was to finish the race as quickly as possible whereas certain issues like building consistent large-scale maps of the environment have been neglected since they where not needed for the race.

Even so our Smart car applied similar techniques than the winning vehicle Stanley [23] for following a specified trajectory, we have a different aim compared to the teams participating in the Grand Challenge. Our goal is to learn consistent and accurate three-dimensional models of the large-scale environments.

III. System Overview

Our instrumented car is equipped with a series of sensors. One group of sensor is used for localization. It consists of the inertial measurement unit, the differential GPS, the optical gyro, and the wheel encoders. The second group of sensors is given by the laser range finders. Three of them point to the front of the car and two are rotating on top of the roof of the car (see Figure 1).

Our software system is based on the modular inter-process communication (IPC) architecture. In this framework, each module can send and receive messages to/from other modules. The diagram in Figure 2 depicts the information flow between the most important modules.

IV. Localization

Our localization system applies the inverse form of the Kalman filter, i.e., the information filter. This filter has the property of summing information contributions from different sources in the update stage. This characteristic is advantageous when many sensors are involved which is the case in our application. The localization is done in two steps: the state prediction and the state update.

A. State Update

The localization algorithm estimates the state of the vehicle in a fixed navigation frame n which is represented by the north, west, and the altitude. The state vector contains the coordinates (x, y, z) of the vehicle and its three-dimensional orientation (roll ϕ, pitch θ and heading ψ to true north). We define the body frame b as the coordinate system attached to the vehicle. This frame is aligned with the vehicle kinematic axes (forward, left, and altitude) and its origin is placed at the center of the rear axle. The measurements models of the sensors are presented here.

• *Inertial measurement unit* (Crossbow NAV420): This unit provides sensor data with a frequency of 100 Hz that contains the measurements from 3 gyroscopes, 3 accelerometers, a 3D magnetic field sensor, and a GPS receiver. The internal digital signal processor of the unit combines the embedded sensors to provide the filtered orientation of the vehicle (roll, pitch, heading to true north) and the position (latitude, longitude, and altitude). This sensor, however, is not well adapted for ground vehicle driving at low speed. We therefore disabled the GPS and used the unit in angle mode: the unit outputs the filtered roll (ϕ_{imu}), pitch (θ_{imu}) and heading (ψ_{imu}) to magnetic north. This improves the pose estimate when driving at low speed. The measurement model for this sensor is

$$z_{imu} = \begin{bmatrix} \phi_{imu} \\ \theta_{imu} \end{bmatrix}_n = \begin{bmatrix} \phi \\ \theta \end{bmatrix}_n + v_{imu} \tag{1}$$

$$\psi_{imu} = \psi + b_{imu} + v_{himu}, \tag{2}$$

where v denotes the sensor noise and b_{imu} the offset between the heading to true north ψ and the heading measurement of the IMU. The bias b_{imu} is estimated by the filter using the heading measurements of the GPS which provides the heading to true north.

• *Car sensors*: The measurements taken by the car sensors are reported with a frequency of 100 Hz and are accessible via the CAN bus of the vehicle. The car provides the motor temperature, gas pedal position, steering wheel angle, wheel velocities, engine RPM, and some further status information. For localization, we use the velocity $\dot{x}_{odo}$ of the car from the CAN bus. Unlike a flight vehicle, the motion of a wheeled vehicle on the ground is governed by nonholonomic constraints. Under ideal conditions, there is no motion normal to the ground surface and no side slip: they can be written respectively as $\dot{z}_{odo} = 0$ and $\dot{y}_{odo} = 0$. In practice, these constraints are often violated. Thus, as in [3], we use zero mean Gaussian noise to model the extent of constraint violation. The measurement model for the odometry is then expressed as

$$z_{odo} = \begin{bmatrix} \dot{x}_{odo} \\ 0 \\ 0 \end{bmatrix}_b = \left[C_b^n\right]^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_n + v_{odo}, \tag{3}$$

where C_b^n is the matrix for transforming velocities expressed in the reference frame b of the car into the navigation frame n. The observation noise covariance is obtained using

$$R_{odo} = C_b^n \cdot diag\left\{\sigma_{enc}^2, \sigma_{vy}^2, \sigma_{vz}^2\right\}\left[C_b^n\right]^T, \tag{4}$$

where σ_{enc}^2 is the variance of the car velocity and σ_{vy}^2,σ_{vz}^2 are the amplitude of the noise related to the constraints.

• *Differential GPS system* (Omnistar Furgo 8300HP): This device provides the latitude, longitude, and altitude together with the corresponding standard deviation and the standard NMEA messages with a frequency of 5 Hz. In case the sensor receives the GPS drift correction signal, the unit changes automatically into the high precision GPS mode. When no correction signal is available, the device outputs

standard GPS information. We use the WGS-84 standard to convert the GPS coordinates in Cartesian coordinates (x, y, z) expressed in a local navigation frame n. The heading to true north ψ is also provided by that unit in the RMC message. The measurement model for the GPS is

$$z_{gps} = \begin{bmatrix} x_{gps} \\ y_{gps} \\ z_{gps} \\ \psi_{gps} \end{bmatrix}_n = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}_n + v_{gps}. \quad (5)$$

In order to reject the erroneous fixes caused by satellite constellation and multi-path change, we use the following gating function [19]

$$z^T(k) \cdot S^{-1} \cdot z(k) \leq \gamma, \quad (6)$$

where S is the innovation covariance of the observation. The value of γ is set to reject innovations exceeding the 95% threshold.

• *Optical gyroscope* (KVH DSP3000): This fiber optic gyroscope can measure very low rotation rates with a frequency of 100 Hz. It is possible to use it as a heading sensor for a comparably long period of time by integrating the angular rate (the unit provides the integrated angle). Contrary to compasses, the integrated heading is not sensitive to earth magnetic field disturbances. Finally, this unit offers much better accuracy than mechanical gyro and is not sensitive to shocks because it contains no moving parts. The measurement model for the optical gyro is

$$z_{opt} = \psi_{opt} = \psi + b_{opt} + v_{opt}, \quad (7)$$

where b_{opt} is the angular offset between the heading to true north ψ and the actual measurement of the gyro.

B. Prediction model

We apply a standard prediction model for the car which has the following form

$$\mathbf{x}_{k+1} = \begin{bmatrix} F_x & \dots & & 0 \\ \vdots & F_y & & \\ & & F_z & \vdots \\ 0 & & \dots & I_{5x5} \end{bmatrix} \cdot \mathbf{x}_k + w_k. \quad (8)$$

The state vector $\mathbf{x}$ contains the position and velocity expressed in the navigation frame n, the orientation of the vehicle represented by the three angles roll ϕ, pitch θ, yaw ψ, and the two biases b_{imu} and b_{opt}:

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \phi & \theta & \psi & b_{imu} & b_{opt} \end{bmatrix}^T \quad (9)$$

The position of the vehicle at time $k+1$ is predicted using the position and velocity at time k. This takes the form of a first order process written as

$$F_{x,y,z} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}_k, \quad (10)$$

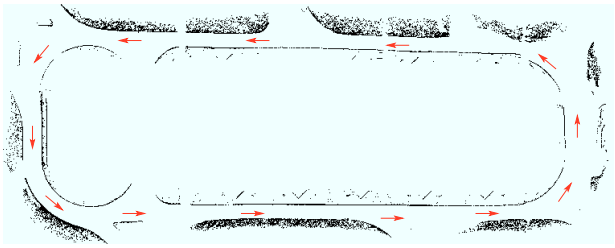


Fig. 3. Obtained traversability map using the fixed sick laser range finders. Black refers to non traversable cells and the red/gray arrows illustrate the trajectory taken by the car.

where T denotes the sampling period (10 ms). All the other elements of the state vector are predicted as simple Gaussian processes. The covariance matrix Q_k associated to the state prediction process is represented as

$$Q_k = G_k \cdot q_k \cdot G_k^T, \quad (11)$$

where q_k is a diagonal matrix containing the variances of the individual elements of the state vector

$$q_x = diag \left\{ \begin{array}{ccccccccc} \sigma_x^2 & \sigma_y^2 & \sigma_z^2 & \sigma_\phi^2 & \sigma_\theta^2 & \sigma_\psi^2 & \sigma_{b_{imu}}^2 & \sigma_{b_{opt}}^2 \end{array} \right\}. \quad (12)$$

Finally, the matrix mapping the noise covariance q_k to the process covariance Q_k is written as

$$G_k = \begin{bmatrix} g_x & \dots & & 0 \\ \vdots & g_y & & \vdots \\ & & g_z & \\ 0 & & \dots & diag_{5x5}(T) \end{bmatrix}_k, \quad (13)$$

where

$$g_{x,y,z} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}. \quad (14)$$

All in all, this information filter framework allows us to robustly and efficiently integrate the information from the different sensor into a pose estimate of the car. The pose information is provided with a high frequency and with small delays only. This is important for online control of the car.

V. Traversability Estimation

Whenever driving with a robot car, a central issue is to identify the obstacle-free terrain. Without a reliable estimation of the traversable area, autonomous car driving is nearly impossible. This paper does not focus on autonomous navigation, the estimation of the traversability, however, is regarded as a mapping task and therefore also addressed in this work.

The car is equipped with five SICK laser range finders whereas two are mounted on a rotating unit and three are fixed (compare Figure 1). We currently use the three static laser range finders in order to estimate the traversability of the area in front of the car. Given a laser range observation, we first compute the end points of the individual beams.

We then add the 3d points to the cells of a local two-dimensional grid map according to the x, y-coordinate of the beam. We then parse the cells and compute the mean and variance of the z-values for each cell. The decision if a cell is locally traversable can be done based on these two values. When adding the data of multiple laser range finders into a single grid, it is likely to get a series of obstacles at locations where actually no obstacle is located. This phenomenon is also described by Thrun *et al.* [23] as phantom obstacles. These phantom obstacles are caused by small errors in the pitch estimate of the location of the car, between the individual laser range scans. Therefore, we compute the traversability estimate individually for each scan and merged the independently estimated traversability values into a common grid structure. We found that this yields good results when moving on streets as well as on unpaved roads and avoids phantom obstacles. An example for a resulting traversability estimate is shown in Figure 3.

VI. Mapping

During the mapping process, we create globally consistent maps using the inputs of the localization module and the five laser range finders mounted on the robot. We use multi-level surface maps (MLS maps) as proposed in our previous work [24]. MLS maps store in each cell of a discrete grid the height of the surface in the corresponding area. In contrast to elevation maps, MLS maps allow us to store multiple surfaces in each cell of the grid. In the remainder of this paper, these surfaces a referred to as patches. This representation enables a mobile robot to model environments with structures like bridges, underpasses, buildings or mines. Additionally, they enable the robot to represent vertical structures.

The localization technique described in Section IV works well for navigation issues. However, applying mapping with known poses based on this pose estimate usually results in globally inconsistent maps. In practice, this typically becomes apparent when the robot encounters a loop, i.e., when it returns to a previously visited place. To achieve the goal of globally consistent maps it is needed to associate the data which is acquired when the robot reaches the same place of the environment at different times. To achieve this, we build local MLS maps and apply the ICP algorithm to iteratively find constraints between poses and to solve this data association problem. This is described in detail in the reminder of this section. After the map matching and loop closing process the local MLS maps can be merged to one global consistent MLS map.

A. Data Acquisition and Local Map Building

During the data acquisition process, we collect three-dimensional points which corresponds directly to the sensed environment. The data is collected while our robot is moving continously through the environment using the five SICK laser range finders. As explained before, three of them are mounted in a fixed position and provide data points about the environment in front of our robot. Additionally, we mounted two laser range finders in vertical direction on a rotating plate on the top of the car. Figure 1 depicts the two lasers and the electric step motor. During data acquisition the step motor rotates the two laser range finders with a constant frequence of 0.37 Hz. Due to this configuration the rotating lasers provide data points which correspond to the environment in all directions around the robot. To build a local MLS map, we now use the data points acquired during a complete 360 degree turn by the rotating lasers. This setup is well-suited to build 3d maps of the environment. Furthermore, we add the data points which are acquired with the three fixed lasers during this period of time. Figure 4 depicts an example of a single local MLS map. From this point on, we discard the point clouds and perform all computations based on the local MLS maps. The example shows a typical scene of an urban environment with street lights and trees. Note that the data of all five SICK laser range finders are used for mapping. For estimating the traversable area in front of the car, however, only the three static sensor are used due to the comparable slow rotation of the rotating laser sensors.

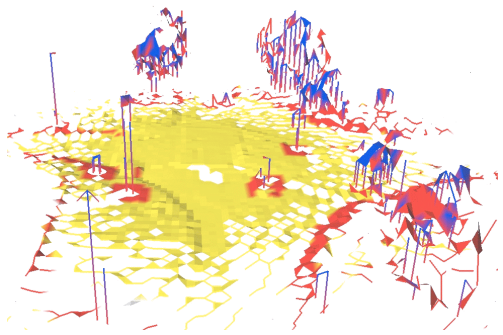


Fig. 4. Example of a single local MLS map. The example shows a typical scene of an urban environment with street lights and trees.

B. Map Matching

In addition to the traversability analysis described in Section V, we can identify vertical objects based on the 3d data. As a result, every patch in the MLS map is labeled as 'traversable', 'non-traversable', and 'vertical'. The labels are used in the ICP-based map matching process to obtain a more robust and accurate registration.

ICP seeks to find a rotation matrix R and a translation vector $\mathbf{t}$ that minimizes an error function computed based on the two maps we aim to match. We integrate the labels of the individual patches into the ICP error function in order to improve the matching result. We only consider matches between patches of the same label.

Let $\mathbf{u}$ be the vertical patches, $\mathbf{v}$ the traversable, and $\mathbf{w}$ the non-traversable ones of the first map. The cells of the second map are indicated by primed variables. We can define the following error function:

$$e(R,\mathbf{t}) = \underbrace{\sum_{c=1}^{C_1} d(\mathbf{u}_{i_c}, \mathbf{u}'_{j_c})}_{\text{vertical objects}} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{v}_{i_c}, \mathbf{v}'_{j_c})}_{\text{traversable}} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{w}_{i_c}, \mathbf{w}'_{j_c})}_{\text{non-traversable}}. \quad (15)$$

In this equation, d is the Mahalanobis distance and the indices i_c and j_c indicate the correspondence between the patches. Minimizing $e(R, \mathbf{t})$ as well as the computation of the correspondences is iterated within the ICP algorithm.

In practical experiments [16], we found that matching only patches with the same label leads to more robust and accurate map estimates. Furthermore, the ICP algorithm converges faster due to the smaller number of potential correspondences.

C. Loop Closing

The ICP-based scan matching technique described above works well for the registratering robot poses into one global reference frame. However, the individual scan matching processes result in small residual errors which accumulate over time and usually result in globally inconsistent maps. In practice, this typically becomes apparent when the robot encounters a loop, i.e., when it returns to a previously visited place. Accordingly, techniques for calculating globally consistent maps are necessary. Therefore, we apply an approach that combines the ideas of Lu and Milios [10] and Olson's algorithm [15] to globally correct the map. This approach applies error minimization via stochastic gradient descent on the whole vector of poses and yields accurate map estimates given a set of constraints between poses.

VII. Experiments

A. Localization

Our localization system has been extensively tested and provides accurate pose estimates in a robust manner when moving though urban environments. A typical result obtained with our smart car is depicted in Figure 5. The figure represents the estimated trajectory of the car overlayed on the ortho-photo of the EPFL campus.

Fig. 5. Overlay of the estimated trajectory and the ortho-photo of the EPFL campus. The zones where the GPS was not available are highlighted. The total traveled distance is around 2300 m. The labels (a), (b), and (c) identify areas which are later on referred to by Figure 6 and 7.

During the experiment, the car drove in areas where the GPS quality was bad or not available, for example along narrow alleys bordered with trees, close to buildings, or in an underground parking lot. However, the localization algorithm was able to cope with GPS faults and provided accurate positioning estimation, such as depicted in Figure 6.

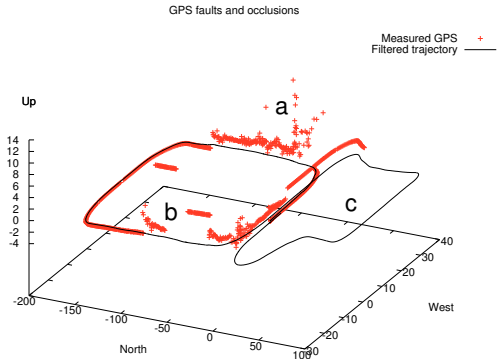


Fig. 6. This graph represents a part of the trajectory depicted in Figure 5. In this urban environment, the GPS signal is disturbed by many objects (trees, buildings, etc.) and GPS faults are of high amplitude (several meters in the horizontal plane and up to 16 m vertically). The localization algorithm was able to reject erroneous GPS fixes and to provide accurate estimations. The labels a,b and c mark areas where GPS is of poor quality (a), (b) or unavailable (c).

The uncertainty associated to the pose estimation mainly depends on the quality of the GPS fixes. As depicted in Figure 7, the standard deviation is low when differential GPS is available (~3 cm) but increases as soon as fixes are unavailable (up to 60 cm).

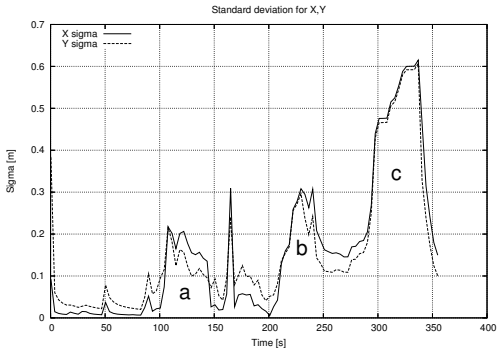


Fig. 7. Standard deviation along the north (x) and west axis (y) for the trajectory depicted in Figure 6. The standard deviation increases when GPS quality is poor and decreases as soon as it gets better. The labels (a), (b) and (c) corresponds to the zones marked in Figure 5.

B. Mapping

To acquire the data, we steered our robotic car depicted in Figure 1 over streets of the EPFL campus. The goal of these experiments is to demonstrate that our representation yields a significant reduction of the memory requirements compared to a point cloud representation, while still providing highly accurate maps. Additionally, they show that our representation is well-suited for global pose estimation and loop closure. Furthermore, the experiments show the necessity of the loop closing procedure. Figure 8 show the resulting map of a dataset acquired along a 2.3 km trajectory. Figure 9 shows a cutout of two MLS maps from that dataset. The left image depicts the resulting MLS Map when only

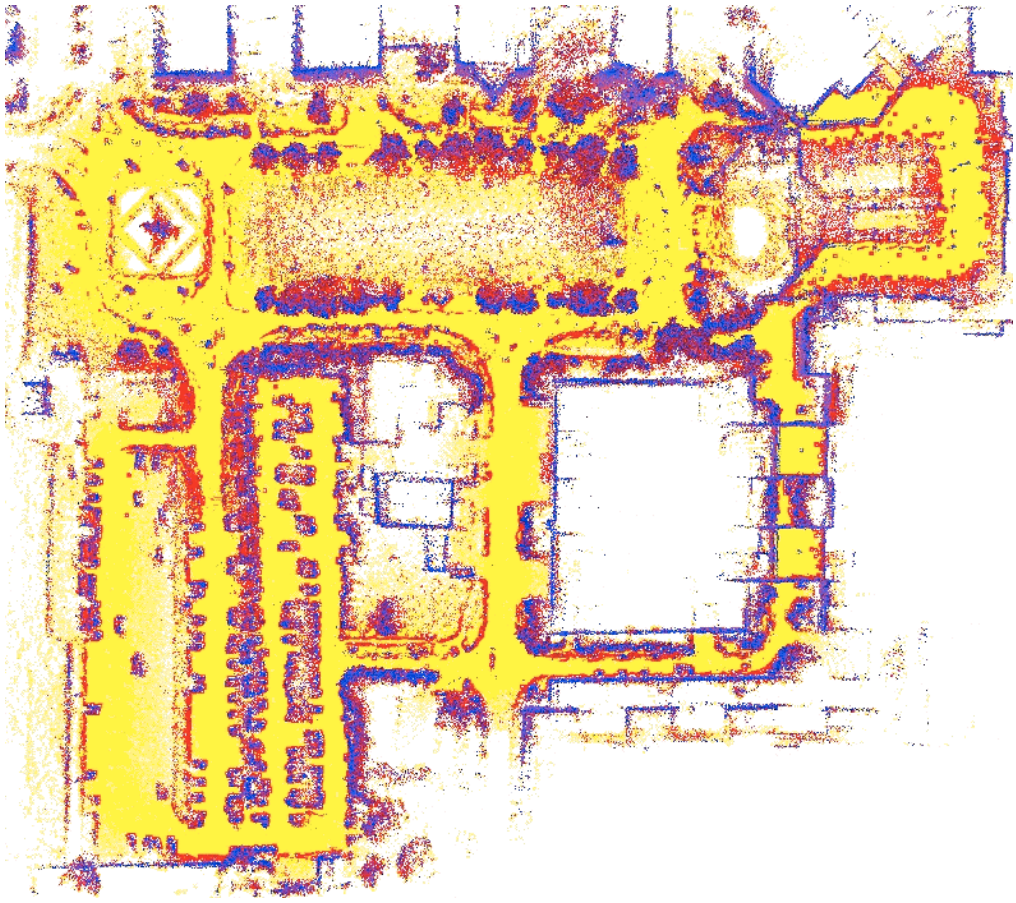


Fig. 8. Top view of the resulting MLS map with a cell size of 50cm x 50cm. The yellow/light gray surface patches are classified as traversable.The area scanned by the robot spans approximately 300 by 250 meters. During the data acquisition, the robot traversed five nested loops with a length of approximately 2,300m.

local map matching is applied. The right image displays the same part of the MLS map where we additionally applied our loop closing algorithm.

In this experiment, we acquired 374 local point clouds consisting of 68,162,000 data points. The area scanned by the robot spans approximately 300 by 250 meters. During the data acquisition, the robot traversed five nested loops with a length of approximately 2,300m. Figure 8 shows a top view of the resulting MLS map with a cell size of 50cm x 50cm. The yellow/light gray surface patches are classified as traversable. It requires 55 MB to store the computed map, where 34% of 300,000 cells are occupied. Compared to this the storage of the 68,162,000 data points requires 1,635 MB. The scan matching between the local MLS maps has been computed online during the data acquisition on a 2GHz dual core laptop computer. The loop closing step of our mapping algorithm is computed offline when the robot finished the data acquisition. In our current approach, the computation time for the optimization of the shown data set is approximately 15 minutes.

VIII. Conclusion

In this paper, we presented our approach towards mapping of large-scale areas like villages or cities. We presented the setup of our modified car and the techniques applied to learn accurate models of the environment and localize the vehicle in the world. Our map representation can be seen as an extention of elevation maps which are able to store different surfaces in the environment. In order to learn these maps, we present our pose estimation technique as well as an approach to match sub-maps in order to correct the poses based on the proximity sensors. In order to accurately close loops, we apply a least square minimization approach. As a result, we obtain high quality three-dimensional models. All techniques have been implemented and tested using a real car equipped with different types of sensors. The experiments presented in this paper, show the result of real world data obtained with this robot.

Acknowledgment

This work has partly been supported by the EC under contract number FP6-IST-027140-BACS, FP6-2005-IST-5-muFly, and by the German Science Foundation (DFG) within the Research Training Group 1103 and under contract number SFB/TR-8. The authors would like to thank Sascha Kolksi, Frederic Pont, and all other members of the *SMART-Team* who contributed to this work.

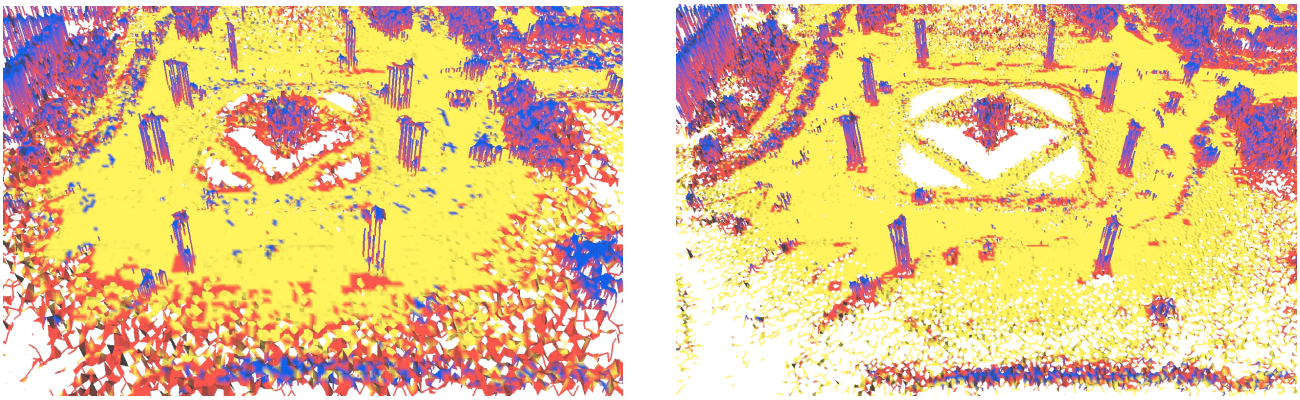


Fig. 9. This figure depicts the lower left corner of the MLS Map shown in Figure 8. The left image illustrates the resulting MLS Map when only local map matching is applied. The right image displays the same part of the MLS map where we additionally applied our loop closing algorithm. The inconsistencies can be seen by the vertical poles in the figure. Furthermore, several traversable patches have been misclassified as non traversable (red/dark gray) due to the misalignment of the maps.

References

[1] L.B. Cremean, T.B. Foote, J.H. Gillula, G.H. Hines, D. Kogan, K.L. Kriechbaum, J.C. Lamb, J. Leibs, L. Lindzey, C.E. Rasmussen, A.D. Stewart, J.W. Burdick, and R.M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*, 2006. Submitted for publication.

[2] DARPA. Darpa grand challenge rulebook. Website, 2004. http://www.darpa.mil/grandchallenge05/Rules 8oct04.pdf.

[3] G. Dissanayake, S. Sukkarieh, and H. Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, 17(5), 2001.

[4] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[5] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[6] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60:5–24, 2004.

[7] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2443–2448, Barcelona, Spain, 2005.

[8] P. Kohlhepp, M. Walther, and P. Steinhaus. Schritthaltende 3D-Kartierung und Lokalisierung für mobile inspektionsroboter. In *18. Fachgespräche AMS*, 2003. In German.

[9] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4), 1991.

[10] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

[11] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

[12] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[14] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[15] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[16] Pfaff P. and Burgard W. An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, pages 165–176, Port Douglas, QLD, Australia, 2005.

[17] S. Singh and A. Kelly. Robot planning in the space of feasible actions: Two examples. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1996.

[18] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[19] S. Sukkarieh, E.M. Nebot, and H. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle application. *IEEE Transactions on Robotics and Automation*, 15(3), 1999.

[20] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[21] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

[22] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[23] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. To appear.

[24] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[25] C. Urmson. *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2005.

[26] J. Weingarten and R. Siegwart. 3d slam using planar segments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[27] O. Wulf, K-A. Arras, H.I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4204–4209, New Orleans, 2004.

[28] C. Ye and J. Borenstein. A new terrain mapping method for mobile robot obstacle negotiation. In *Proc. of the UGV Technology Conference at the 2002 SPIE AeroSense Symposium*, 1994.

[C24] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Speeding-up rao-blackwellized slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 442–447, Orlando, FL, USA, 2006.

Speeding Up Rao-Blackwellized SLAM

Giorgio Grisetti*† Gian Diego Tipaldi† Cyrill Stachniss* Wolfram Burgard* Daniele Nardi†

*University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

†Dipartimento Informatica e Sistemistica, Universitá "La Sapienza", I-00198 Rome, Italy

***Abstract*— Recently, Rao-Blackwellized particle filters have become a popular tool to solve the simultaneous localization and mapping problem. This technique applies a particle filter in which each particle carries an individual map of the environment. Accordingly, a key issue is to reduce the number of particles and/or to make use of compact map representations. This paper presents an approximative but highly efficient approach to mapping with Rao-Blackwellized particle filters. Moreover, it provides a compact map model. A key advantage is that the individual particles can share large parts of the model of the environment. Furthermore, they are able to re-use an already computed proposal distribution. Both techniques substantially speed up the overall process and reduce the memory requirements. Experimental results obtained with mobile robots in large-scale indoor environments and based on published, standard datasets illustrate the advantages of our methods over previous Rao-Blackwellized mapping approaches.**

I. Introduction

Learning maps is a fundamental task of mobile robots and a lot of researchers focused on this problem. In the literature, the mobile robot mapping problem is often referred to as the *simultaneous localization and mapping (SLAM)* problem [3, 7, 8, 9, 13, 14, 15, 20]. In general, SLAM is a complex problem because for learning a map the robot requires a good pose estimate while at the same time a consistent map is needed to localize a robot. This dependency between the pose and the map estimate makes the SLAM problem hard and requires to search for a solution in a high-dimensional space.

Murphy, Doucet, and colleagues [15, 2] introduced Rao-Blackwellized particle filters (RBPFs) as an effective means to solve the SLAM problem. The main problem of the Rao-Blackwellized approaches is their complexity, measured in terms of the number of particles required to learn an accurate map. Reducing this quantity is one of the major challenges for this family of algorithms.

The contribution of this paper is a technique that reduces the computational and the memory requirements in the context of Rao-Blackwellized mapping. In this way, it becomes feasible to maintain a comparably large set of particles online. This is achieved by enabling a subset of samples to share large parts of the map and to use the same proposal distribution. Our system allows a standard laptop computer to perform all computations necessary to learn accurate maps with more than one thousand samples online.

This paper is organized as follows. After the discussion of related work, we briefly introduce Rao-Blackwellized mapping. We then describe our technique for efficiently drawing particles from a proposal distribution. After this, we present our map representation. Finally, we show experiments illustrating the improvements of our approach to Rao-Blackwellized mapping.

II. Related Work

Solutions to the SLAM problem can be classified according to their underlying estimation technique. The most popular approaches are Extended Kalman filters (EKFs), maximum likelihood techniques, sparse extended information filters (SEIFs), and Rao Blackwellized particle filters (RBPFs). The effectiveness of the EKF comes from the fact that it estimates the fully correlated posterior over landmark positions and robot poses [10, 17]. Its weakness lies in the strong assumptions regarding the motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely identifiable. There exist techniques [16] to deal with unknown data association in the SLAM context. However, if certain assumptions are violated the filter is likely to diverge [6].

An alternative approach is to use a maximum likelihood algorithm that computes a map by constructing a network of relations. The relations represent the spatial constraints between the poses of the robot [8, 12].

Thrun *et al.* [20] proposed a SEIF method which uses the inverse of the covariance matrix. In this way, measurements can be integrated efficiently. Eustice *et al.* [5] presented an improved technique to accurately compute the error-bounds within the SEIF framework and thus reduces the risk of becoming overly confident.

In [15, 2], Rao-Blackwellized particle filters have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a potential trajectory of the robot and a map of the environment. The framework has been subsequently extended by Montemerlo *et al.* [13, 14] for approaching the SLAM problem with landmarks. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [3] and Hähnel *et al.* [9]. Whereas the first work describes an efficient map representation, the second one presents an improved motion model that reduces the number of required particles. A combination of the approach of Hähnel *et al.* and Montemerlo *et al.* as been presented by Grisetti *et al.* [7], which extends the ideas of FastSLAM-2 to the grid map case. We present in this paper an approximative solution to Rao-Blackwellized mapping which describes how to draw particles and how to represent maps so that the system can be executed significantly faster and needs less memory resources.

Lisien *et al.* [11] realized an hierarchical map model in the context of SLAM and reported that this improves loop-closing. Bosse *et al.* [1] describe a generic framework for

SLAM in large-scale environments. They use a graph structure of local maps with relative coordinate frames similar to the work described in [4]. This approach is able to reduce the complexity and at the same time it can better deal with linearization problems in the context of EKF-SLAM. Our approach is related to this framework since we also use local maps attached to a graph structure to model the environment. However, our motivation to use such a map representation is to allow multiple particles to share a map.

The contribution of the paper is a computational and memory efficient Rao-Blackwellized particle filter for SLAM. Our approach allows the robot to efficiently determine the proposal distributions to sample the next generation of particles in an approximative manner. Additionally, we present a compact map model in which multiple particles share a map. This enables us to maintain substantially more samples with less memory and computational requirements compared to state-of-the-art Rao-Blackwellized mapping approaches.

III. Rao-Blackwellized Mapping

RBPFs for SLAM are used to estimate the posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the trajectory $x_{1:t}$ of the robot and the map m of the environment given the observations $z_{1:t}$ and odometry measurements $u_{1:t-1}$. Its key idea is to separate the estimation of the trajectory of the robot from the map estimation process

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) p(x_{1:t} \mid z_{1:t}, u_{1:t-1}). \quad (1)$$

This can be done efficiently, since the posterior over maps $p(m \mid x_{1:t}, z_{1:t})$ can be computed analytically given the knowledge of $x_{1:t}$ and $z_{1:t}$. Computing the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ is similar to the localization problem, since only the trajectory of the vehicle needs to be estimated. This is done using a particle filter which incrementally processes the observations and the odometry readings. The overall process can be summarized by the following four steps:

1) *Sampling*: The next generation of particles is obtained from the current generation by sampling from a so-called proposal distribution.
2) *Importance Weighting*: An individual importance weight is assigned to each particle according to the most recent observation, the pose estimate, and the map associated with this particle.
3) *Resampling*: Particles with a low importance weight are typically replaced by samples with a high weight. This step is necessary since only a finite number of particles is used to approximate a continuous distribution.
4) *Map Estimation*: The map of each particle is updated based on pose represented by that particle.

Several authors proposed optimizations to Rao-Blackwellized mapping. They either presented compact map representations [3] to deal with large particle sets or accurate proposal distributions [7, 9, 13] in order to keep the number of samples small.

IV. Speeding Up Rao-Blackwellized Mapping

In this section, we present our approach to Rao-Blackwellized mapping which is able to handle large particle sets while reducing the memory and computational requirements. Our implementation is based on the open-source implementation [18] of the mapping system of Grisetti *et al.* [7]. The mayor drawback of this approach lies in its complexity. It runs online only for small particle sets. This is due to an informed but expensive to compute proposal distribution which is determined for each particle individually. Furthermore, each particle maintains a full grid map.

In the context of Rao-Blackwellized particle filters for SLAM, the proposal is used to model the relative movement of the vehicle under uncertainty. In most situations, this uncertainty is similar for all samples within one movement. It therefore makes sense to use the same uncertainty to propagate the particles as long as they appear to represent similar state hypotheses. In this section, we derive a way to sample multiple particles from the same proposal. As a result, the time consuming computation of the proposal distribution can be carried out for a few particles that are representatives for groups of similar samples.

Furthermore, local maps which are represented in a robot-centered coordinate frame look similar for many particles. We therefore present a compact map model in which multiple particles can share a map. Instead of storing an individual map, each sample maintains only a set of reference frames for the different local maps. This substantially reduces the memory requirements of the mapping algorithm.

A. Different Situations During Mapping

Before we derive our new proposal distributions, we start with a brief analysis of the behavior of a RBPF. One can distinguish three different types of situations during mapping:

- The robot is moving through *unknown* areas,
- is moving through *well-known* areas, or
- is *closing a loop*.

In each of those situations, the filter behaves differently. Whenever the robot is moving through unknown terrain, the trajectory uncertainty grows. This is due to the fact that the errors are accumulated along the trajectory. The resulting uncertainty can only be bounded by observations which cover a (partially) known region.

In the second case, a map of the surroundings of the robot is known and in this way the SLAM problem turns into a localization problem which is typically easier to handle. Whenever the robot is closing a loop, the particle cloud is often widely spread. By reentering known areas, the filter can typically determine which particles are consistent with their own map and which are not. Such a situation leads to an unbalanced distribution of particle weights. The next resampling action then eliminates a series of unlikely hypotheses.

For each of these three situations, we will present a proposal distribution that needs to be computed only for a small set of representatives rather than for all particles. For the beginning, let us assume that

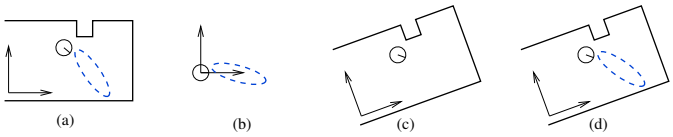


Fig. 1. Image (a) depicts the pose hypothesis of a particle, its local map, and the computed proposal which represented by the blue/dashed ellipse. Image (b) illustrates the proposal distribution represented in the ego-centric reference frame of that sample. Image (c) shows a second particle and its map. By carrying out a coordinate transform, the proposal of the first particle can be used by the second particle as long as their maps are locally similar (d).

1) the current situation is known, which means that the robot can determine whether it is moving through unknown terrain, within a known area, or is closing a loop,
2) the corresponding local maps of two samples are similar if considered in a particle-centered reference frame. In the following, we refer to this property as *local similarity* of the maps,
3) an accurate algorithm for pose tracking is used and the observations are affected by a limited sensor noise.

B. Computing the Proposal for Unknown Terrain

When moving through unknown areas, most parts of the map are irrelevant for computing the proposal distribution. Only a local map around the current pose is therefore taken into account. This map, called $\tilde{m}^{(i)}_{t-1}$, refers to the local map of particle i with respect to the pose $x^{(i)}_{t-1}$ of that particle at time step $t-1$. In the surroundings of the robot, we can approximate

$$p(x_t \mid m^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}) \simeq p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}). \quad (2)$$

Under Assumption 2, which requires that the maps of particle i and j are locally similar, we can write

$$\tilde{m}^{(i)}_{t-1} \ominus x^{(i)}_{t-1} \simeq \tilde{m}^{(j)}_{t-1} \ominus x^{(j)}_{t-1}. \quad (3)$$

Here $\oplus$ and $\ominus$ are the standard pose compounding operators (see [12]). E.g., $a \ominus b$ is an operator that translates all the points in the domain of the function a so that the new origin of the domain of a is b and $\oplus$ is its inverse.

We observed that the proposal distributions for different particles are similar if transformed to an ego-centric reference frame

$$\begin{aligned} &p(x_t \mid \tilde{m}^{(j)}_{t-1}, x^{(j)}_{t-1}, z_t, u_{t-1}) \ominus x^{(j)}_{t-1} \\ &\quad \simeq p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}) \ominus x^{(i)}_{t-1}. \end{aligned} \quad (4)$$

As a result, we can determine the proposal for a particle j by computing the proposal in the reference frame of particle i and translating it to the reference frame of particle j

$$\begin{aligned} &p(x_t \mid \tilde{m}^{(j)}_{t-1}, x^{(j)}_{t-1}, z_t, u_{t-1}) \\ &\quad \simeq x^{(j)}_{t-1} \oplus (p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}) \ominus x^{(i)}_{t-1}). \end{aligned} \quad (5)$$

This computation is illustrated in Figure 1. Eq. (5) shows how transform a proposal between particles while the robot moves through unknown terrain. The complex proposal computation needs to be performed only once and can then be translated to the reference frame of the other particles.

C. Computing the Proposal for Already Visited Areas

Whenever the robot moves through known areas, each particle stays localized in its own map according to Assumption 3. To update the new pose of each particle while the robot moves, we maximize the likelihood of the observation around the pose predicted by odometry

$$x^{(i)}_t = \operatorname*{argmax}_{x_t} p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}). \quad (6)$$

Analog to Eq. (3)-(5), we can express the proposal of particle j using the one of particle i. The only difference is that we do not apply the $\oplus$ and $\ominus$ operators based on the poses of the samples. Instead, the operators are applied based on the particle dependent reference frames $l^{(i)}$ and $l^{(j)}$ of the local maps. These reference frames were established when previously mapping the terrain. This results in

$$\begin{aligned} &p(x_t \mid \tilde{m}^{(j)}_{t-1}, x^{(j)}_{t-1}, z_t, u_{t-1}) \\ &\quad \simeq l^{(j)} \oplus (p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}) \ominus l^{(i)}). \end{aligned} \quad (7)$$

Combining Eq. (6) and Eq. (7) leads to

$$x^{(j)}_t = \operatorname*{argmax}_{x_t} p(x_t \mid \tilde{m}^{(j)}_{t-1}, x^{(j)}_{t-1}, z_t, u_{t-1}) \quad (8)$$

$$\simeq l^{(j)} \oplus \underbrace{(\operatorname*{argmax}_{x_t} p(x_t \mid \tilde{m}^{(i)}_{t-1}, x^{(i)}_{t-1}, z_t, u_{t-1}) \ominus l^{(i)})}_{x^{(i)}_t} \quad (9)$$

$$= l^{(j)} \oplus (x^{(i)}_t \ominus l^{(i)}). \quad (10)$$

Under the Assumptions 2 and 3, we can estimate the poses of all samples according to Eq. (10). In this way, the complex computation of an informed proposal needs to be done only once. When the robot is in one of the two situations described above, the computation of the importance weights is done as proposed in [7] except that we have to evaluate the weights only once.

D. Computing the Proposal When Closing a Loop

In contrast to the two situations described before, the computation of the proposal is more complex in case of a loop-closure. This is due to the fact that Assumption 2 (local similarity) is typically violated even for subsets of particles. This fact can be illustrated by supposing a widely spread cloud of particles when closing a loop. The different samples re-enter the previously mapped terrain at different locations. This results in different hypotheses about the topology of the environment and definitively violates Assumption 2. Dealing with such a situation, requires additional effort in the estimation process.

Let us start with the informed proposal considering all sensor observations $z_{1:t}$ and the most recent odometry reading u_{t-1}. The proposal can be factorized as

$$\begin{aligned} &p(x_t \mid z_{1:t}, x^{(i)}_{1:t-1}, u_{t-1}) \\ &\quad = \eta p(z_t \mid z_{1:t-1}, x^{(i)}_{1:t-1}) p(x_t \mid x^{(i)}_{t-1}, u_{t-1}) \quad (11) \\ &\quad = \eta p(z_t \mid x_t, m^{(i)}_{t-1}) p(x_t \mid x^{(i)}_{t-1}, u_{t-1}), \quad (12) \end{aligned}$$

where η is a normalizer resulting from Bayes' rule.

Whenever a particle i closes a loop, we consider that its map $m^{(i)}_{t-1}$ consists of two components. The first one is a local map $m^{(i)}_{\text{local}}$, which has no overlap with the previously seen area and does not affect the loop closure. Secondly, a loop map $m^{(i)}_{\text{loop}}$ which models a previously mapped part of the environment re-visited after moving through unknown terrain for a long period of time.

$$p(z_t \mid x_t, m^{(i)}_{t-1}) \quad = \quad p(z_t \mid x_t, m^{(i)}_{\text{local}}, m^{(i)}_{\text{loop}}) \tag{13}$$

Under the assumption that these two maps are disjoint, it is possible to choose a likelihood function that allows us to apply the following factorization

$$p(z_t \mid x_t, m^{(i)}_{\text{local}}, m^{(i)}_{\text{loop}}) \;\propto\; p(z_t \mid x_t, m^{(i)}_{\text{local}}) p(z_t \mid x_t, m^{(i)}_{\text{loop}}) \tag{14}$$

Notice that the computation of the proposal in case of a loop-closure is more expensive than in the two other situations. Fortunately, loop-closing situations occur rarely. The robot has to travel through unknown and eventually known terrain for a comparably long period of time before a loop-closure can occur.

According to the importance sampling principle, the particle weights are given by

$$w^{(i)}_t = w^{(i)}_{t-1} \frac{p(x^{(i)}_t \mid z_t, x^{(i)}_{t-1}, m^{(i)}_{\text{local}}, m^{(i)}_{\text{loop}}, u_{t-1})}{p(x^{(i)}_t \mid z_t, x^{(i)}_{t-1}, m^{(i)}_{\text{local}}, u_{t-1})} \tag{15}$$

$$= w^{(i)}_{t-1} \frac{\eta^{(i)}_1 p(z_t \mid x^{(i)}_t, m^{(i)}_{\text{local}}) p(z_t \mid x^{(i)}_t, m^{(i)}_{\text{loop}})}{\eta^{(i)}_2 p(z_t \mid x^{(i)}_t, m^{(i)}_{\text{local}})} \tag{16}$$

$$= w^{(i)}_{t-1} p(z_t \mid x^{(i)}_t, m^{(i)}_{\text{loop}}) \frac{\eta^{(i)}_1}{\eta^{(i)}_2}, \tag{17}$$

where η_1 and η_2 are normalization factors resulting from Bayes' rule.

E. Approximative Importance Weight Computation

Eq. (17) tells us how to update the particle weights in case of a loop closure. Unfortunately, the computation of the normalizing factors η_1 and η_2 cannot be done efficiently. Therefore, in our current implementation, the weights are evaluated according to the raw observation model based on the loop map m_{loop}

$$w^{(i)}_t \simeq w^{(i)}_{t-1} p(z_t | x^{(i)}, m^{(i)}_{\text{loop}}) \tag{18}$$

rather than according to Eq. (17). This means that we ignore the ratio of the normalizing factors η_1 and η_2 and approximate the importance weights when closing a loop. This is significantly faster to compute and as we will demonstrate in the experiments, the approximation error is small.

V. Achieving Situation Estimation, Local Similarity, and Pose Tracking

All of the derivations made in the previous section require the robot to know whether it is moving through unknown terrain, through a previously mapped area, or is currently closing a loop (Assumption 1). Here, we describe how to distinguish the different cases. Detecting the first two situations

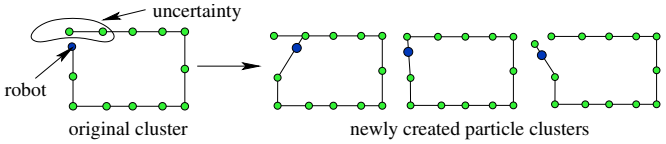


Fig. 2. The left image depicts a cluster while the robot is approaching a loop-closure. The shown particle cluster splits up into three different clusters (topology hypotheses) as depicted in the right image.

can be done in a straightforward way by comparing the area covered by the current observation given the particle pose and the map constructed so far.

More difficult is to decide whether or not the robot is closing a loop. To make this decision, we apply the approach proposed by Stachniss *et al.* [19] in the context of exploration with active loop-closing. This approach uses a dual representation consisting of a grid map and a topologic map that models the trajectory of the vehicle. By comparing both representations, one is able to accurately determine whether or not a robot is closing a loop.

Assumption 2 (local similarity) typically holds only up to the first loop closure but is then violated. By explicitly modeling the different topological hypotheses, it is still possible to represent the posterior in an appropriate way. To achieve local similarity, we introduce the notation of a *particle cluster* which describes a subset of particles for which the assumption of local similarity between maps holds. Ambiguities in the posterior can then be modeled using multiple particle clusters. Such clusters are obtained by grouping similar samples so that the maps within one cluster represent the same topology.

In the following, we explain how to represent such a set of samples and how to split up a particle cluster in case the assumption of local similarity is violated.

In our current system, we represent a map as a set of local maps also called patches. A global map for a given particle can be obtained by specifying the location of each patch within a global reference frame. Each sample therefore has to store only a list of reference frames $l^{(i)}_n$ for the patches. In this way, the individual patches $\mathcal{P}_1, \ldots, \mathcal{P}_N$ need to be stored only once per cluster. The map of particle i can be computed by

$$m^{(i)} \quad = \quad \bigcup_n l^{(i)}_n \oplus \mathcal{P}_n. \tag{19}$$

Within one particle cluster, the local maps of each particle fulfills the assumption of local similarity. Therefore, they can share their patches. This results in a more compact representation compared to storing individual grid maps. In our current implementation, we used a graph structure where each node is a reference to the corresponding patch. To actually implement this representation, we store for each particle the state vector $s^{(i)}_t$

$$s^{(i)}_t = \langle \underbrace{x^{(i)}_t}_{\text{robot pose}}, \underbrace{k}_{\text{cluster ID}}, \underbrace{l^{(i)}_1, \ldots, l^{(i)}_{N_k}}_{\text{patches locations}} \rangle. \tag{20}$$

Each cluster C_k is represented by

$$C_k = \langle \underbrace{\mathcal{P}_1, \ldots, \mathcal{P}_{N_k}}_{\text{pointer to patches}}, \underbrace{\{e_{l,m}\}}_{\text{graph edges}} \rangle. \tag{21}$$

Note that the number N_k of patches does not grow with the length of trajectory traveled by the robot. It grows with the

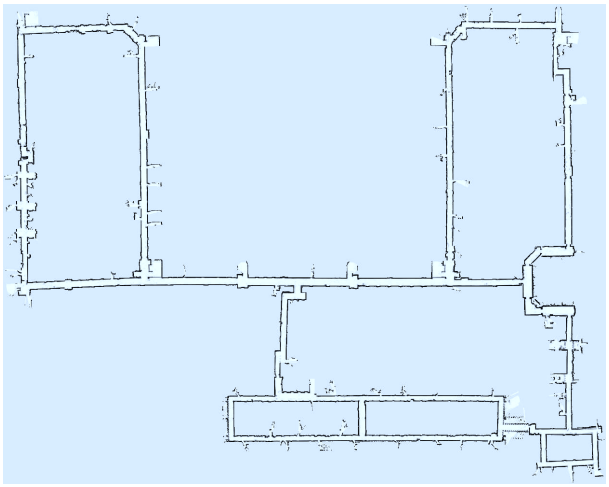


Fig. 3. Learned map of the MIT Killian Court using our approach.

number of relevant patches which is related to the size of the environment.

In the beginning of the mapping process, we start with a single cluster, but after closing a loop, multiple topology hypotheses typically occur. Whenever a topological hypothesis represented by a particle cluster needs to be split up, one has to determine which particle belongs to which topological hypothesis. In our current implementation, we cluster the samples according to their Euclidian distance to the different nodes in their own graph structure of reference frames. For each particle, we determine the list of nodes in the field of view of that sample. We order this list according to the Euclidian distance from the pose represented by the sample to the corresponding node. Then, a cluster is given by the samples which have the same list of nodes. An example which illustrates how new clusters are created in case of a loop-closure is depicted in Figure 2.

Throughout our experiments, we observed that multiple particle clusters are created when closing a loop. The actual number ranges up to 50. However, after a few iterations only a small number of cluster (up to 5) typically survive.

Note that it might be possible to represent each cluster by an EKF and not by particles like we do. However, in this case one would have to deal with linearization problems and Gaussian uncertainty. Furthermore, our approach uses grid maps and does not rely on predefined feature extractors like typical EKF approaches do.

To fulfill Assumption 3, we apply an incremental scan alignment technique based on laser range finder data. The experiments presented in this paper indicate that this setup/implementation is sufficient to satisfy the three assumptions. As a result, we obtain a mapping system which provides highly accurate maps in a fast and memory efficient manner.

VI. Experiments

In this section, we present experiments based on real robot datasets which are commonly used within the SLAM community. In the first experiment, we corrected several log files using our approach. Figure 3 depicts the resulting map of the MIT Killian Court. This is a challenging dataset, since it is a large (it took 2.5h to record this log file) and it contains

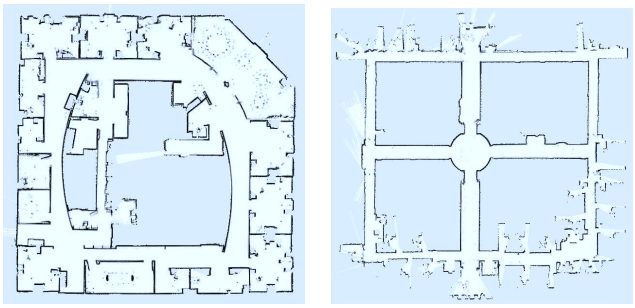


Fig. 4. The left image depict the Intel Research Lab and the right one the Austin ACES building at the University of Texas.

TABLE I

Comparison of memory and computational resources based on the MIT dataset using a PC with a 1.3 GHz CPU.

	#particles	**execution time**	**max. memory**
our approach	2,000	51 min	210 MB
our approach	1,000	41 min	180 MB
our approach	500	30 min	165 MB
RBPF of [18]	150	(memory swapping)	2.9 GB
RBPF of [18]	80	300 min	1.5 GB
RBPF of [18]	50	190 min	1 GB

several nested loops which can lead to particle depletion. As shown in this figure, the map does not show inconsistencies like for example double walls. Comparable results have been obtained using the Intel Research Lab and the Austin ACES dataset which are both depicted in Figure 4.

The second experiment is designed to show the advantages of our approach compared to a Rao-Blackwellized mapper without our optimizations. For this comparison, we used the open-source mapper provided in [18]. We compared the overall time, needed to correct the MIT Killian Court dataset and the memory used to store the maps. This was done using a (comparably slow) PC with a 1.3 GHz CPU and 1.5 GB RAM. The results of both mapping approaches are summarized in Table I. In our current implementation, the filter update for *each cluster* takes in average 20 ms when moving through known terrain and 200 ms when moving through unknown terrain. When actually closing a loop, *each particle* requires approximatively 2 ms of execution time while the check for the closure takes around 0.3 ms per sample.

Since the approximated proposal is not as accurate as the original one, we need more particles to achieve the same robustness in filter convergence and quality of the resulting maps. However, we can maintain more than one order of magnitude more particles while requiring less runtime and memory. In all our experiments, this sufficiently accounted for the less accurately drawn samples.

The savings on runtime are mainly caused by transforming an already computed proposal distribution so that it can be used for several particles instead of computing it from scratch each time. The memory savings are due to the fact that all particles within a cluster can share a single map model. Furthermore, the memory usage and runtime of our approach grows much slower when increasing the number of particles. The reason is that the complexity of our filter grows mainly with the number of topological hypotheses (particle clusters) which need to be maintained and not directly with the number

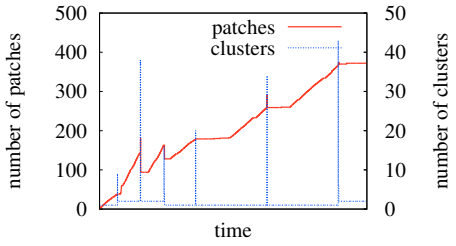


Fig. 5. This plot depicts the number of patches in the memory and the number of clusters over time for the MIT dataset using 1500 particles.

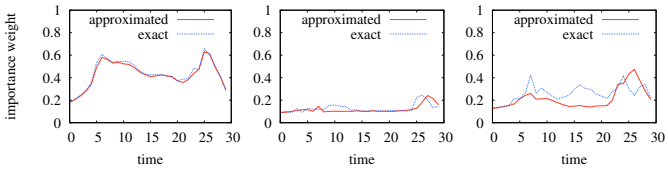


Fig. 6. Difference in the particle weights caused the approximative computation for three different samples during a loop closure. The left and middle image show typical results, the right one depicts the one of the worst results during our experiments.

of samples. Notice that the *maximum* memory usage shown of our approach is much higher than the typical one. There exist a few peaks in the memory usage which arise from a loop closure where several clusters are temporarily created but deleted after a few steps (compare Figure 5). The typical memory usage is around 20% of the maximum usage.

Figure 5 depicts the number of patches that need to be stored and the number of clusters during the estimation process of the MIT dataset with 1,500 particles. As can be seen, the number of clusters is typically small until a loop closure occurs. At this point, the number of clusters increases. However, after a short period of time most of the clusters vanish.

The last experiment evaluates the error introduced by our approximative importance weight computation when closing a loop. We ignore the normalization factors to achieve a faster estimation. We analyzed the loop-closing actions and in most situations the approximation error was small. Figure 6 depicts the differences between the sound computation and our approximation for three different particles during a loop closure. For a more quantitative evaluation between both methods, we computed the KL-divergence (KLD) between the distribution of the importance weights in both cases. It turned out, that the average KLD was only 0.02 (a KLD of 0 means that the distributions are equal and the higher the value the more different are the distributions). Substantiated by the good approximation quality, we ignore the evaluation of η_1 and η_2 when computing the particle importance weight.

VII. Conclusion

In this paper, we presented efficient optimizations for Rao-Blackwellized SLAM on grid maps. We are able to update the complex posterior requiring substantially less resources by performing the computations only for a set of representatives instead of for all particles. We extended a state-of-the-art mapping system in a way that the computation of the proposal distribution is significantly faster and needs only a fraction of the memory resources. The key idea is that clusters of particles can share a compact map representation as well as an informed proposal distribution to draw the next generation of particles.

With our optimizations, we are able to maintain more than one order of magnitude more samples and at the same time require less memory and computational resources compared to other state-of-the-art Rao-Blackwellized mapping techniques. This increase in number of particles we are able to maintain compensates for the errors introduced by our approximations.

Our approach has been implemented, tested, and evaluated based on real robots and standard log files used within the SLAM community to demonstrate the accuracy as well as the benefits of our system.

References

[1] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An altas framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[2] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[3] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[4] C. Estrada, J. Neira, and J.D. Tardós. Hierachical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[5] R. Eustice, M. Walter, and J.J. Leonard. Sparse extended information filters: Insights into sparsification. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 641–648, 2005.

[6] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17–26, Seattle, WA, USA, 2001.

[7] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2443–2448, Barcelona, Spain, 2005.

[8] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symp. on Computational Intelligence in Robotics & Automation (CIRA)*, pages 318–325, 1999.

[9] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

[10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

[11] B. Lisien, D. Silver D. Morales, G. Kantor, I.M. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[12] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

[13] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[14] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2002.

[15] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 1999.

[16] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6), 2001.

[17] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[18] C. Stachniss and G. Grisetti. Mapping results obtained with Rao-Blackwellized particle filters. http://www.informatik.uni-freiburg.de/~stachnis/research/rbpfmapper/, 2004.

[19] C. Stachniss, D. Hähnel, W. Burgard, and G. Grisetti. On actively closing loops in grid-based fastslam. *Advanced Robotics*, 19:1059–1080, 2005.

[20] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7/8), 2004.

[C25] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 143–157. Springer-Verlag Berlin Heidelberg, Germany, 2006.

Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera

Maren Bennewitz, Cyrill Stachniss, Wolfram Burgard, and Sven Behnke

University of Freiburg, Computer Science Institute, D-79110 Freiburg, Germany

Abstract. The Scale Invariant Feature Transform (SIFT) has become a popular feature extractor for vision-based applications. It has been successfully applied to metric localization and mapping using stereo vision and omnivision. In this paper, we present an approach to Monte-Carlo localization using SIFT features for mobile robots equipped with a single perspective camera. First, we acquire a 2D grid map of the environment that contains the visual features. To come up with a compact environmental model, we appropriately down-sample the number of features in the final map. During localization, we cluster close-by particles and estimate for each cluster the set of potentially visible features in the map using ray-casting. These relevant map features are then compared to the features extracted from the current image. The observation model used to evaluate the individual particles considers the difference between the measured and the expected angle of similar features. In real-world experiments, we demonstrate that our technique is able to accurately track the position of a mobile robot. Moreover, we present experiments illustrating that a robot equipped with a different type of camera can use the same map of SIFT features for localization.

1 Introduction

Self-localization is one of the fundamental problems in mobile robotics. The topic was studied intensively in the past. Many approaches exist that use distance information provided by a proximity sensor for localizing a robot in the environment. However, for some types of robots, proximity sensors are not the appropriate choice because they do not agree with their design principle. Humanoid robots, for example, which are constructed to resemble a human, are typically equipped with vision sensors and lack proximity sensors like laser scanners. Therefore, it is natural to equip these robots with the ability of vision-based localization.

In this paper, we present an approach to vision-based mobile robot localization that uses a single perspective camera. We apply the well-known Monte-Carlo localization (MCL) technique [5] to estimate the robot's position. MCL uses a set of

random samples, also called particles, to represent the belief of the robot about its pose. To locate features in the camera images, we use the Scale Invariant Feature Transform (SIFT) developed by Lowe [15]. SIFT features are invariant to image translation, scale, and rotation. Additionally, they are partially invariant to illumination changes and affine or 3D projection. These properties make SIFT features particularly suitable for mobile robots since, as the robots move around, they typically observe landmarks from different angles and distances, and with a different illumination.

Whereas existing systems, that perform metric localization and mapping using SIFT features, apply stereo vision in order to compute the 3D position of the features [20, 7, 21, 2], we rely on a single camera only during localization. Since we want to concentrate on the localization aspect, we facilitate the map acquisition process by using a robot equipped with a camera and a proximity sensor. During mapping, we create a 2D grid model of the environment. In each cell of the grid, we store those features that are supposed to be at that 2D grid position. Since the number of observed SIFT features is typically high, we appropriately down-sample the number of features in the final map. During MCL, we then rely on a single perspective camera and do not use any proximity information. Our approach estimates for clusters of particles the set of potentially visible features using ray-casting on the 2D grid. We then compare those features to the features extracted from the current image. In the observation model of the particle filter, we consider the difference between the measured and the expected angle of similar features. By applying the ray-casting technique, we avoid comparing the features extracted out of the current image to the whole database of features (as the above mentioned approaches do), which can lead to serious errors in the data association. As we demonstrate in practical experiments with a mobile robot in an office environment, our technique is able to reliably track the position of the robot. We also present experiments illustrating that the same map of SIFT features can be used for self-localization by different types of robots equipped with a single camera only and without proximity sensors.

This paper is organized as follows. After discussing related work in the following section, we describe the Monte-Carlo localization technique that is applied to estimate the robot's position. In Section 4, we explain how we acquire 2D grid maps of SIFT features. In Section 5, we present the observation model used for MCL. Finally, in Section 6, we show experimental results illustrating the accuracy of our approach to estimate the robot's position.

2 Related Work

Monte-Carlo methods are widely used for vision-based localization and have been shown to yield quite robust estimates of the robot's position. Several localization approaches are image-based, which means that they store a set of reference images taken at various locations that are used for localization. Some of the image-based methods rely on an omnidirectional camera in order to localize a mobile robot. The advantages of omnidirectional images are the circular field of view and thus, the

knowledge about the appearance of the environment in all possible gaze directions. Recent techniques were for example presented by Andreasson et al. [1] who developed a method to match SIFT features extracted from local interest points in panoramic images, by Menegatti et al. [16] who use Fourier coefficients for feature matching in omnidirectional images, and by Gross et al. [9] who compare the panoramic images using color histograms. Wolf et al. [23] apply a combination of MCL and an image retrieval system in order to localize a robot equipped with a perspective camera. The systems presented by Ledwich and Williams [12] and by Kŏsécka and Li [11] perform Markov localization within a topological map. They use the SIFT feature descriptor to match the current view to the reference images. Whenever using those image-based methods, care has to be taken in deciding at which positions to collect the reference images in order to ensure a complete coverage of the space the robot can travel in. In contrast to this, our approach stores features at the positions where they are located in the environment and not for all possible poses the robot can be in.

Additionally, localization techniques have been presented that use a database of observed visual landmarks. SIFT features have become very popular for metric localization as well as for SLAM (simultaneous localization and mapping, [21, 2]). Se et al. [20] were the first who performed localization using SIFT features in a restricted area. They did not apply a technique to track the position of the robot over time. Recently, Elinas and Little [7] presented a system that uses MCL in combination with a database of SIFT features learned in the same restricted environment. All these approaches use stereo vision to compute the 3D position of a landmark and match the visual features in the current view to all those in the database to find correspondences. To avoid matching the observations to the whole database of features, we present a system that determines the sets of visible features for clusters of particles. These relevant features are then matched to the features in the current image. This way, the number of ambiguities, which can occur in larger environments, is reduced. The relevant features are determined by applying a ray-casting technique in the map of features. The main difference to existing metric localization systems using SIFT features is however that our approach is applicable to robots that are equipped with a single perspective camera only, whereas the other approaches require stereo vision or omnivision.

Note that Davison et al. [3] and Lemaire et al. [13] presented approaches to feature-based SLAM using a single camera. These authors use extended Kalman filters for state estimation. Both approaches have only been applied to robots moving within a relatively small operational range.

Vision-based MCL was first introduced by Dellaert et al. [4]. The authors constructed a global ceiling mosaic and use simple features extracted out of images obtained with a camera pointing to the ceiling for localization. Systems that apply vision-based MCL are also popular in the RoboCup domain. In this scenario, the robots use environment-specific objects as features [19, 22].

3 Monte-Carlo Localization

To estimate the pose x_t (position and orientation) of the robot at time t, we apply the well-known Monte-Carlo localization (MCL) technique [5], which is a variant of Markov localization. MCL recursively estimates the posterior about the robot's pose:

$$p(x_t \mid z_{1:t}, u_{0:t-1}) \\ = \eta \cdot p(z_t \mid x_t) \cdot \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_{t-1}) \cdot p(x_{t-1} \mid z_{1:t-1}, u_{0:t-2})\, dx_{t-1} \quad (1)$$

Here, η is a normalization constant resulting from Bayes' rule, $u_{0:t-1}$ denotes the sequence of all motion commands executed by the robot up to time $t-1$, and $z_{0:t}$ is the sequence of all observations. The term $p(x_t \mid x_{t-1}, u_{t-1})$ is called motion model and denotes the probability that the robot ends up in state x_t given it executes the motion command u_{t-1} in state x_{t-1}. The observation model $p(z_t \mid x_t)$ denotes the likelihood of making the observation z_t given the robot's current pose is x_t. To determine the observation likelihood, our approach compares SIFT features in the current view to those SIFT features in the map that are supposed to be visible (see Section 5).

MCL uses a set of random samples to represent the belief of the robot about its state at time t. Each sample consists of the state vector $x_t^{(i)}$ and a weighting factor $\omega_t^{(i)}$ that is proportional to the likelihood that the robot is in the corresponding state. The update of the belief, also called particle filtering, is typically carried out as follows. First, the particle states are predicted according to the motion model. For each particle a new pose is drawn given the executed motion command since the previous update. In the second step, new individual importance weights are assigned to the particles. Particle i is weighted according to the likelihood $p(z_t \mid x_t^{(i)})$. Finally, a new particle set is created by resampling from the old set according to the particle weights. Each particle survives with a probability proportional to its importance weight.

Due to spurious observations it is possible that good particles vanish because they have temporarily a low likelihood. Therefore, we follow the approach proposed by Doucet [6] that uses the so-called number of effective particles [14] to decide when to perform a resampling step

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2}, \quad (2)$$

where N is the number of particles. N_{eff} estimates how well the current particle set represents the true posterior. Whenever N_{eff} is close to its maximum value N, the particle set is a good approximation of the true posterior. Its minimal value 1 is obtained in the situation in which a single particle has all the probability mass contained in its state.

We do not resample in each iteration, instead, we only resample each time N_{eff} drops below a given threshold (here set to $\frac{N}{2}$). In this way, the risk of replacing good particles is drastically reduced.

4 Acquiring 2D Maps of Scale-Invariant Features

We use maps of visual landmarks for localization. To detect features, we use the Scale Invariant Feature Transform (SIFT). Each image feature is described by a vector $\langle p, s, r, f \rangle$ where p is the subpixel location, s is the scale, r is the orientation, and f is a descriptor vector, generated from local image gradients. The SIFT descriptor is invariant to image translation, scaling, and rotation and also partially invariant to illumination changes and affine or 3D projection. Lowe presented results illustrating robust matching of SIFT descriptors under various image transformations [15]. Mikolajczyk and Schmid compared SIFT and other image descriptors and showed that SIFT yields the highest matching accuracy [17].

Ke and Sukthankar [10] presented an approach to compute a more compact representation for SIFT features, called PCA-SIFT. They apply principal components analysis (PCA) to determine the most distinctive components of the feature vector. As shown in their work, the PCA-based descriptor is more distinctive and more robust than the standard SIFT descriptor. We therefore use that representation in our current approach. As suggested by Ke and Sukthankar, we apply a 36 dimensional descriptor vector resulting from PCA.

To acquire a 2D map of SIFT features, we used a B21r robot equipped with a perspective camera and a SICK laser range finder. We steered the robot through the environment to obtain image data as well as proximity and odometry measurements. The robot was moving with a speed of $40cm/s$ and collected images at a rate of $3Hz$. To be able to compute the positions of features and to obtain ground truth data, we used an approach to grid-based SLAM with Rao-Blackwellized particle filters [8]. Using the information about the robot's pose and extracted SIFT features out of the current camera image, we can estimate the positions of the features in the map. More specifically, we use the distance measurement of the laser beam that corresponds to the horizontal angle of the detected feature and the robot's pose to calculate the 2D position of the feature. Thus, we assume that the features are located on the obstacles detected by the laser range finder. In the office environment in which we performed our experiments, this assumption leads to quite robust estimates even if there certainly exist features that are not correctly mapped. In each 2D grid cell, we store the set of features that are supposed to be at that 2D grid position. Currently, we use a grid resolution of 10 by $10cm$. In the first stage of mapping, we store all observed features.

After the robot moved through the environment, the number of observed SIFT features is extremely high. Typically, we have 150-500 features extracted per image with a resolution of 320 by 240 pixels. This results in around 600,000 observed features after the robot traveled for $212m$ in a typical office environment. After map acquisition, we down-sample a reduced set of features that is used for localization. For each grid cell, we randomly draw features. A drawn feature is rejected if there is already a similar feature within the cell. We determine similar features by comparing their PCA-SIFT vectors (see below). We sample a maximum of 20 features for each grid cell. Using the sampling process, features that were observed more often have a higher chance to be selected and features that were detected only once (due to

failure observations or noise) are eliminated. The goal of this sampling process is to reduce computational resources and at the same time obtain a representative subset of features. To choose good representatives for the features, a clustering based on the descriptor vectors can be carried out.

The left image of Figure 3 shows a 2D grid map of SIFT features of an office environment that was acquired by the described method. The final map contains approximately 100,000 features. Note that also a stereo camera system, which was not available in our case, would be an appropriate solution for map building. The presented map acquisition approach is not restricted to robots equipped with a laser range finder.

5 Observation Model for SIFT Features

In the previous section, we described how to built a map of SIFT features using a robot equipped with a camera and a proximity sensor. In this section, we describe how a robot without a proximity sensor can use this environmental model for localization with a single perspective camera.

Sensor observations are used to compute the weight of each particle by estimating the likelihood of the observation given the pose of the particle in the map. Thus, we have to specify how to compute $p(z_t \mid x_t)$. In our case, an observation z_t consists of the SIFT features in the current image: $z_t = \{o_1, \ldots, o_M\}$ where M is the number of features in the current image. To determine the likelihood of an observation given a pose in the map, we compare the observed features with the features in the map by computing the Euclidean distance of their PCA-SIFT descriptor vectors.

In order to avoid comparing the features in the current image to the whole set of features contained in the map, we determine the potentially visible features. This helps to cope with an environment that contains similar landmarks at different locations (e.g. several similar rooms). In case one matches the current observation against the whole set of features, this leads to serious errors in the data association.

To compute the relevant features, we group close-by particles to a cluster. We determine for each particle cluster the set of features that are potentially visible from these locations. This is done using ray-casting on the feature grid map. To speed-up the process of finding relevant features, one could also precompute for each grid cell the set of features that are visible. However, in our experiments, computing the similarity of the feature vectors took substantially longer than the ray-casting operations. Typically, we have 150-500 features per image.

In order to compare two SIFT vectors, we use a distance function based on the Euclidian distance. The likelihood that the two PCA-SIFT vectors f and f' belong to the same feature is computed as

$$p(f = f') = exp\Big(- \frac{\|f - f'\|}{2 \cdot \sigma_1^2} \Big), \tag{3}$$

where σ_1 is the variance of the Gaussian.

In general, one could use Eq. (3) to determine the most likely correspondence between an observed feature and the map features. However, since it is possible that different landmarks exist that have a similar descriptor vector, we do not determine a unique corresponding map feature for each observed feature. In order to avoid misassignments, we instead consider all pairs of observed features and relevant map features. This set of pairs of features is denoted as C. For each pair of features in C we use Eq. (3) to compute the likelihood that the corresponding PCA-SIFT vectors belong to the same feature.

This information is than used to compute the likelihood $p(z_t \mid x_t^{(i)})$ of an observation given the pose $x_t^{(i)}$ of particle i, which is required for MCL. Since a single perspective camera does not provide depth information, we can use only the angular information to compute this likelihood. We therefore consider the difference between the horizontal angles of the currently observed features in the image and the features in the map to compute $p(z_t \mid x_t^{(i)})$. More specifically, we compute the distribution over the angular displacement of a particle given the observation and the map. For each particle, we compute a histogram over the angular differences between the observed features and the map features. The x-values in that histogram represent the angular displacement and the y-values its likelihood. The histogram is computed using the pairs of features in C evaluated using Eq. (3).

In particular, we compute for each pair $(o, l) \in C$ the difference between the horizontal angle at which the feature was observed and the angle at which the feature should be located according to the map and the particle pose. We add the likelihood that these features are equal, which is given by Eq. (3), to the corresponding bin of the histogram. As a result, we obtain a distribution about the angular error of the particle.

In mathematical terms, the value $h(b)$ of a bin b (representing the interval of angular differences from $\alpha^-(b)$ to $\alpha^+(b)$) in the histogram is given by

$$h(b) = \beta + \sum_{\left\{(o,l)\in C \,\middle|\, \alpha^-(b) \le \alpha(o) - \alpha(l) < \alpha^+(b)\right\}} p(f_o = f_l), \tag{4}$$

where $\alpha(\cdot)$ is the function that computes the horizontal angle of a feature for a given pose of the robot, f_o is the PCA-SIFT descriptor of feature o, and f_l of feature l accordingly. β is a constant greater that zero ensuring that no angular displacement has zero probability.

The histograms of particles that are close to the correct pose of the robot have high values around zero. In case that there are several similar features in the environment, the histogram has multiple modes.

One finally needs to compute the observation likelihood of a particle. So far, we computed the distribution about the horizontal angular displacement, not its actual value. In case of a uni-modal or Gaussian distribution it would be sufficient to consider only the distance of the mean from zero taking into account the variance. However, in real-world situations, it is likely that one obtains multi-modal distributions.

Each bin of that histogram stores the probability mass of the corresponding angular displacement of the particle. Therefore, we compute the observation likelihood given we have the angular displacement of that bin and multiply it with the value stored in that bin. The observation likelihood given the histogram is then computed by the sum over these values

$$p(z_t \mid x_t^{(i)}) = \sum_b h(b) \cdot exp\Big(- \frac{1}{2 \cdot \sigma_2^2} \cdot \Big[\frac{\alpha^+(b) + \alpha^-(b)}{2} \Big]^2 \Big), \tag{5}$$

where σ_2 is the variance of a Gaussian describing the likelihood of a particle depending on the angular displacement. Figure 1 illustrates the whole process of computing the observation likelihood for a single particle.

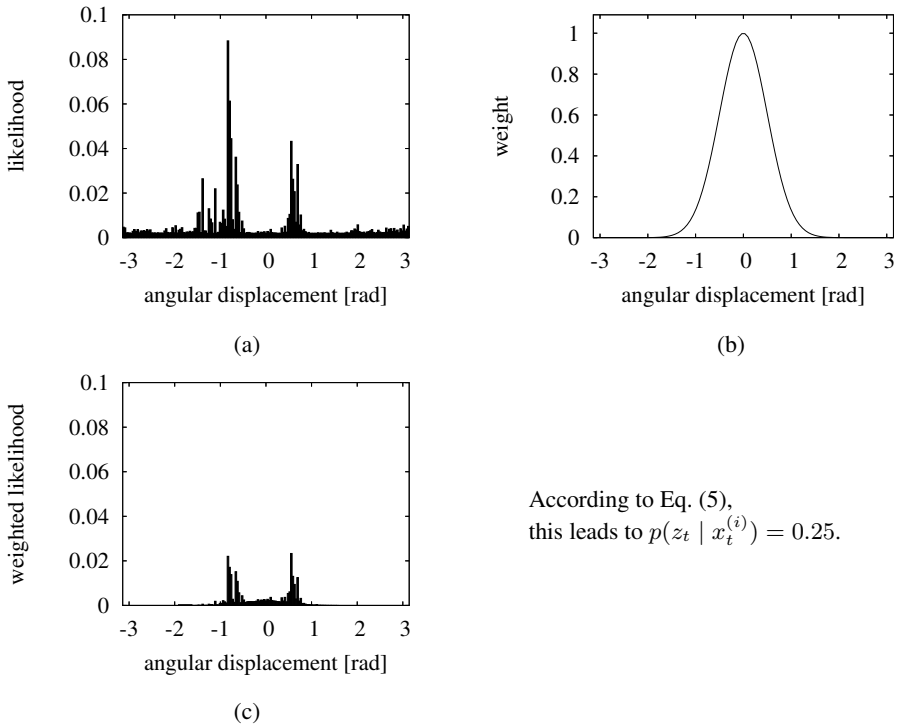


Fig. 1. Image (a) shows the distribution about the horizontal angular displacement for a particular particle computed according to Eq. (4). The plot shown in (b) depicts the Gaussian that is used to compute the weight of a sample depending on the displacement. Finally, image (c) shows the resulting histogram in which each bin of the histogram (a) is multiplied by the corresponding value of the Gaussian. Summing up the bins leads to an observation likelihood of 0.25.

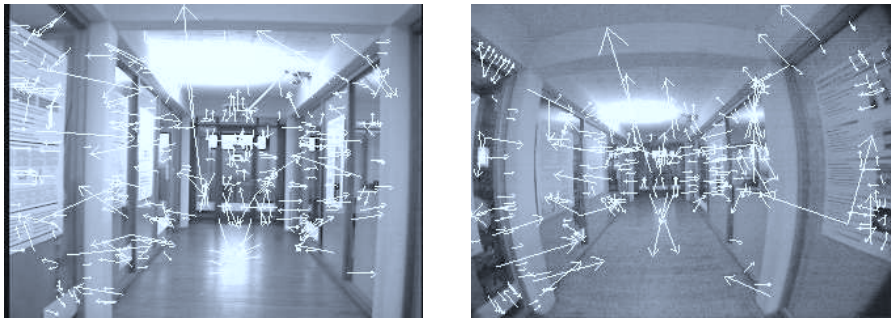


Fig. 2. Example images with generated SIFT features. The images were obtained from two different cameras used in the experiments. The standard camera (left) was used for map acquisition as well as for localization and a low-cost wide-angle camera (right) for further evaluation of our localization approach.

Note that a further improvement of the sensor model can be obtained by using the joint compatibility test between pairs of feature as proposed by Neira and Tardós [18] and not considering all possible data associations.

6 Experimental Results

To evaluate our approach to estimate the pose of the robot equipped with a single perspective camera, we carried out a series of real-world experiments with wheeled and humanoid robots in an office environment. The B21r robot that performed the mapping task carries a standard camera with an opening angle of approximately $65°$. In order to show that the acquired feature map can be used by robots equipped with different cameras, we performed the localization experiments using a low-cost wide-angle camera (with an opening angle of about $130°$). The difference between typical images of both cameras can be seen in Figure 2. The arrows indicate the location, orientation, and scale of the generated SIFT features. The acquired map is depicted in Figure 3.

6.1 Localization Accuracy

In this experiment, the wheeled robot traveled a distance of approximately $20m$. Figure 3 shows the estimated trajectory as well as the true pose of the robot during this experiment. The ground truth has been determined using laser range data. The evolution of the particle filter is illustrated in Figure 4. It shows the particle clouds as well as the true position and the pose estimate provided by odometry.

A more quantitative analysis showing the localization error over time can be found in Figure 5. Between time step 40 and 50, the error in the pose of the vehicle was comparably high. This is because we used the weighted mean of the samples

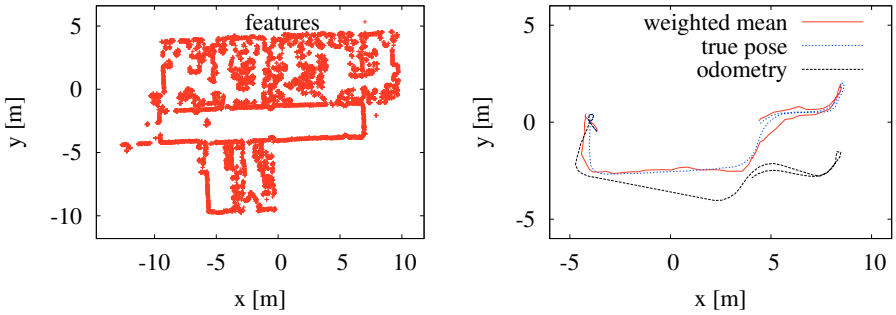


Fig. 3. The left image shows the 2D map acquired in a typical office environment. Each cross represents the estimated 2D position of a SIFT feature. The right image depicts the estimated trajectory as well as the ground truth of a localization experiment. As can be seen, the weighted mean of the particles is close to the true pose of the robot.

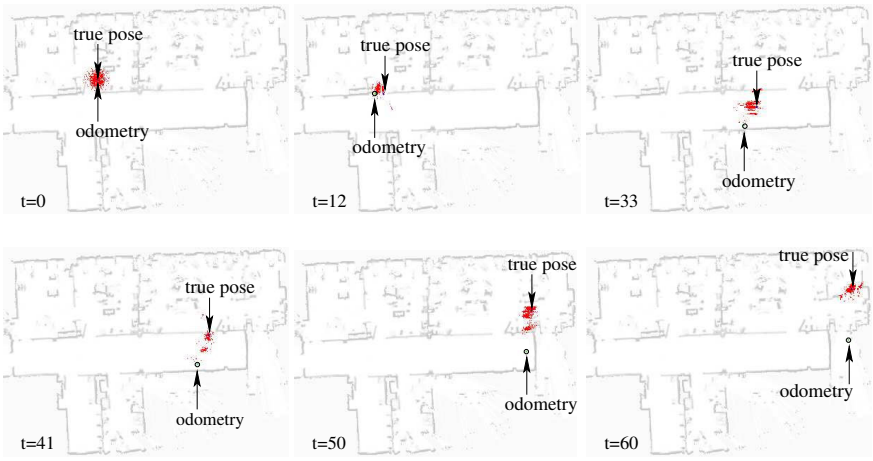


Fig. 4. The particle set during localization. The two arrows indicate the pose resulting from odometry information as well as the true pose of the robot. The true pose of the vehicle was determined by using a laser range finder that was mounted on the robot for this purpose. The occupancy grid map is only shown for a better illustration and was not used for localization.

for the error computation and because the belief was temporarily multi-modal. This fact can be observed in the snapshots depicted in Figure 4. As this experiment illustrates, our technique is able to accurately estimate the pose of the robot. The average error in the x/y-position was $39cm$. The average error in the orientation of the vehicle was $4.5°$. We got comparable localization results when using different cameras with a more constrained field of view like the one which was used for map acquisition. During our experiments, we used 800 particles in our particle filter, which were initialized with a Gaussian centered at the starting pose of the robot.

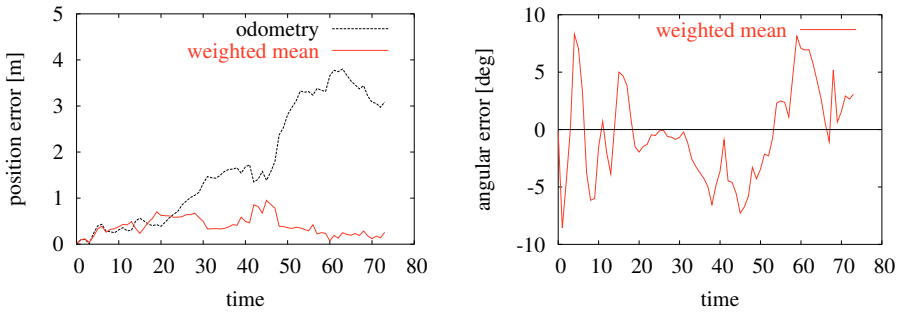


Fig. 5. Evolution of the error during the localization experiment depicted in Figure 3.

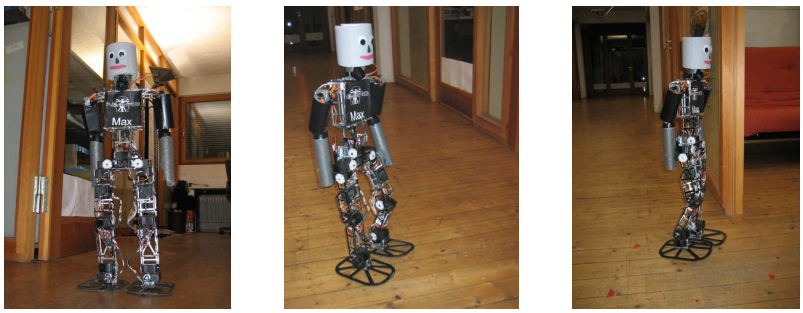


Fig. 6. The humanoid robot Max.

6.2 Tracking the Pose of a Humanoid Robot

To further evaluate our approach, we applied our localization technique to the humanoid robot depicted in Figure 6. To estimate the pose of the robot based on executed motion commands, we perform dead reckoning. The gait control input consists of motor currents that control the lateral, speed, sagittal, and the rotational speed. The estimated velocities are integrated to determine the relative movement. Compared to a wheeled robot equipped with odometry sensors, this leads to a noisy pose estimate. Furthermore, due to the design of the humanoid robot, the camera images are often blurred because of vibrations.

In this experiment, the robot Max traveled along the trajectory shown in Figure 7. The red circles correspond to position where an observation was made. The particle clouds obtained in this experiment are given in Figure 8. In case no sensor information is integrated, the pose estimate has a high uncertainty as can be seen in the first row of that figure. In contrast to this, the use of our vision-based localization technique reduces the uncertainty and enables to localize the humanoid. Note that due to unstable motion of the humanoid, missing odometry sensors, vibrations, and the shaking camera, the localization is less robust compared to a wheeled robot.

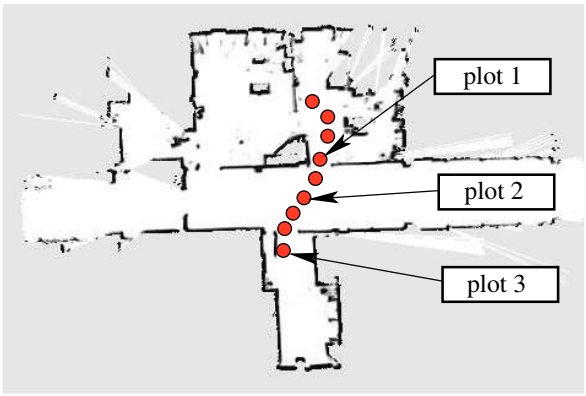


Fig. 7. The trajectory of Max. The red circles indicate the positions where observations were made. The corresponding plots of the particle clouds are shown in Figure 8.

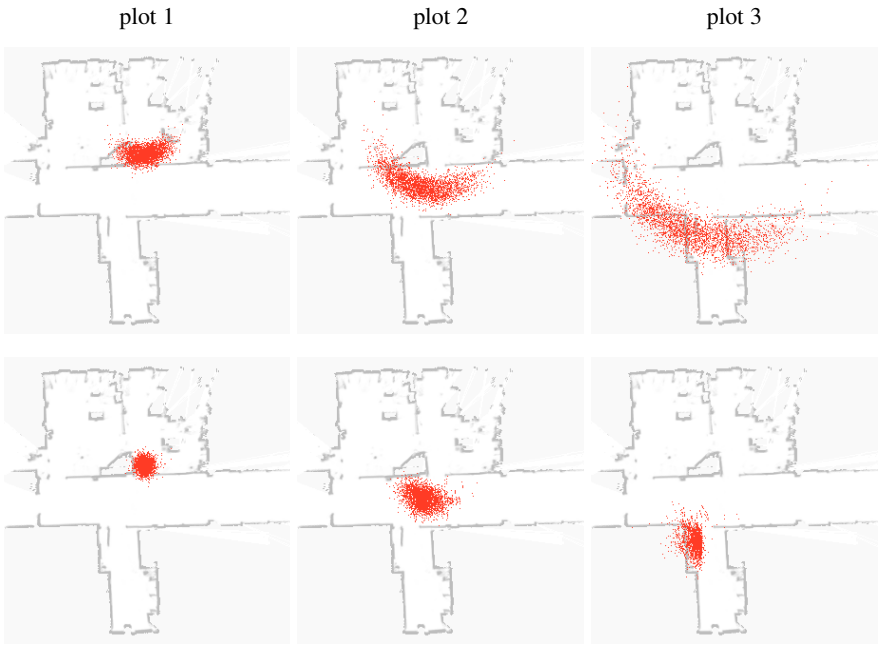


Fig. 8. Vision-based localization of a humanoid robot. The images in the first row depict the evolution of the particles in case no sensor information is used. The high uncertainty in the particle cloud results from the poor motion estimate resulting from dead reckoning. The images in the second row show the result of our localization approach. As can be seen, the visual information allows to accurately estimate the pose of the humanoid robot.

7 Conclusions

In this paper, we presented an approach to mobile robot localization that relies on a single perspective camera. Our technique is based on Monte-Carlo localization and uses SIFT features extracted from camera images. In the observation model of our particle filter, we compare descriptor vectors of features in the current image to the set of potentially visible map features given the pose of the particles. Based on this information, we compute a distribution about the angular displacement for each sample given the current observation. The evaluation of potential correspondences between features is done efficiently by performing the necessary computations for clusters of particles. By using only the relevant features in the vicinity of the particles in the observation model, we reduce the number of data association failures. As we demonstrate in real-world experiments carried out with a wheeled as well as with a humanoid robot, our system provides an accurate metric pose estimate for a mobile robot without requiring proximity sensors, omnivision, or a stereo camera.

Acknowledgment

This project is partially supported by the German Research Foundation (DFG), grant BE 2556/2-1 and SFB/TR-8 (A3). We would like to thank D. Lowe for providing his software to detect SIFT features and Y. Ke for his PCA-SIFT implementation. Further thanks to J. Stückler and M. Schreiber for helping us carrying out the experiments with the humanoid robot.

References

1. H. Andreasson, A. Treptow, and T. Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
2. T.D. Barfoot. Online visual motion estimation using FastSLAM with SIFT features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.
3. A.J. Davison, Y. González Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.
4. F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the Condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.
5. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.
6. A Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Departement of Engeneering, University of Cambridge, 1998.
7. P. Elinas and J.J. Little. σMCL: Monte-Carlo localization for mobile robots with stereo vision. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

8. G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
9. H.-M. Gross, A. Köning, C. Schröter, and H.-J. Böhme. Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
10. Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.
11. J. Kŏsécka and L. Li. Vision based topological Markov localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.
12. L. Ledwich and S. Williams. Reduced SIFT features for image retrieval and indoor localization. In *Australian Conf. on Robotics and Automation (ACRA)*, 2004.
13. T. Lemaire, S. Lacroix, and J. Solà. A practical 3D bearing-only SLAM algorithm. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.
14. J.S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statist. Comput.*, 6:113–119, 1996.
15. D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 1999.
16. E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based Monte-Carlo localisation with omnidirectional images. *Robotics & Autonomous Systems*, 48(1):17–30, 2004.
17. K. Mikolajczk and C. Schmid. A performance evaluation of local descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.
18. J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.
19. T. Röfer and M. Jüngel. Vision-based fast and reactive Monte-Carlo Localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.
20. S. Se, D.G. Lowe, and J.J. Little. Global localization using distinctive visual features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.
21. R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.
22. M. Sridharan, G. Kuhlmann, and P. Stone. Practical vision-based Monte Carlo localization on a legged robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
23. J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image retrieval system with Monte Carlo Localization. *IEEE Transactions on Robotics and Automation*, 21(2):208–216, 2005.

[W1] P. Pfaff, R. Kuemmerle, D. Joho, C. Stachniss, R. Triebel, and W. Burgard. Navigation in combined outdoor and indoor environments using m ulti-level surface maps. In *Workshop on Safe Navigation in Open and Dynamic Environments a t the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

Navigation in Combined Outdoor and Indoor Environments using Multi-Level Surface Maps

P. Pfaff* R. Kümmerle* D. Joho* C. Stachniss* R. Triebel+ W. Burgard*

*Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany
+Autonomous Systems Lab, Swiss Federal Institute of Technology, 8092 Zurich, Switzerland

Abstract— **Whenever mobile robots are used in real world applications, the ability to learn an accurate model of the environment and to localize itself based on such a model are important prerequisites for reliable operation. Whereas these problems have been successfully solved in the past for most indoor tasks, in which the robot is assumed to operate on a flat surface, such approaches are likely to fail in combined indoor and outdoor environments in which the three-dimensional structure of the world needs to be considered. In this paper, we consider the problem of localizing a vehicle that operates in 3D indoor as well as outdoor settings. Our approach is entirely probabilistic and does not rely on GPS information. It is based on so-called multi-level surface maps which are an extension of the well-known elevation maps. In addition to that, we present a technique that allows the robot to actively explore the environment. This algorithm applies a decision-theoretic approach and considers the uncertainty in the model to determine the next action to be executed. In practical experiments, we illustrate the properties as well as advantages of our approach compared to other techniques.**

I. Introduction

Robots that are able to acquire an accurate model of their environment and to localize themselves based on such a model are regarded as fulfilling a major precondition of truly autonomous mobile vehicles.

The problem of mobile robot localization with range sensors in outdoor environments arises whenever GPS signals are missing due to occlusions caused by buildings, bridges, or trees. Furthermore, in case of combined outdoor and indoor environments, relying on GPS information will obviously lead to failure in the pose estimate. In such situations, a mobile robot typically has to estimate its position in the environment using its exteroceptive sensors and a map of the environment. However, when a robot attempts to perceive its environment to localize itself, the choice of the direction of the perception can substantially influence the accuracy of the position estimate. The localization task requires a given map of the environment. In case such a model is not available, it has to be learned by the robot. This problem is also known as autonomous exploration. So far, most approaches to mobile robot exploration assume that the robot lives in a plane. They typically focus on generating motion commands that minimize the time needed to cover the whole terrain [13], [24]. A frequently used technique is to build an occupancy grid map since it can model unknown locations efficiently. The robot seeks to reduce the number of unobserved cells or the uncertainty in the grid map. In the three-dimensional space, however, such approaches are not directly applicable. The size of occupancy grid maps in 3D, for example, prevents the robot from exploring an environment larger than a few hundred square meters.

The contribution of this paper are solutions to the localization and to the autonomous exploration problem in three-dimensional, combined outdoor and indoor environments. Both techniques use multi-level surface maps to provide an appropriate model of the environment. The MCL-based localization technique does not require GPS information and uses only proximity data from a laser range finder as well as odometry information. Our exploration technique extents existing exploration approaches used in 2D to the three-dimensional space. It selects actions that reduce the uncertainty of the robot about the world. It does so by reasoning about potential measurements that can be obtained when selecting an action. Our approach is able to deal with negative obstacles like, for example, abysms, which is a problem of robots exploring a three-dimensional world. Experiments carried out in simulation and on a real robot show the effectiveness of our techniques.

II. Related Work

The problem of localizing a mobile robot in indoor and outdoor environments with range sensors or cameras has been studied intensively in the past. In indoor environments, Monte-Carlo localization (MCL) [5] is one of the current state-of-the-art approaches. Outdoors, Adams *et al.* [1] extract predefined features from range scanners and apply a particle filter for localization. Davison and Kita [4] utilize a Kalman filter for vision-based localization with point features on non-flat surfaces. Recently, Agrawal and Konolige [2] presented an approach to robot localization in outdoor terrains based on feature points that are tracked across frames in stereo images. Lingemann *et al.* [15] recently described a method for fast localization in in- and outdoor environments. Their system operates on raw data sets, which results in huge memory requirements. Additionally, they apply a scan-matching routine for localization, which does not facilitate global localization. To reduce the memory requirements of outdoor terrain representations, several researchers applied elevation maps [3], [12], [14], [17]. A probabilistic approach to localize a planetary rover in such elevation maps has been described by Olson [16]. In this system, elevation maps were sufficient to robustly localize the vehicle, mainly because

the number of vertical and overhanging objects is negligible in environments like on Mars. However, environments on earth contain many objects like buildings or trees which have vertical or even overhanging surfaces. To address this issue, we use multi-level surface (MLS) maps [22] to represent the environment in this paper. MLS maps discretize the environment into cells and store for each cell a list of patches representing the individual layer in the environment as well as vertical structures.

So far, most approaches to mobile robot exploration assume that the robot lives in a plane. They typically focus on generating motion commands that minimize the time needed to cover the whole terrain [13], [24]. A frequently used technique is to build an occupancy grid map since it can model unknown locations efficiently. The robot seeks to reduce the number of unobserved cells or the uncertainty in the grid map [24], [18]. In the three-dimensional space, however, such approaches are not directly applicable. The size of occupancy grid maps in 3D, for example, prevents the robot from exploring an environment larger than a few hundred square meters.

Whaite and Ferrie [23] presented an exploration approach in 3D that uses the entropy to measure the uncertainty in the geometric structure of objects that are scanned with a laser range sensor. In contrast to the work described here, they use a fully parametric representation of the objects and the size of the object to model is bounded by the range of the manipulator. Surmann *et al.* [20] extract horizontal planes from a 3D point cloud and construct a polygon with detected lines (obstacles) and unseen lines (free space connecting detected lines). They sample candidate viewpoints within this polygon and use 2D ray-casting to estimate the expected information gain. In contrast to this, our approach uses an extension of 3D elevation maps and 3D ray-casting to select the next viewpoint. González-Baños and Latombe [9] also build a polygonal map by merging safe regions. Similar to our approach, they sample candidate poses in the visibility range of frontiers to unknown area. But unlike in our approach, they build 2D maps and do not consider the uncertainty reduction in the known parts of the map. Fournier *et al.* [8] present an 3D exploration approach utilizing an octree structure to represent the environment. However, it is unclear if the presented approach is able to explore on multiple levels.

The contribution of this paper are techniques for autonomously learning MLS maps with a mobile robot based on laser range finder and odometry only. We furthermore describe how a robot can utilize such a model to track its own pose and to globally localize itself. Our approach does not rely on GPS information and thus allows a robot to operate in combined indoor and outdoor scenarios.

III. 3D Model of the Environment

Our exploration system uses multi-level surface maps (MLS maps) as proposed by Triebel *et al.* [22]. MLS maps use a two-dimensional grid structure that stores different elevation values. In particular, they store in each cell of a discrete grid the height of the surface in the corresponding area. In contrast to elevation maps, MLS maps allow us to store multiple surfaces in each cell. Each surface is represented by a Gaussian with the mean elevation and its uncertainty σ. In the remainder of this paper, these surfaces are referred to as patches. This representation enables a mobile robot to model environments with structures like bridges, underpasses, buildings, or mines. They also enable the robot to represent vertical structures by storing a vertical depth value for each patch. Figure 1 shows two example maps from the same environment. The left image shows that it is not possible to represent an underpass , overhanging and vertical objects correctly using elevation maps. On the other hand the right image illustrates the ability of the MLS map approach to represent all these structures correctly.

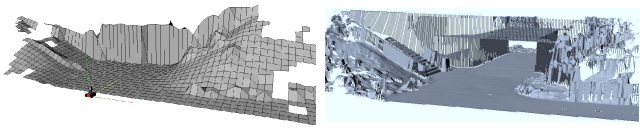


Fig. 1. Standard elevation map (left) which is not able to represent the underpass under the bridge correctly, and multi-level surface map (right) that correctly represents the height of the vertical objects and is able to model multiple levels.

IV. GPS-free Localization using MLS Maps

In this chapter, we assume that the robot already has a multi-level surface map available for localization. In the next chapter, we then present a technique to autonomously learn a MLS map.

To estimate the pose $\mathbf{x} = (x, y, z, \varphi, \vartheta, \psi)$ of the robot in its environment, we consider probabilistic localization, which follows the recursive Bayesian filtering scheme. The key idea of this approach is to maintain a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1})$ of the robot's location $\mathbf{x}_t$ at time t given all observations $\mathbf{z}_{1:t}$ up to time t and all control inputs $\mathbf{u}_{0:t-1}$ up to time $t-1$. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1}) = \alpha \cdot p(\mathbf{z}_t \mid \mathbf{x}_t) \cdot \int p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1}) \cdot p(\mathbf{x}_{t-1}) \, d\mathbf{x}_{t-1}. \quad (1)$$

Here, α is a normalization constant ensuring that $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1})$ sums up to one over all $\mathbf{x}_t$. The terms to be described in Eqn. (1) are the *prediction model* $p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1})$ and the *sensor model* $p(\mathbf{z}_t \mid \mathbf{x}_t)$. One major contribution of this paper is an appropriate computation of these models in the case that an MLS map is given.

For the implementation of the described filtering scheme, we use a sample-based approach which is commonly known as *Monte Carlo localization* [5]. Monte-Carlo localization is a variant of particle filtering [6] where each particle corresponds to a possible robot pose and has an assigned weight w_i. The *belief update* from Eqn. (1) is performed by the following two alternating steps:

1) In the **prediction step**, we draw for each particle with weight w_i a new particle according to w_i and to the prediction model $p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1})$.

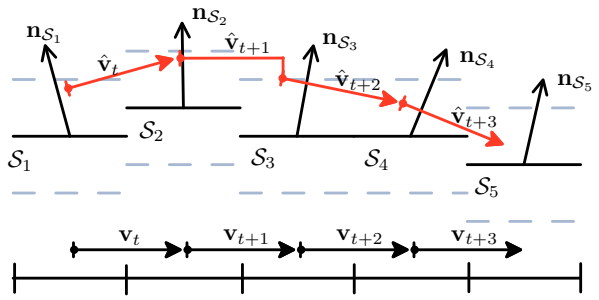


Fig. 2. Application of our prediction model to a series of 2D motion vectors (black). They are rotated to estimate the 3D motion vectors (red). The dashed line indicates the tolerance interval for the z-coordinate.

2) In the **correction step**, a new observation $\mathbf{z}_t$ is integrated. This is done by assigning a new weight w_i to each particle according to the sensor model $p(\mathbf{z}_t \mid \mathbf{x}_t)$.

A. Prediction Model for MLS Maps

The prediction model $p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1})$ we use is based on an approach introduced by Eliazar *et al.* [7]. It reflects systematic errors such as drift, as well as the uncertainty in the execution of an action $\mathbf{u} = (x_u, y_u, \theta_u)$, where (x_u, y_u) is the translation and θ_u the rotation angle. To incorporate this 2D motion into our 3D map we proceed as follows. First, we obtain a possible outcome (x_v, y_v, θ_v) of the action by applying the probabilistic model. Then, we adapt the motion vector $\mathbf{v} = (x_v, y_v)$ to the shape of the 3D surface traversed by the robot. This surface is obtained from the given MLS map and consists of planar square patches. To adapt the motion vector, we discretize it into segments of length c, which is the cell size of the MLS map, in our case 0.1 m. For each segment, we determine the corresponding surface patch $\mathcal{S}$ and rotate the segment according to the orientation (φ_S, ϑ_S) of the patch, where φ_S is the rotation about the x-axis and ϑ_S the rotation about the y-axis. The patch orientation is computed from the normal vector $\mathbf{n}_S$ of the patch $\mathcal{S}$, which in turn is obtained by fitting a plane into the local vicinity of $\mathcal{S}$. The normal vector computation is done beforehand and constitutes an extension to the framework of MLS maps. In general, it is not robust against noise and small errors in the MLS map, which results in an uncertainty of the patch orientation. In our approach, we model this uncertainty by adding Gaussian noise to the orientation parameters φ_S and ϑ_S. Thus, our prediction model expresses the uncertainty in 5 out of 6 position parameters – x, y and ψ by the 2D motion model and φ and ϑ by our 3D extension. For the last one – the height value z – we have the constraint that the robot must stay on the ground. Therefore, we adjust the z-value manually whenever it is too high or too low. This is illustrated in Figure 2. Finally, after concatenating all transformed motion vector segments, we obtain a new 3D motion vector $\hat{\mathbf{v}}$ which is added to the current estimate of the robot position $\mathbf{x}_{t-1}$ to obtain a new position estimate $\mathbf{x}_t$.

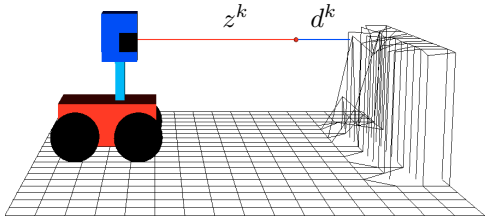


Fig. 3. Example of a single beam which ends close to vertical object in the MLS map. In the end point model, the probability $p_{hit}(z_k \mid \mathbf{x})$ only depends on the distance d^k between the end point of the k-th laser beam and the closest obstacle in the map.

B. Endpoint Sensor Model for MLS Maps

In our sensor model, we treat each beam independently and determine the likelihood of a whole laser scan by factorizing over all beams. Thus, we have

$$p(\mathbf{z} \mid \mathbf{x}) = \prod_{k=1}^{K} p(z^k \mid \mathbf{x}) \tag{2}$$

where K is the number of beams in each laser measurement $\mathbf{z}$. In Eqn. (2) and in the following, we drop the index t for convenience. Our sensor model $p(z^k \mid \mathbf{x})$ is based on an approach that has been introduced by Thrun [21] as *likelihood fields* (LF) or *end point model*. In particular, we formulate the sensor model $p(z^k \mid \mathbf{x})$ for each particular beam as a mixture of three different distributions:

$$\begin{aligned} p(z^k \mid \mathbf{x}) = \alpha_{hit} p_{hit}(z^k \mid \mathbf{x}) + \\ \alpha_{rand} p_{rand}(z^k \mid \mathbf{x}) + \alpha_{max} p_{max}(z^k \mid \mathbf{x}), \end{aligned} \tag{3}$$

where p_{hit} is a normal distribution $\mathcal{N}(0, \sigma^2)$ that models situations in which the sensor detects an obstacle. Random measurements are modeled using a uniform distribution $p_{rand}(z^k \mid \mathbf{x})$. Maximum range measurements are covered by a point mass distribution $p_{max}(z^k \mid \mathbf{x})$. These three distributions are weighted by the non-negative parameters α_{hit}, α_{rand}, and α_{max}, which sum up to one. The values for α_{hit}, α_{rand}, α_{max}, and σ^2 used in our current implementation have been determined empirically.

In the end point model, the probability $p_{hit}(z_k \mid \mathbf{x})$ only depends on the distance d^k between the end point of the k-th laser beam and the closest obstacle in the map. Figure 3 shows an example of a single beam z^k which ends close to vertical object in the MLS map. Thus, the physical property of the laser beam is ignored, because the model just uses the end point and does not consider the beam characteristic of the laser. Therefore, we need to calculate the global coordinates for a beam end point. If we denote the angle of the k-th beam relative to the zero angle with ζ^k, then the end point $\tilde{\mathbf{p}}^k = (\tilde{x}^k, \tilde{y}^k, \tilde{z}^k)^T$ of that beam in the robot's own coordinate frame is calculated as

$$\begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \\ \tilde{z}^k \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} + R z^k \begin{pmatrix} \cos(\zeta^k) \\ \sin(\zeta^k) \\ 0 \end{pmatrix}, \tag{4}$$

where $(\hat{x}, \hat{y}, \hat{z})^T$ denotes the position of the sensor at time t and R is a rotation matrix that expresses the 3D sensor

orientation in the robot's coordinate frame. For a given robot pose $\mathbf{x} = (x, y, z, \varphi, \vartheta, \psi)$ at time t we can compute the global coordinates $\mathbf{p}^k = (x^k, y^k, z^k)^T$ of the k-th beam end point $\mathbf{p}^k$ as follows

$$\begin{pmatrix} x^k \\ y^k \\ z^k \end{pmatrix} = R(\varphi, \vartheta, \psi) \begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \\ \tilde{z}^k \end{pmatrix} + \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \tag{5}$$

where $R(\varphi, \vartheta, \psi)$ denotes the rotation matrix for the given Euler angles φ, ϑ, and ψ. In MLS maps, obstacles are represented as *vertical surface patches*, which can be seen as vertical segments of occupied space. Unfortunately, there is no efficient way to find the closest of all vertical segments to a given beam end point. Therefore, we use an approximation by uniformly sampling a set $\mathcal{P}$ of 3D points from all vertical patches. The distance d^k of the k-th beam end point $\mathbf{p}^k$ to the closest obstacle is then approximated as the Euclidean distance $d(\mathbf{p}^k, \mathcal{P})$ between $\mathbf{p}^k$ and $\mathcal{P}$. This distance can be efficiently calculated by storing all points from $\mathcal{P}$ in a kD-tree.

Equations. (4) and (5) describe a 3D transform $T(z^k; \mathbf{x})$ of the measurement z^k at position $\mathbf{x}$. Using this and the fact that p_{hit} is Gaussian, we can compute p_{hit} as

$$p_{hit}(z^k \mid \mathbf{x}) \approx \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{d(\mathbf{p}^k, \mathcal{P})}{\sigma}\right)^2\right), \tag{6}$$

where $\mathbf{p}^k = T(z^k; \mathbf{x})$. Plugging this into Eqn. (3) and the result into Eqn. (2), we obtain the entire sensor model.

V. Autonomous Exploration in Three-dimensional Environments

The previous section covered the problem of localizing a vehicle in a MLS map. In this section, we relax the assumption that such a model is provided and present an approach to autonomously learn a MLS map with our mobile robot.

In order to autonomously explore the environment, we first need to perform a traversability analysis, thereby avoiding positive and negative obstacles. Then we determine candidate viewpoints in the vicinity of unexplored areas and evaluate those candidate viewpoints by considering the travel costs to a particular viewpoint and the expected information gain of a measurement at this viewpoint.

A. Traversability Analysis

A grid based 2D traversability analysis usually only takes into account the occupancy probability of a grid cell – implicitly assuming an even environment with only positive obstacles. In the 3D case, especially in outdoor environments, we additionally have to take into account the slope and the roughness of the terrain, as well as negative obstacles such as abysms which are usually ignored in 2D representations.

Each patch p will be assigned a traversability value $\tau(p) \in [0, 1]$. A value of zero corresponds to a non-traversable patch, a value greater zero to a traversable patch, and a value of one to a perfectly traversable patch. In order to determine $\tau(p)$, we fit a plane into its local 8-patch neighborhood by minimizing the z-distance of the plane to the elevation values of the neighboring patches. We then compute the slope and the roughness of the local terrain and detect obstacles. The slope is defined as the angle between the fitted plane and a horizontal plane and the roughness is computed as the average squared z-distances of the height values of the neighboring patch to the fitted plane. The slope and the roughness are turned into traversability values $\tau_s(p)$ and $\tau_r(p)$ by linear interpolation between zero and a maximum slope and roughness value respectively. In order to detect obstacles we set $\tau_o(p) \in \{0, 1\}$ to zero, if the maximum squared z-distance of a neighboring patch exceeds a threshold, thereby accounting for positive and negative obstacles, or if the patch has less than eight neighbors. The latter is important for avoiding abysms in the early stage of an exploration process, as some neighboring patches are below the edge of the abysm and therefore are not visible yet.

The combined traversability value is defined as $\tau(p) = \tau_s(p) \cdot \tau_r(p) \cdot \tau_o(p)$. Next, we iteratively propagate the values by convolving the traversability values of the patch and its eight neighboring patches with a Gaussian kernel. For non-existent neighbors, we assume a value of 0.5. The number of iterations depends on the used cell size, the robot's size and a safety margin. In order to enforce obstacle growing, we do not perform a convolution if one of the neighboring patches is non-traversable ($\tau = 0$), but rather set the patch's traversability directly to zero in this case.

B. Viewpoint Generation

We follow the popular frontier-based approach to exploration [24] and adapt it to the needs of a 3D environment. In our approach, a patch is considered as explored if it has eight neighbors and its uncertainty, measured by the entropy in the patch, is below a threshold. Additionally, we track the entropy as well as the number of neighbors of a patch. If the entropy or number of non-existing neighbors cannot be reduced as expected over several observations, we consider it to be explored nonetheless since further observations do not seem to change the state of the patch.

A frontier patch is defined as an unexplored patch with at least one explored neighboring patch. Most of these patches have less than eight neighbors and therefore are considered as non-traversable, since they might be at the edge of an abysm. Therefore, we cannot drive directly to a frontier patch. Instead, we use a 3D ray-casting technique to determine close-by candidate viewpoints. A patch is considered as a candidate viewpoint, if it is reachable and there is at least one frontier patch that is likely to be observable from that viewpoint. Instead of using ray-casting to track emitted beams from the sensor at every reachable position, we use a more efficient approach. We emit virtual beams from the frontier patch instead and then select admissible sensor locations along those beams (Figure 4). This will reduce the number of needed ray-casting operations as the number of frontier patches is much smaller than the number of reachable patches.

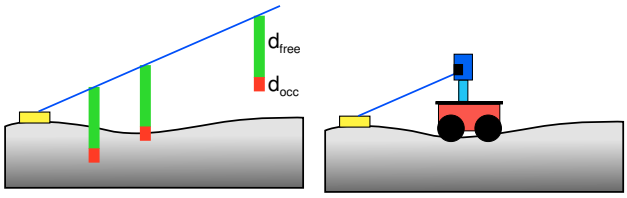


Fig. 4. To generate viewpoints, we emit laser beams from viewpoints and determine admissible sensor positions along those beams. The interval d_{free} needs to be free and the interval d_{occ} has to contain a reachable patch.

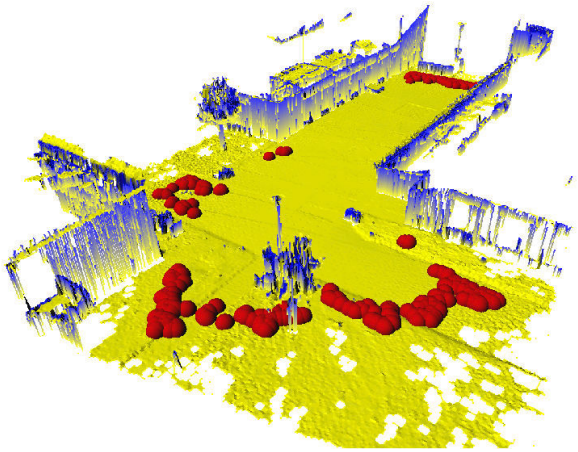


Fig. 5. Outdoor map showing sampled candidate viewpoints as red (dark gray) spheres.

In practice, we found it useful to reject candidate viewpoints, from which the expected information gain is below a threshold. We also cluster the frontier patches by the neighboring relation, and prevent patches from very small frontier clusters to generate candidate viewpoints. This will lead to a more reliable termination of the exploration process. Candidate viewpoints of an example map are shown in Figure 5.

C. Viewpoint Evaluation

The utility $u(v)$ of a candidate viewpoint v, is computed using the expected information gain $E\{I(v)\}$ and the travel costs $t(v)$. As the evaluation involves a costly 3D ray-casting operation, we reduce the set of candidate viewpoints by sampling uniformly a fixed number of viewpoints from that set.

In order to simultaneously determine the shortest paths to all candidate viewpoints, we use a deterministic variant of the value iteration. The costs of moving from a patch p to p' can be defined as

$$c(p, p') = d(p, p') + w(1 - \tau(p')) \tag{7}$$

where $d(p, p')$ describes the Euclidian distance and $\tau(p')$ the traversability of p'. The constant w is used to weight the penalization for traversing poorly traversable patches. The travel costs $t(v)$ of a viewpoint v is defined as the accumulated step costs of the shortest path to that viewpoint.

The expected information gain considers the uncertainty reduction in the known parts of the map as well as the information gain caused by new patches that are expected to be discovered.

To determine the patches that are likely to be hit by a laser measurement, we first perform a ray-cast operation similar to [19]. We determine the intersection points of the cell boundaries and the 3D ray projected onto the 2D grid. In a second step, we determine for each cell the height interval covered by the ray and check for collisions with patches contained in that cell by considering their elevation and depth values.

Let the sequence $L = \langle l_1, \ldots, l_m \rangle$ be an equidistant discretization of the maximum laser range. If the simulated laser ray hits a patch in distance that falls into l_h, we can divide L into three subsequences L^f, L^h, and L^n, whereas $L^f = \langle l_1, \ldots, l_{h-1} \rangle$ contains the collision free traversed distances, $L^h = \langle l_h \rangle$ contains the above mentioned discretized distance to the patch that has been hit, and $L^n = \langle l_{h+1}, \ldots, l_m \rangle$ contains the non-traversed distances. Accordingly, if the simulated ray does not hit a patch, this will result in three subsequences $L^f = L$ and $L^h = L^n = \langle\rangle$. For each traversed distance $l \in L^f \cup L^h$ we expect the ray during a real measurement to end after distance l with probability $p(l)$. If $l \in L^f$, then this corresponds to the discovery of a new patch, which implies an information gain $I_f(l)$. If $l \in L^h$, then this corresponds to a measurement of an already known patch, which implies an information gain $I_h(l)$. The expected information gain of ray r then is defined as

$$E\{I(r)\} = \sum_{l \in L} p(l) I(l) = \sum_{l \in L^f} p(l) I_f(l) + \sum_{l \in L^h} p(l) I_h(l). \tag{8}$$

Here we assume $p(l) = 0$ for $l \in L^n$, as we do not expect the ray to travel through a known patch.

To assess the probabilities $p(l)$, we created statistics through simulated measurements in a large outdoor map which yielded a conditional probability distribution $p_s(d \mid \alpha_v)$ denoting the probability of hitting an obstacle after distance d when the elevation angle of the ray is α_v. The intuition behind this is, that it is much more likely for downward pointing rays to hit a patch than for upward pointing rays. Secondly, the probability to hit an obstacle is not equally distributed along the laser range, especially not for downward pointing rays. Using this distribution, we can define

$$p(l) = \begin{cases} p_s(l \mid \alpha_v) & l \in L^f \\ \sum_{l_i \in L^h \cup L^n} p_s(l_i \mid \alpha_v) & l \in L^h \\ 0 & l \in L^n \end{cases} \tag{9}$$

with α_v being the elevation angle of the current ray r.

The information gain I_h is defined by the uncertainty reduction in the known map. We therefore temporary add a new measurement m_h into the grid cell of the hit patch p_h with a corresponding mean and variance that depends on the distance l_h of the simulated ray. The mean and variance of

the patch p_h will then be updated by using a Kalman filter. As a patch is represented as a Gaussian, we can compute the entropy $H(p)$ of a patch as

$$H(p) = \frac{1}{2} \log \left(2e\pi\sigma^2\right). \quad (10)$$

The information gain $I_h(l)$ is then defined as the difference

$$I_h(l) = H(p_h) - H(p_h \mid m_h) \quad l \in L^h. \quad (11)$$

between the entropy $H(p_h)$ of the patch p_h before and the entropy $H(p_h \mid m_h)$ after the temporary incorporation of the simulated measurement m_h.

For the information gain I_f we will proceed similarly. As a newly discovered patch p_f will be inserted with an uncertainty σ proportional to the distance $l \in L^f$ of measurement m_f, we can thereby compute $H(p_f \mid m_f)$ as in Eqn. 10. We assume that the uncertainty σ_b of the patch before it has been measured, is bounded by the distance d_p to the nearest patch in that cell and choose, as a heuristic, an uncertainty so that $3\sigma_b = d_p$. Using σ_b we can define $H(p_f)$ and finally compute

$$I_f(l) = H(p_f) - H(p_f \mid m_f) \quad l \in L^f. \quad (12)$$

The expected information gain $E\{I(v)\}$ of a viewpoint v is then defined as the sum $E\{I(v)\} = \sum_{r \in R} E\{I(r)\}$ of the expected information gains of all casted rays $r \in R$.

Finally, the utility $u(v)$ of each candidate viewpoint is computed by a relative expected information gain and travel costs as

$$u(v) = \alpha \frac{E\{I(v)\}}{\max_x E\{I(x)\}} + (1-\alpha) \frac{\max_x t(x) - t(v)}{\max_x t(x)}. \quad (13)$$

By varying the constant $\alpha \in [0,1]$ one can alter the exploration behavior by trading off the travel costs and the expected information gain.

D. Overlap

As explained before, we choose the viewpoint with the best utility as the next goal point. However, to ensure that we can construct a globally consistent map, we have to continously track the position of the vehicle. We construct a network of constraints between poses according to the observations. We then apply an efficient global optimization approach [10], [11] to correct the poses.

To ensure that the relations between poses can be accurately determined, a certain overlap between consecutive 3D scans is required. We perform several 3D scans along the way to ensure this sufficient overlap. We use the 3D ray-casting technique to simulate a 3D scan and estimate the overlap of a real scan at each patch p_i of the planned path $\langle p_1, \ldots, p_n \rangle$. The estimated overlap $\hat{o}(p_i) = r_l/|R|$ is ratio of the number of rays r_l that hit a patch of the last local map to the number of all casted rays $|R|$ for a simulated scan at patch p_i. The patch p_i with the highest index $i \in \{1, \ldots, n\}$ whose overlap $\hat{o}(p_i)$ is above a threshold is chosen as the subgoal for the next 3D scan.

Based on the map estimate so far, we apply the localization approach described in the previous chapter. Based on the most likely pose reported by the localization module, we perform scan-matching to refine the estimate. The relation between poses that are determined in this way are then added to the constraint network. The exploration ends, if the set of candidate viewpoints is empty.

VI. Experiments

In this section, we present experiments designed to illustrate the properties of the presented techniques as well as their advantages compared to other techniques. First, we present experiments that evaluate the GPS-free localization approach using laser range finder only. Then, we investigate the properties of our uncertainty-driven exploration approach.

A. Localization

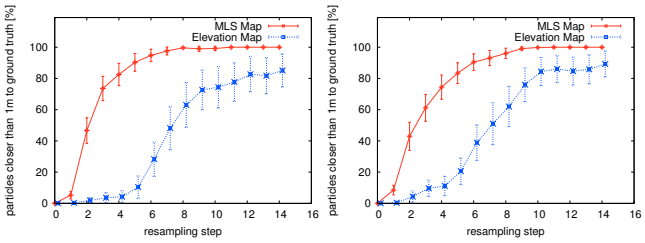


Fig. 6. Convergence of the particles to the true position of the robot with 500,000 (left) and 1,000,000 (right) particles. The x-axes depict the number of resampling steps, while the y-axes show the percentage of particles that are closer than $1m$ to the true position.

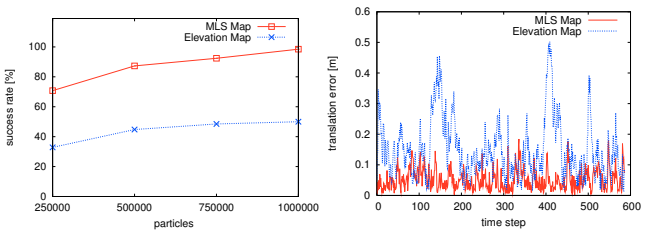


Fig. 7. The left image depicts the number of successful localizations after 15 resampling steps for the two different map representations for particle numbers from 250,000 up to 1,000,000. The right image shows the average localization error over all particles for a tracking experiment with 1,000 particles. In average the use of the MLS maps leads to smaller errors.

The first set of experiments is designed to evaluate the performance of the MLS map approach in the context of a global localization task. Figure 6 depicts the convergence of the particles to the true position of the robot with $500,000$ and $1,000,000$ particles. Whereas the x-axis corresponds to the resampling step, the y-axis shows the number of particles in percent that are closer than $1m$ to the true position, which has been computed by a tracking experiment with $100,000$ particles. Shown are the evolutions of these numbers when the MCL is applied on standard elevation maps and on MLS maps. Note that the elevation map does not reach 100%. This is due to the fact that the sensor model for the standard elevation map relies on a highly smoothed likelihood function, which is good for global localization but does not achieve maximal accuracy during tracking. The application

of a more peaked sensor model in the case of the standard elevation map would lead to much higher divergence rates. In both cases, a t-test showed that it is significantly better to apply the MLS maps than the standard elevation maps for the global localization task. Experiments with 250,000 and 750,000 particles showed the same behavior. The left image of Figure 7 shows the number of successful localizations for the two different map representations and for different numbers of particles. Here, we assumed that the localization was achieved when every particle differed by at most $1m$ from the true location of the robot. We can see that the global localization performs more robust on the MLS map than on the standard elevation map.

Fig. 8. MLS map used for the localization experiments. The area represented by this map spans approximately 195 by 146 meters. The blue / dark gray line shows the localized robot poses. The yellow / light gray line shows the pure odometry. The traversed trajectory has a length of 284 meters.

As a second set of experiments we carried out experiments, in which we analyzed the accuracy of the MLS map approach in the context of a position tracking task. To obtain the corresponding data set, we steered along a loop in our campus environment. The traversed trajectory has a length of 284 meters. Figure 8 depicts a top view of the MLS map of our test environment. The blue / dark gray line shows the localized robot poses. The yellow / light gray line shows the pure odometry. The right image of Figure 7 depicts the average localization error for a tracking experiment with 1,000 particles. As can be seen from the figure, the MLS map

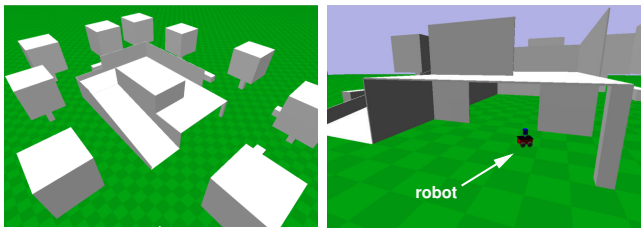


Fig. 9. Overview of the simulation environment and a detailed view of the entrance on the first floor with the robot in front of it.

approach outperforms the standard elevation map approach. The tracking experiments have been computed online on a standard PC with an AMD Athlon 64 3200+ processor. In the practical experiments we found that the use of the MLS maps results in a computational overhead of no more than 10% compared to elevation maps.

B. Exploration

The first exploration experiment is designed to show the ability of our exploration technique to take full advantage of the capabilities that MLS maps provide, e.g. representing multiple surface layers on top of each other. In a simulation environment with realistic rigid body physics we constructed a two-story building (Figure 9). It consists of two rooms located on top of each other, each 12 by 8 meters in size, and an unsecured balcony, where the robot is initially located. The house is surrounded by some trees and bushes, which are approximated by cuboids. We restricted the location of possible viewpoints to a rectangular area around the house in order to focus on the exploration of the house rather than the free space around the house. The robot explored the balcony, traversed the upper room and proceeded down a ramp that connects the upper room with the ground floor. The robot drove around the house and then entered the entrance to the room in the first floor. During the exploration of the lower room several 3D loops with positions at the upper room have been closed. He then visited a last viewpoint at the back of the house and then the exploration ended. The robot visited 18 viewpoints, performed 29 3D scans and traveled a distance of 212 meters. The final map consists of 185,000 patches. We demonstrated with this experiment, that we are able to deal with several challenges that simple mapping approaches are not able to deal with, e.g. negative obstacles and multiple surface layers. A 2D approach would simply have fallen down the unsecured balcony, and simple 3D mapping approaches like, for example, elevation maps, would not support the exploration of the two storys on top of each other. Figure 10 shows the constructed map with a detailed view of the entrance to the lower room.

To demonstrate the ability to explore real environments, we performed an experiment on the campus of the University of Freiburg using an ActivMedia Pioneer 2-AT equipped with a SICK laser range scanner mounted on a pan-tilt unit. To give the exploration an initial direction, we restricted the generation of viewpoints to the half-plane in front of the initial location of the robot. The robot followed a path

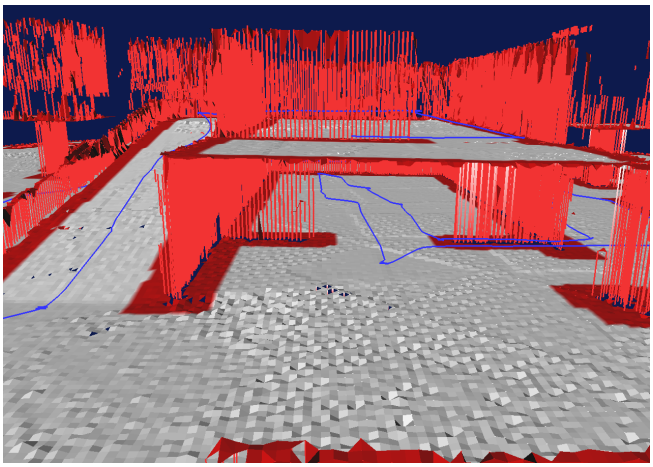


Fig. 10. Detailed view of the final traversability map showing the robot's trajectory as a blue line.

bordered by the wall of a house on the left side and grassland on the right side (Figure 11). Then he entered a small courtyard on the left, which was sufficiently explored after a few scans. He then proceeded to explore the rest of the campus until he reached the border of the defined half-plane. The figure shows four snapshots of the exploration process. In the last image, the robot traveled 186 meters, visited 18 viewpoints and performed 26 3D scans. The corresponding map including the traversability information contains about 410,000 patches and is depicted in Figure 12. In both experiments, we set $\alpha = 0.5$ in order to equally consider the travel costs and expected information gain.

VII. Conclusion

In this paper, we considered the problem of autonomously learning a three-dimensional model for combined outdoor and indoor environments with a mobile robot. We furthermore demonstrated how to localize a mobile vehicle based on such a model without requiring GPS information. Our approach uses proximity data from a laser range finder as well as odometry. Using our three-dimensional model of the environment, namely multi-level surface maps, we obtain significantly better results compared to elevation maps. We also presented an algorithm to actively acquire such maps from an unknown environment. This approach is decision-theoretic and trades off the cost of carrying out an action with the expected information gain of future observations. The approach also considers negative obstacles such as absyms which is an important prerequisite for robots operating in 3D environments.

Acknowledgment

This work has partly been supported by the DFG within the Research Training Group 1103 and under contract number SFB/TR-8, by the EC under contract number FP6-IST-34120-muFly, action line: 2.5.2.: micro/nano based subsystems, and under contract number FP6-004250-CoSy.

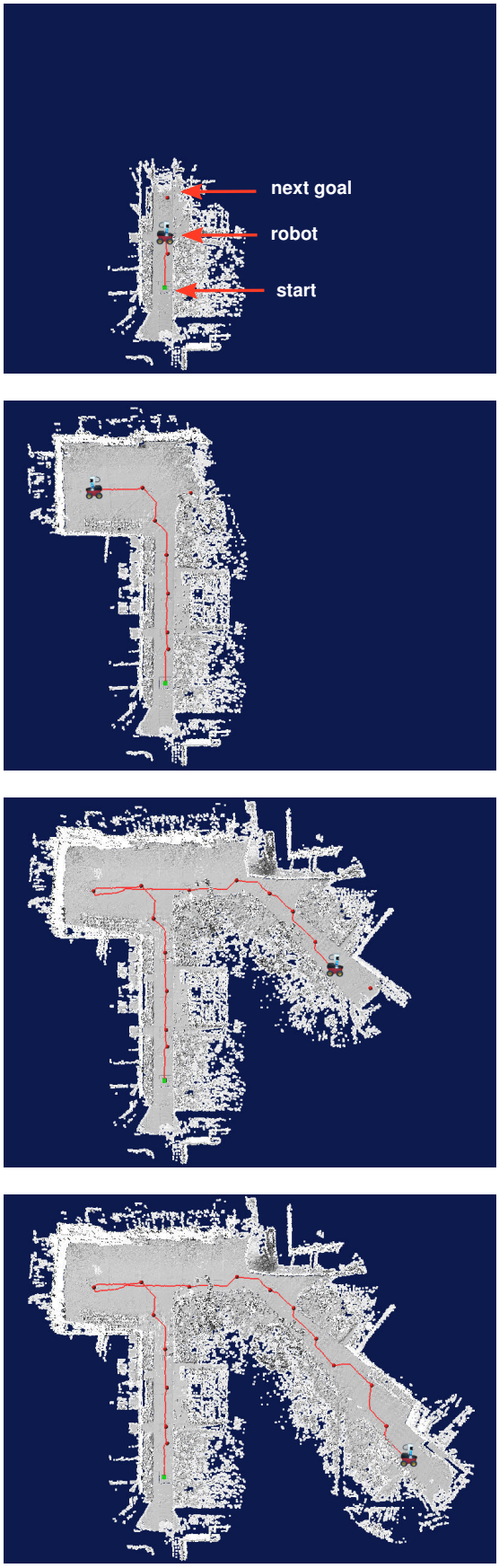


Fig. 11. Course of the exploration process on the university campus.

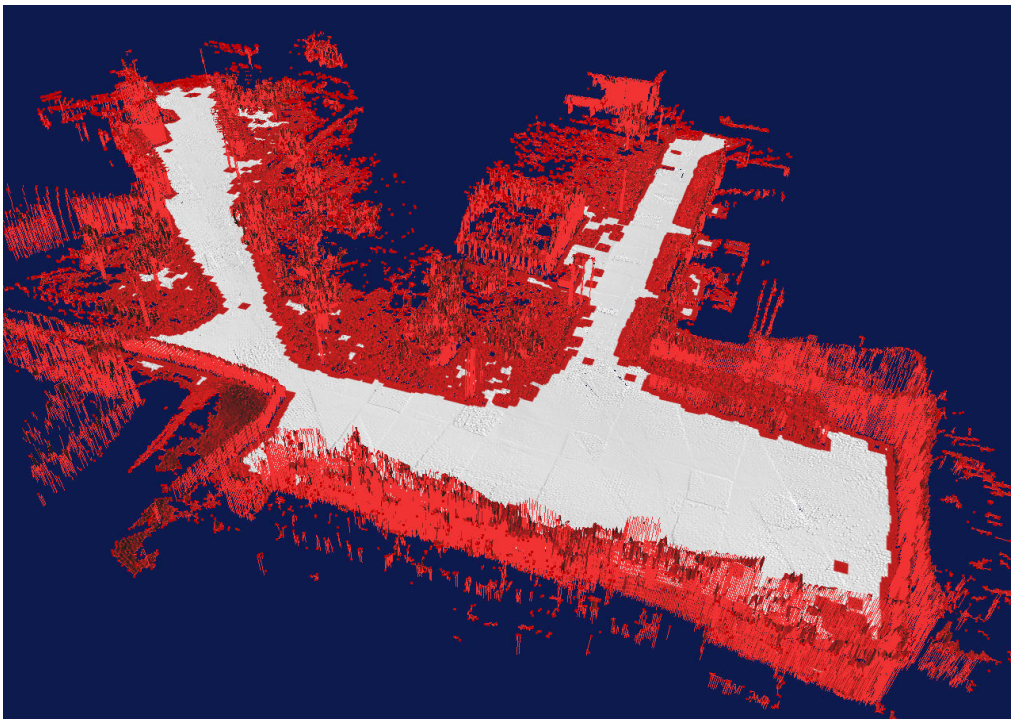


Fig. 12. Traversability map of the university campus.

REFERENCES

[1] M. Adams, S. Zhang, and L. Xie. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[2] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *International Conference on Pattern Recognition (ICPR)*, 2006.

[3] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. R. L. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer Society Press*, 22(6):18–22, 1989.

[4] A Davison and N. Kita. 3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Kauai*. IEEE Computer Society Press, December 2001.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[6] A. Doucet, N. de Freitas, and N. Gordan, editors. *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[7] A. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

[8] Jonathan Fournier, Benoit Ricard, and Denis Laurendeau. Mapping and exploration of complex environments using persistent 3D model. In *Proc. of the Canadian Conf. on Computer and Robot Vision (CRV)*, pages 403–410, Montreal, Canada, 2007.

[9] H.H. González-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *Int. Journal of Robotics Research*, 21(10-11):829–848, 2002.

[10] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007. To appear.

[11] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[12] M. Hebert, C. Caillas, E. Krotkov, I.S. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 997–1002, 1989.

[13] S. Koenig and C. Tovey. Improved analysis of greedy mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

[14] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury and; M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *Int. Journal of Robotics Research*, 21(10-11):917–942, 2002.

[15] K. Lingemann, H. Surmann, A. Nüchter, and J. Hertzberg. High-speed laser localization for mobile robots. *Journal of Robotics & Autonomous Systems*, 51(4):275–296, 2005.

[16] C.F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, 2000.

[17] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot. 3-d modelling and robot localization from visual and range data in natural scenes. In *1st International Conference on Computer Vision Systems (ICVS)*, number 1542 in LNCS, pages 450–468, 1999.

[18] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1127–1132, Acapulco, Mexico, 2003.

[19] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

[20] H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Journal of Robotics & Autonomous Systems*, 45(3-4):181–198, 2003.

[21] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[22] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[23] P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

[24] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Second International Conference on Autonomous Agents*, pages 47–53, Minneapolis, MN, USA, 1998.

[W2] H. Strasdat, C. Stachniss, M. Bennewitz, and W. Burgard. Visual bearing-only simultaneous localization and mapping with improved feature matching. In *Fachgespräche Autonome Mobile Systeme (AMS)*, Kaiserslautern, Germany, 2007.

Visual Bearing-Only Simultaneous Localization and Mapping with Improved Feature Matching

Hauke Strasdat, Cyrill Stachniss, Maren Bennewitz, and Wolfram Burgard

Computer Science Institute, University of Freiburg, Germany

Abstract. In this this paper, we present a solution to the simultaneous localization and mapping (SLAM) problem for a robot equipped with a single perspective camera. We track extracted features over multiple frames to estimate the depth information. To represent the joint posterior about the trajectory of the robot and a map of the environment, we apply a Rao-Blackwellized particle filter. We present a novel method to match features using a cost function that takes into account differences between the feature descriptor vectors as well as spatial information. To find an optimal matching between observed features, we apply a global optimization algorithm. Experimental results obtained with a real robot show that our approach is robust and tolerant to noise in the odometry information of the robot. Furthermore, we present experiments that demonstrate the superior performance of our feature matching technique compared to other approaches.

1 Introduction

Mapping is one of the fundamental problems in mobile robotics since representations of the environment are needed for a series of high level applications. Without an appropriate model of the environment, for example, delivery tasks cannot be carried out efficiently. A large group of researchers investigated the so-called simultaneous localization and mapping (SLAM) problem. The majority of approaches focuses on proximity sensors to perceive the environment such as laser range finders, sonars, radars, or stereo vision cameras.

In this paper, we address the problem of learning maps using a mobile robot equipped with a single perspective camera only. Compared to a laser range finder, cameras have the advantage that they are cheap and lightweight. One of the problems, however, is the missing distance information to observed landmarks. This information is not provided by a perspective camera. We present a mapping system that can use this sensor setup to learn maps of the environment. Our approach applies a Rao-Blackwellized particle filter to maintain the joint posterior about the trajectory of the robot and the map of the environment. We furthermore present a novel method to establish the data association between features. It takes into account the individual feature descriptor vectors as well as spatial constraints. Our approach is able to compute the optimal matching between observed and already tracked features. To achieve this, we apply the Hungarian method which is an efficient global optimization algorithm. Experiments carried out with a real robot illustrate the advantages of our technique for learning maps with robots using a single perspective camera.

2 Related Work

Davison et al. [1,2] proposed a visual SLAM approach using a single camera that does not require odometry information. The system works reliable in room-size environments but is restricted in the number of landmarks it can handle. Landmarks are matched by looking back into the image at the expected region and by perfoming a local match. Sim et al. [3] use a stereo camera in combination with FastSLAM [4]. SIFT features [5] in both cameras are matched using their description vectors as well as the epipolar geometry of the stereo system. The matching between observations and landmarks is done using the SIFT descriptor only. In the bearing-only algorithm of Lemaire et al. [6], the feature depth is estimated using a mixture of Gaussians. The Gaussians are initialized along the first observation and they are pruned in the following frames.

3 Visual SLAM and Feature Matching

The joint posterior about the robot's trajectory and the map is represented by a Rao-Blackwellized Particle Filter (RBPF) similar to FastSLAM [4]. It allows the robot to efficiently model the joint posterior in a sampled fashion.

To obtain landmarks, we extract speeded-up visual features (SURF) [7] out of the camera images. These features are invariant to translation and scale. They can be extracted using a Fast-Hessian keypoint detector. The 64-dimensional feature descriptor vector d is computed using horizontal and vertical Haar wavelet responses. A rotational dependent version of SURF is used since the roll angle of the camera is fixed when it is attached to a wheeled robot.

In order to obtain spherical coordinates of a feature given its position in the image, we apply a standard camera model. In this way, pixel coordinates of detected keypoints are transformed into the azimuthal angle θ and the spherical angle ϕ. The distance ρ to the observed feature cannot be measured since we use only a monocular camera. The tuple (θ, ϕ) is referred to as bearing-only observation $\mathbf{z}$.

3.1 Observation Model

In this section, we assume that a map of 3D-landmark is given. Each landmark l is modeled by a 3D Gaussian (μ, Σ). Moreover, we assume data association problem between observed features and landmarks is solved. These assumption are relaxed in the subsequent sections.

For each particle k, each observation $\mathbf{z} = (\theta, \phi)^T$ perceived in the current frame is matched with a landmark $l \in M^{[k]}$, where $M^{[k]}$ is the map carried by particle k. For each complete assignment of the currently observed features to map features, an update of the Rao-Blackwellized particle filter is carried out.

In order to determine the likelihood of an observation $\mathbf{z}$ in the update step of the particle filter, we need to compute the predicted observation $\hat{\mathbf{z}}$ of landmark $l = (\mu, \Sigma)$ for particle k. To achieve this, we have to apply two transformations. First, we transform world coordinates $\mu = (\mu_1, \mu_2, \mu_3)^T$ into camera-centric coordinates $c = (c_1, c_2, c_3)^T$

using the function g. Afterwards, the camera-centric Cartesian point c is transformed into camera-centric spherical coordinates $\hat{\mathbf{z}} = (\hat{\rho}, \hat{\theta}, \hat{\phi})^T$ using the function h:

$$\hat{\mathbf{z}} = h(g(\mu_l, \mathbf{x}^{[k]})) \tag{1}$$

Here, $\mathbf{x}^{[k]}$ is the current pose of particle k. The corresponding measurement uncertainty Q is predicted using the Jacobean $G = h'(g'(\mu_l, x^{[k]}))$ as

$$Q = G \cdot \Sigma \cdot G^T + \mathrm{diag}(\sigma_\rho, \sigma_\theta, \sigma_\phi). \tag{2}$$

Here, Σ is the covariance matrix corresponding to landmark l, and σ_θ and σ_ϕ represent the uncertainty over the two spherical angles. The uncertainty over the depth σ_ρ is set to a high value in order to represent the bearing-only aspect of the update. The observation likelihood λ is based on a Gaussian model as

$$\lambda = |Q|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \begin{pmatrix} 0 \\ \theta - \hat{\theta} \\ \phi - \hat{\phi} \end{pmatrix}^T Q^{-1} \begin{pmatrix} 0 \\ \theta - \hat{\theta} \\ \phi - \hat{\phi} \end{pmatrix} \right). \tag{3}$$

Since the depth ρ to the observed feature is unknown, the pretended innovation $(\rho - \hat{\rho})$ is set to zero. We weight each particle k with respect to its observation likelihood λ.

Finally, the Kalman gain is calculated by $K = \Sigma \cdot G^T \cdot Q^{-1}$ so that the landmark (μ, Σ) can be updated using an *extended Kalman filter* (EKF) approach.

3.2 Depth estimation and Landmark Initialization

Although it is possible to integrate bearing-only observations into the RBPF, the full 3D information is necessary in order to initialize landmarks in a 3D map. We track features over consecutive frames and estimate the depth ρ of the features using discrete probability distributions similar to [1] but in a bottom-up manner. When a feature f is initially observed, a 3D ray is cast from the camera origin o towards the observed feature. Equally weighted bins $b^{[j]}$ – representing different distances $\rho^{[j]}$ – are distributed uniformly along this ray within a certain interval. This reflects the fact that initially the distance to the feature is unknown. To get an estimate about the depth of the features, they are tracked over consecutive frames (the next subsection explains the feature matching process). In case the initial feature f is matched with a feature $\bar{f}$ in the consecutive frame, the bins are projected back into that frame. They lie on the so-called *epipolar line* [8], the projection of the 3D ray into the image. The depth hypotheses $\rho^{[j]}$ are weighted according to the distance to the pixel location of feature $\bar{f}$ using a Gaussian model. Figure 1 illustrates the estimation process for two features in consecutive frames. As soon as the variance of the histogram $Var(\rho^{[j]})$ falls below a certain threshold, the depth is estimated by the weighted average over the histogram $\rho = \sum_j h^{[j]} \cdot \rho^{[j]}$. If it is not possible to initialize a landmark within n frames, the corresponding feature is discarded (here n=5).

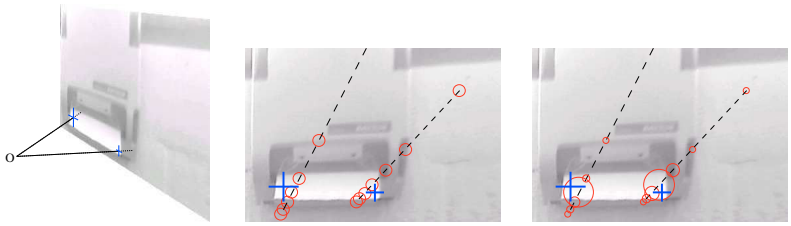


Fig. 1. This figure shows the depth estimation process for two features (crosses). Left: A ray is cast from the camera origin through the initial feature. Center: The ray is re-projected in the consecutive frame. This line (dashed) is called *epipolar line*. Depth hypotheses (circles) were distributed uniformly on the ray in the Cartesian space, which results in an irregular distribution in the image space. Right: Hypotheses are weighted according to their distance to the corresponding feature.

Depth Estimation as Preprocessing Step The robot's pose at the point in time, when the corresponding feature is observed initially, determines where the 3D ray is located in the world. Naïvely, for each particle a histogram of depth hypotheses has to be maintained so that the bins can be updated accordingly to the individual particle poses. However, this would lead to an overhead in computation time and memory. Fortunately, it is possible to maintain a depth histogram independently of the particles. The 3D ray is described by the angles θ, ϕ and an arbitrary origin o. Over the following n frames, the relative motion is added to o, so that the projection of the hypotheses' positions into the current frame can be calculated. Since the motion noise for wheeled robots is negligible within n frames, it can be omitted for the depth estimation process only.

Landmark Initialization Once a feature is reliably tracked and the depth of a feature is estimated, a landmark l is initialized. This has to be done for each particle k individually. Using the particle pose at the time t_f when feature f was observed for the first time, the global Cartesian landmark position μ can be calculated by

$$\mu_l := g^{-1}\left(h^{-1}\left(\rho_f, \theta_f, \phi_f\right), \mathbf{x}_{t_f}^{[k]}\right), \tag{4}$$

whereas the landmark uncertainty results from

$$\Sigma := G^{-T} \cdot H^{-1} \cdot R \cdot H^{-T} \cdot G^{-1}. \tag{5}$$

The uncertainty over the depth estimation process is reflected by the diagonal covariance matrix $R = \mathrm{diag}(Var(\rho_f^{[j]}), \sigma_\theta, \sigma_\phi)$.

3.3 Data Association

Finally, we describe how to match the current feature observations with landmarks in the map as well as with tracked features which are not yet contained in the map. This is done using the Hungarian method [9]. The Hungarian method is a general method to determine the optimal assignment under a given cost function. In our case, we use a cost function that takes into account differences of the feature descriptor vectors as well as well as spatial information to determine matches between observations and tracked features as well as between observations and landmarks.

Feature Matching Intuitively, features that are tracked to obtain the corresponding depth ρ can be matched based on their descriptor vectors using the Euclidean distance. However, this approach has a serious short-coming. Its performance is low on similar looking features since it completely ignores the feature positions. Thus, we instead use the distance of the descriptor vectors as a hard constraint. Only if the Euclidean distance falls below a certain threshold, a matching is considered. We define the cost function by means of the epipolar line introduced in Section 3.2. By setting the matching cost to the distance of the feature to this line in the image space (see Figure 2), not only the pixel locations of the comparing features are considered but also the relative movement of the robot between the corresponding frames is incorporated.

Landmark Matching Using Observation Likelihoods Similarly, during the matching process between landmarks and observations we use the distance of the descriptor vectors as a constraint. Landmarks are matched with observations using their positions. Since the observations are bearing-only, the distance to the landmark position cannot be computed directly. For this reason, the observation likelihood in Eq. (3) is used. It is high if and only if the distance between the observation and prediction is small. Thus, the cost is defined by the reciprocal of the observation likelihood $1/\lambda$. If the observation likelihood lies below a certain threshold, the cost are set to a maximum value. This refers to the fact that the features are regarded as different features with probability one.

4 Experimental Results

The first mapping experiment is performed on a wheeled robot equipped with a perspective camera and a laser range scanner (see Figure 3). The robot was steered through a 10m by 15m office environment for around 10 minutes. Two camera frames per second and odometry data was recorded. In addition, laser range data is stored in order to calculate a ground truth estimate of the robot's trajectory using scan matching on the laser data [10].

The results are illustrated in Figure 4. Following the presented approach, the average error of the robot path in terms of the Euclidean distance in the x/y-plane is 0.28m. The error in the orientation averages 3.9°. Using the odometry of the robot only, one obtains an average error of 1.69m in the x/y-plane and 22.8° in orientation.

We compared our feature matching approach using the Hungarian method on the distance to the projected line to other three techniques. Figure 5 shows a qualitative

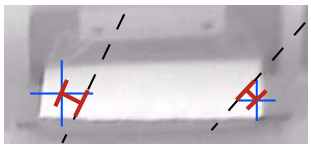


Fig. 2. Hungarian Matching: The cost function is set to the distance between the epipolar line and the feature location in the image space.

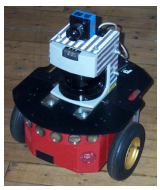


Fig. 3. Wheeled robot equipped with a perspective camera.

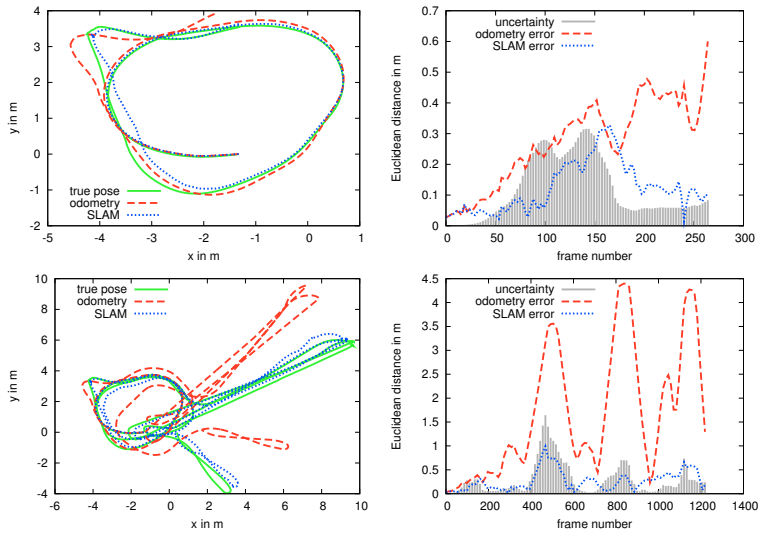


Fig. 4. The robot's trajectory is shown on the left, the corresponding error functions and uncertainty are shown on the right. Top: If the robot explores an unknown environment, the error values go up as well as the uncertainty. As soon as the loop is closed, the estimation error and uncertainty decreases, whereas the odometry error still goes up. Bottom: Complete trajectory.

evaluation of our approach on a difficult example – a heater which has a number of very similar looking features close to each other. Furthermore, we compare the Hungarian method quantitatively to the nearest neighbor approach, both using the distance to the epipolar line as cost function. If the Hungarian method is used, approximately 2% more landmarks are initialized. This number – obtained by four different sequences of 500 images each – can be explained by the fact that mismatches are likely to yield too high variances in the depth estimation. Landmarks, however, are initialized only if the depth can be estimated with low variance. By manual inspection, one can see that the data association has less errors than the nearest neighbor approach (see Figure 5).

5 Conclusions

In this paper, we presented a novel technique for learning maps with a mobile robot equipped with a single perspective camera only. Our approach applies a RBPF to maintain the joint posterior about the trajectory of the robot and the map of the environment. Using our approach, the robot is able to compute the optimal data association between observed and already mapped features by applying the Hungarian method. Experiments carried out with real a robot showed the effectiveness of our approach.

Acknowledgment

This work has partialy been supported by the German Research Foundation (DFG) under the contract numbers SFB/TR-8 and BE 2556/2-1. Special thanks to Dieter Fox,

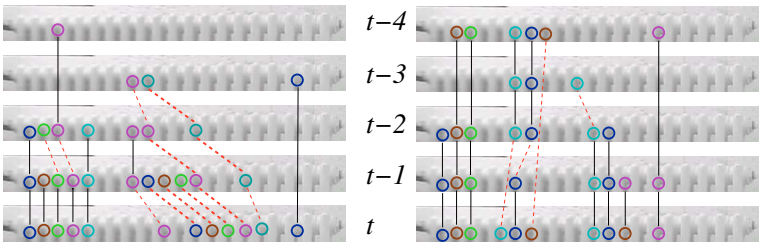


Fig. 5. Starting from the current frame at time t, we look back to evaluate how many features in the current frame were reliable tracked over the last four frames. The nearest neighbor assignment on SURF descriptors (left) results in 10 matches and 12 mismatches, whereas our approach results in 20 matches and 4 mismatches (right). Our approach also outperforms the two other combinations: nearest neighbor assignment using the projected line (14 matches, 7 mismatches) and the Hungarian method on the SURF descriptors (14 matches, 4 mismatches).

who supervised the first author in the early stages of developing the presented framework. We would like to thank Herbert Bay and Luc Van Gool for making the SURF binaries publicly available.

References

1. Davison A: Real-time simultaneous localization and mapping with a single camera. In Proc. of European Conf. on Computer Vision (ECCV), 2003.
2. Davision A, Reid I, Molton N, and Stasse O: MonoSLAM: Real-time single camera SLAM. IEEE Transaction on Pattern Analysis and Machine Intelligence 29(6), 2007.
3. Sim R, Elinas P, Griffin M, and Little J: Vision-based SLAM using a Rao-Blackwellized particle filter. In Proc. of IJCAI Workshop on Reasoning with Uncertainty in Robotics, 2005.
4. Montemerlo M, Thrun S, Koller D, and Wegbreit B: FastSLAM: A factored solution the to simultanieous localization and mapping problem. In Proc. of National Conference on Artificial Intelligence (AAAI), 2002.
5. Lowe D: Distinctive image feature from scale-invariant keypoints. In Proc. of International Journal of Computer Vision (IJCV), 2003.
6. Lemaire T, Lacroix S, and Sol J: A practical bearing-only SLAM algorithm. In Proc. of IEEE International Conf. on Intelligent Robots and Systems (IROS), 2005.
7. Bay H, Tuytelaars T, and Van Gool L: SURF: Speeded up robust features. In Proc. of European Conf. on Computer Vision (ECCV), 2006.
8. Hartley R and Zisserman A: Multiple View Geometry in Computer Vision. Cambirdige university press, second edition, 2003.
9. Kuhn H: The Hungarian method for the assigment problem. Naval Research Logistic Quaterly, 2:83-97, 1955.
10. Lu F and Milios E: Robot pose estimation in unkown environments by matching 2d range scans. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 935-938, 1994.

[W3] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous exploration for 3d map learning. In *Fachgespräche Autonome Mobile Systeme (AMS)*, Kaiserslautern, Germany, 2007.

Autonomous Exploration for 3D Map Learning

Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard

University of Freiburg, Department of Computer Science, Germany
{joho, stachnis, pfaff, burgard}@informatik.uni-freiburg.de

Abstract. Autonomous exploration is a frequently addressed problem in the robotics community. This paper presents an approach to mobile robot exploration that takes into account that the robot acts in the three-dimensional space. Our approach can build compact three-dimensional models autonomously and is able to deal with negative obstacles such as abysms. It applies a decision-theoretic framework which considers the uncertainty in the map to evaluate potential actions. Thereby, it trades off the cost of executing an action with the expected information gain taking into account possible sensor measurements. We present experimental results obtained with a real robot and in simulation.

1 Introduction

Robots that are able to acquire an accurate model of their environment are regarded as fulfilling a major precondition of truly autonomous mobile vehicles. So far, most approaches to mobile robot exploration assume that the robot lives in a plane. They typically focus on generating motion commands that minimize the time needed to cover the whole terrain [1,2]. A frequently used technique is to build an occupancy grid map since it can model unknown locations efficiently. The robot seeks to reduces the number of unobserved cells or the uncertainty in the grid map. In the three-dimensional space, however, such approaches are not directly applicable. The size of occupancy grid maps in 3D, for example, prevents the robot from exploring an environment larger than a few hundred square meters.

Whaite and Ferrie [3] presented an exploration approach in 3D that uses the entropy to measure the uncertainty in the geometric structure of objects that are scanned with a laser range sensor. In contrast to the work described here, they use a fully parametric representation of the objects and the size of the object to model is bounded by the range of the manipulator. Surmann et al. [4] extract horizontal planes from a 3D point cloud and construct a polygon with detected lines (obstacles) and unseen lines (free space connecting detected lines). They sample candidate viewpoints within this polygon and use 2D ray-casting to estimate the expected information gain. In contrast to this, our approach uses an extension of 3D elevation maps and 3D ray-casting to select the next viewpoint. González-Baños and Latombe [5] also build a polygonal map by merging safe regions. Similar to our approach, they sample candidate poses in the visibility range of frontiers to unknown area. But unlike in our approach, they build 2D maps and do not consider the uncertainty reduction in the known parts of the map.

The contribution of this paper is an exploration technique that extents known techniques from 2D into the three-dimensional space. Our approach selects actions that reduce the uncertainty of the robot about the world and constructs a full three-dimensional

model using so-called multi-level surface maps. It reasons about the potential measurements when selecting an action. Our approach is able to deal with negative obstacles like, for example, abysms, which is a problem of robots exploring a three-dimensional world. Experiments carried out in simulation and on a real robot show the effectiveness of our technique.

2 3D Model of the Environment

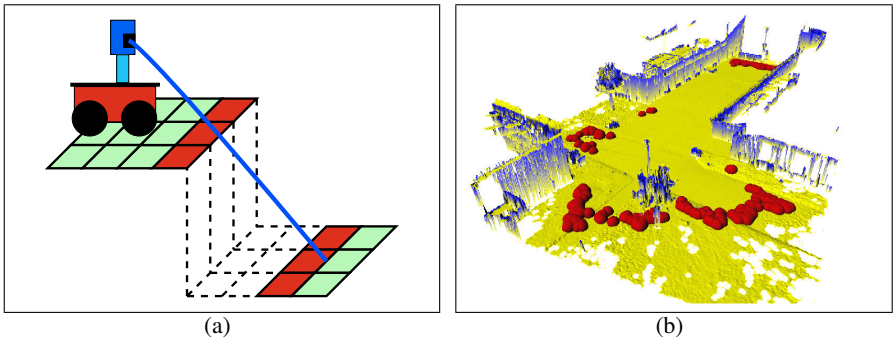
Our exploration system uses multi-level surface maps (MLS maps) as proposed by Triebel et al. [6]. MLS maps use a two-dimensional grid structure that stores different elevation values. In particular, they store in each cell of a discrete grid the height of the surface in the corresponding area. In contrast to elevation maps, MLS maps allow us to store multiple surfaces in each cell. Each surface is represented by a Gaussian with the mean elevation and its uncertainty σ. In the remainder of this paper, these surfaces are referred to as patches. This representation enables a mobile robot to model environments with structures like bridges, underpasses, buildings, or mines. They also enable the robot to represent vertical structures by storing a vertical depth value for each patch.

2.1 Traversability Analysis

A grid based 2D traversability analysis usually only takes into account the occupancy probability of a grid cell – implicitly assuming an even environment with only positive obstacles. In the 3D case, especially in outdoor environments, we additionally have to take into account the slope and the roughness of the terrain, as well as negative obstacles such as abysms which are usually ignored in 2D representations.

Each patch p will be assigned a traversability value $\tau(p) \in [0,1]$. A value of zero corresponds to a non-traversable patch, a value greater zero to a traversable patch, and a value of one to a perfectly traversable patch. In order to determine $\tau(p)$, we fit a plane into its local 8-patch neighborhood by minimizing the z-distance of the plane to the elevation values of the neighboring patches. We then compute the slope and the roughness of the local terrain and detect obstacles. The slope is defined as the angle between the fitted plane and a horizontal plane and the roughness is computed as the average squared z-distances of the height values of the neighboring patch to the fitted plane. The slope and the roughness are turned into traversability values $\tau_s(p)$ and $\tau_r(p)$ by linear interpolation between zero and a maximum slope and roughness value respectively. In order to detect obstacles we set $\tau_o(p) \in \{0,1\}$ to zero, if the squared z-distance of a neighboring patch exceeds a threshold, thereby accounting for positive and negative obstacles, or if the patch has less than eight neighbors. The latter is important for avoiding abysms in the early stage of an exploration process, as some neighboring patches are below the edge of the abysm and therefore are not visible yet (see Fig. 1 (a)).

The combined traversability value is defined as $\tau(p) = \tau_s(p) \cdot \tau_r(p) \cdot \tau_o(p)$. Next, we iteratively propagate the values by convolving the traversability values of the patch and its eight neighboring patches with a Gaussian kernel. For non-existent neighbors, we assume a value of 0.5. The number of iterations depends on the used cell size and the robot's size. In order to enforce obstacle growing, we do not perform a convolution



(a) (b)

Fig. 1. (a) While scanning at an abysm, some of the lower patches will not be covered by a laser scan (dashed area). Since the patches at the edge of the abysm have less than eight neighbors, we can recognize them as an obstacle (red / dark gray area). **(b)** Outdoor map showing sampled candidate viewpoints as red (dark gray) spheres.

if one of the neighboring patches is non-traversable ($\tau = 0$), but rather set the patch's traversability directly to zero in this case.

3 Our Exploration Technique

An exploration strategy has to determine the next viewpoint the robot should move to in order to obtain more information about the environment. Identifying the best viewpoint is a two step procedure in our system. First, we define the set of possible viewpoints or candidate viewpoints. Second, we evaluate those candidates to find the best one.

3.1 Viewpoint Generation

One possible definition of the set of candidate viewpoints is that every reachable position in the map is a candidate viewpoint. However, this is only feasible if the evaluation of candidate viewpoints is computationally cheap. If the evaluation is costly, one has to settle for heuristics to determine a smaller set. A popular heuristic is the frontier approach [2] that defines candidate viewpoints as viewpoints that lie on the frontier between obstacle-free and unexplored areas. In our approach, a patch is considered as explored if it has eight neighbors and its uncertainty, measured by the entropy in the patch, is below a threshold. Additionally, we track the entropy as well as the number of neighbors of a patch. If the entropy or number of non-existing neighbors cannot be reduced as expected over several observations, we consider it to be explored nonetheless since further observations do not seem to change the state of the patch.

A frontier patch is defined as an unexplored patch with at least one explored neighboring patch. Most of these patches have less than eight neighbors and therefore are considered as non-traversable, since they might be at the edge of an abysm. Therefore, we cannot drive directly to a frontier patch. Instead, we use a 3D ray-casting technique to determine close-by candidate viewpoints. A patch is considered as a candidate

viewpoint, if it is reachable and there is at least one frontier patch that is likely to be observable from that viewpoint. Instead of using ray-casting to track emitted beams from the sensor at every reachable position, we use a more efficient approach. We emit virtual beams from the frontier patch instead and then select admissible sensor locations along those beams. This will reduce the number of needed ray-casting operations as the number of frontier patches is much smaller than the number of reachable patches.

In practice, we found it useful to reject candidate viewpoints, from which the unseen area is below a threshold. We also cluster the frontier patches by the neighboring relation, and prevent patches from very small frontier clusters to generate candidate viewpoints. This will lead to a more reliable termination of the exploration process. Candidate viewpoints of an example map are shown in Fig. 1 (b).

3.2 Viewpoint Evaluation

The utility $u(v)$ of a candidate viewpoint v, is computed using the expected information gain $I(v)$ and the travel costs $t(v)$. As the evaluation involves a costly 3D ray-casting operation, we reduce the set of candidate viewpoints by sampling uniformly a fixed number of viewpoints from that set.

In order to simultaneously determine the shortest paths to all candidate viewpoints, we use a deterministic variant of the value iteration [7]. The costs

$$c(p,p') = \mathrm{dist}(p,p') + w(1-\tau(p')) \tag{1}$$

from patch p to a traversable neighboring patch p' considers the distance $\mathrm{dist}(p,p')$, as well as the traversability $\tau(p')$. A constant factor w is used to weight the penalization for traversing poorly traversable patches. The travel costs $t(v)$ of a viewpoint v is defined as the accumulated step costs of the shortest path to that viewpoint.

In order to evaluate the information gain of a viewpoint candidate, we perform a ray-cast operation to determine the patches that are likely to be hit by a laser measurement similar to [8]. We therefore determine the intersection points of the cell boundaries and the 3D ray projected onto the 2D grid. Next we determine for each cell the height interval covered by the ray and check for collisions with patches contained in that cell by considering their elevation and depth values. Using a standard notebook computer, our approach requires around 25 ms to evaluate one potential viewpoint including the 3D ray-cast operation. This allows us to run our algorithm with minimal delays only for typical environments.

For each casted ray that hits a patch, we temporary add a new measurement into the patch's grid cell with a corresponding mean and variance that depends on the distance of the laser ray. The mean and variance of the patch will then be updated using the Kalman update. As a patch is represented as a Gaussian, we can compute the entropy $H(p)$ of the patch as

$$H(p) = \frac{1}{2}\left(1+\log\left(2\pi\sigma^2\right)\right). \tag{2}$$

The information gain $I(p)$ of a ray-cast is then defined as the difference between the entropy $H(p)$ of the patch before and the entropy $H(p \mid m)$ after the temporary incorporation of the simulated measurement

$$I(p) = H(p) - H(p \mid m). \tag{3}$$

Additionally, we add a constant value for each empty cell traversed by the ray. In this way, we reward viewpoints from which unseen areas are likely to be visible, while we are still accounting for the reduction of existing uncertainties in the known map. Rays that do not hit any patch and do not traverse any empty cells, will result in an information gain of zero. The information gain $I(v)$ of a viewpoint v is then defined as the sum of the information gains of all casted rays. Finally, the utility $u(v)$ of each candidate viewpoint is computed by a relative information gain and travel costs as

$$u(v) = \alpha \frac{I(v)}{\max_x I(x)} + (1-\alpha)\frac{\max_x t(x) - t(v)}{\max_x t(x)}. \quad (4)$$

By varying the constant $\alpha \in [0,1]$ one can alter the exploration behavior by trading off the travel costs and the information gain.

3.3 Localization and Termination

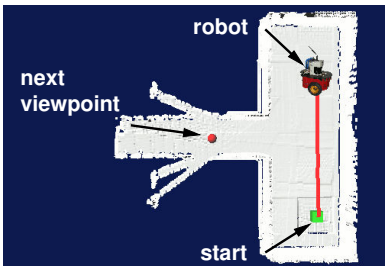
The registration of newly acquired information involves a scan matching procedure with the previous local map. We therefore cannot drive directly to the next viewpoint, as the resulting overlap with the previous local map may be to small. Hence, we perform several 3D scans along the way, which has the benefit, that it allows us to optimize the localization of the robot with the pose returned by the scan matcher. We apply a 6D Monte Carlo localization proposed by Kümmerle et al. [9]. After each 3D scan, we replan the path to the selected viewpoint. If the viewpoint is unreachable, we choose a new one. The exploration terminates if the set of candidate viewpoints is empty.

4 Experiments

The experiments described here are designed to illustrate the benefit of our exploration technique which is able to build three-dimensional models of the environment and takes into account the travel costs and the expected change in the map uncertainty to evaluate possible actions. For the real-world experiments we used an ActivMedia Pioneer2-AT robot with a SICK laser range finder mounted on a pan-tilt unit to acquire three-dimensional range data. For a 3D scan we tilt the laser in a range of 40 degrees at four equally spaced horizontal angles while acquiring the laser data.

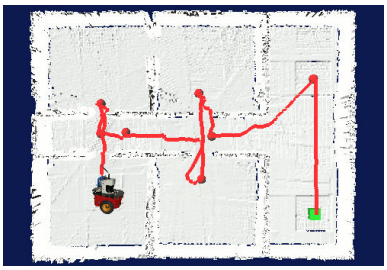
We tested our approach in simulation and in a real-world scenario. For the simulation experiments, we used a physical simulation environment that models our Pioneer robot with its pan-tilt unit. The simulated indoor environment consisted of four rooms, each connected to a corridor, and a foyer where the robot is located initially. The upper two rooms are connected directly through a door, while the lower ones are not. The robot efficiently covered the environment taking into account its constraints like travel cost, and information gain. The robot traveled 59 meters, visited eight viewpoints, and performed 15 scans (see Fig. 2 (a)-(c)). The final map, depicted in Fig. 2 (c) and (d), covers an area of $22m \times 17m$ and contains about 41,000 patches.

Real-world experiments have been carried out on the university campus. In the experiment shown in Fig. 2 (e)-(h) the robot traveled 84 meters, visited six viewpoints, and performed 23 scans. The map depicted in Fig. 2 (g) and (h) contains about 197,000

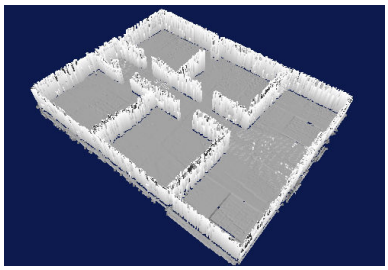


(a) Robot reached the first viewpoint.

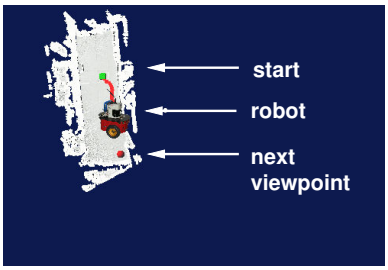
(b) Robot reached the fourth viewpoint.



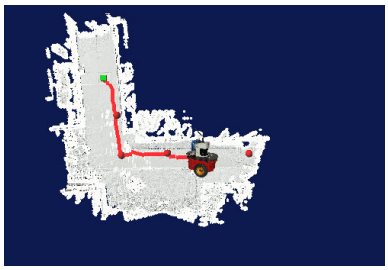
(c) Robot reached the final viewpoint.



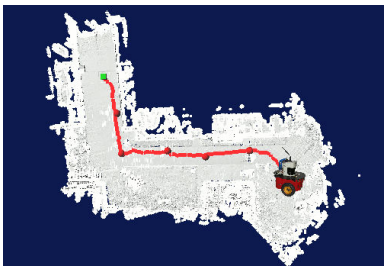
(d) Perspective view of the final map.



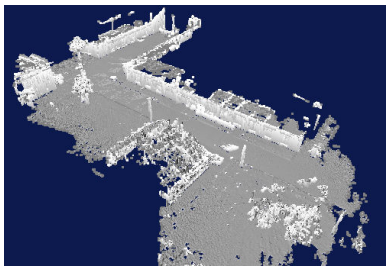
(e) Robot reached the first viewpoint.



(f) Robot reached the fourth viewpoint.



(g) Robot reached the sixth viewpoint.



(h) Perspective view of the map.

Fig. 2. (a)-(d) Exploration in a simulated indoor environment. One can see four rooms, a corridor, and the foyer where the robot started the exploration. **(e)-(h)** Real-world exploration in an outdoor scenario. One can see the walls of three buildings, the pitched roof of a green house, and several street lamps and trees.

patches and its bounding box roughly covers an area of about $70m \times 75m$. In both experiments, we set $\alpha = 0.55$ and used a cell size of $0.1m \times 0.1m$.

5 Conclusion

In this paper, we presented an approach to autonomous exploration for mobile robots that is able to acquire a three-dimensional model of the environment, which is compactly represented by a multi-level surface map. We addressed problems which are not encountered in traditional 2D representations such as negative obstacles, roughness, and slopes of non-flat environments. The viewpoint generation and evaluation procedure utilizes 3D ray-casting operations to account for the 3D structure of the environment. We applied a decision-theoretic framework which considers both the travel costs and the expected information gain to efficiently guide the exploration process. Simulation and real-world experiments showed the effectiveness of our technique.

6 Acknowledgment

This work has been partly supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and within the Research Training Group 1103.

7 References

1. Tovey C, Koenig S: Improved analysis of greedy mapping. Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2003.
2. Yamauchi B: Frontier-based exploration using multiple robots. Proc. of the Second Int. Conf. on Autonomous Agents 47–53, 1998.
3. Whaite P, Ferrie FP: Autonomous exploration: Driven by uncertainty. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(3):193–205, 1997.
4. Surmann H, Nüchter A, Hertzberg J: An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. Journal of Robotics and Autonomous Systems 45(3-4):181–198, 2003.
5. González-Baños HH, Latombe JC: Navigation strategies for exploring indoor environments. Int. Journal of Robotics Research 21(10-11):829–848, 2002.
6. Triebel R, Pfaff P, Burgard W: Multi-level surface maps for outdoor terrain mapping and loop closing. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2006.
7. Burgard W, Moors M, Stachniss C, et al.: Coordinated multi-robot exploration. IEEE Transactions on Robotics 21(3):376–378, 2005.
8. Stachniss C, Grisetti G, Burgard W: Information gain-based exploration using rao-blackwellized particle filters. Proc. of Robotics: Science and Systems (RSS) 65–72, 2005.
9. Kümmerle R, Triebel R, Pfaff P, et al.: Monte carlo localization in outdoor terrains using multi-level surface maps. Proc. of the Int. Conf. on Field and Service Robotics (FSR), 2007.

[W4] P. Lamon, C. Stachniss, R. Triebel, P. Pfaff, C. Plagemann, G. Grisetti, S. Kolski, W. Burgard, and R. Siegwart. Mapping with an autonomous car. In *Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

Mapping with an Autonomous Car

Pierre Lamon* Cyrill Stachniss* Rudolph Triebel† Patrick Pfaff†
Christian Plagemann† Giorgio Grisetti† Sascha Kolski* Wolfram Burgard† Roland Siegwart*

*Eidgenössische Technische Hochschule (ETH), Inst. of Robotics and Intelligent Systems, 8092 Zurich, Switzerland
†University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

***Abstract*— In this paper, we present an approach towards mapping and safe navigation in real, large-scale environments with an autonomous car. The goal is to enable the car to autonomously navigate on roads while avoiding obstacles and while simultaneously learning an accurate three-dimensional model of the environment. To achieve these goals, we apply probabilistic state estimation techniques, network-based pose optimization, and a sensor-based traversability analysis approach. In order to achieve fast map learning, our system compresses the sensor data using multi-level surface maps. The overall systems runs on a modified Smart car equipped with different types of sensors. We present several results obtained from extensive experiments which illustrate the capabilities of our vehicle.**

I. Introduction

Learning models of the environment and safely navigating based on that models is a fundamental task of mobile robots. Many researcher focused on learning map of indoor as well as outdoor scenes. Recently, modified cars became a new platform in robotics. Compared to standard robots, cars offer the possibility to travel longer distances, carry more sensors, and thus being more suitable for mapping large areas. On the one hand, this offers the opportunity to address robotic tasks on a larger scale but on the other hand requires efficient solutions to common problems like mapping or localization.

In this paper, we describe our modified Smart car equipped with different sensor to monitor the environment. We present our approach to mapping large outdoor areas with that vehicle. Our map representation can be seen as an extention of elevation maps that allows us to model different layers in the environment and therefore different drivable areas like, for example, bridges or underpasses. It overcomes serious limitations of elevation maps and is able to model the environment in an adequate way needing only a few more memory resources compared to elevation maps.

The reminder of this paper is organized as follows. After the discussion of related work, we present detail about our modified Smart car. Then, we will explain our approach to localization using different sensors. In Section V, we introduce our model of the environment and present a method to learn these model. Finally, we present experiments illustrating the maps obtained with our robot in real world experiments.

II. Related Work

The problem of learning models of the environment has been studied intensively in the past. Most approaches generate two-dimensional models from range sensor data and a series of different approach have been developed [22, 24, 37, 8, 7, 40, 11, 19]. In the literature, those approaches are often referred to as solutions to the simultaneous mapping and localization (SLAM) problem. Recently, several techniques for acquiring three-dimensional data with 2d range scanners installed on a mobile robot have been developed. A popular approach is to use multiple scanners that point towards different directions [41, 13, 42]. An alternative is to use pan/tilt devices that sweep the range scanner in an oscillating way [32, 26]. More recently, techniques for rotating 2d range scanners have been developed [17, 50].

Many authors have studied the acquisition of three-dimensional maps from vehicles that are assumed to operate on a flat surface. For example, Thrun *et al.* [41] present an approach that employs two 2d range scanners for constructing volumetric maps. Whereas the first is oriented horizontally and is used for localization, the second points towards the ceiling and is applied for acquiring 3d point clouds. Früh and Zakhor [10] apply a similar idea to the problem of learning large-scale models of outdoor environments. Their approach combines laser, vision, and aerial images. Furthermore, several authors have considered the problem of simultaneous mapping and localization (SLAM) in an outdoor environment [5, 12, 43]. These techniques extract landmarks from range data and calculate the map as well as the pose of the vehicles based on these landmarks. Our approach described in this paper does not rely on the assumption that the surface is flat. It uses surface maps to capture the three-dimensional structure of the environment and is able to estimate the pose of the robot in all six degrees of freedom.

One of the most popular representations are raw data points or triangle meshes [1, 20, 32, 44]. Whereas these models are highly accurate and can easily be textured, their disadvantage lies in the huge memory requirement, which grows linearly in the number of scans taken. Accordingly, several authors have studied techniques for simplifying point clouds by piecewise linear approximations. For example, Hähnel *et al.* [13] use a region growing technique to identify planes. Liu *et al.* [21] as well as Martin and Thrun [23] apply the EM algorithm to cluster range scans into planes. Recently, Triebel *et al.* [46] proposed a hierarchical version that takes into account the parallelism of the planes during the clustering procedure. An alternative is to use three-dimensional grids [27] or tree-based representations [34], which only grow linearly in the size of the environment. Still, the memory requirements for such maps in outdoor environments are high.

In order to avoid the complexity of full three-dimensional maps, several researchers have considered elevation maps as an

attractive alternative. The key idea underlying elevation maps is to store the $2\frac{1}{2}$-dimensional height information of the terrain in a two-dimensional grid. Bares *et al.* [2] as well as Hebert *et al.* [14] use elevation maps to represent the environment of a legged robot. They extract points with high surface curvatures and match these features to align maps constructed from consecutive range scans. Singh and Kelly [36] extract elevation maps from laser range data and use these maps for navigating an all-terrain vehicle. Ye and Borenstein [51] propose an algorithm to acquire elevation maps with a moving vehicle carrying a tilted laser range scanner. They propose special filtering algorithms to eliminate measurement errors or noise resulting from the scanner and the motions of the vehicle. Wellington *et al.* [49] construct a representation based on Markov Random Fields. They propose an environment classification for agricultural applications. They compute the elevation of the cell depending on the classification of the cell and its neighbors. Compared to these techniques, the contribution of the mapping approach presented in this paper lies in two aspects. First, we classify the points in the elevation map into horizontal points see from above, vertical points, and gaps. This classification is important especially when a rover is deployed in an urban environment. In such environments, typical structures like the walls of buildings cannot be represented in standard elevation maps. Second, we describe how this classification can be used to improve the matching of different elevation maps.

In the context of autonomous cars, a series of successful systems [45, 3, 48] have been developed for the DARPA Gran Challenge [4], which was a desert race for autonomous vehicles along an approximatively 130 mile course. As a result of this challenge, there exist autonomous cars that reliably avoid obstacle and navigate at comparably high speeds. The focus of the Gran Challenge was to finish the race as quickly as possible whereas certain issues like building consistent large-scale maps of the environment have been neglected since they where not needed for the race. Our approach towards mapping large areas therefore has a different aim compared to the vehicles participating in the Gran Challenge. Nevertheless, our Smart car also benefited from different techniques used within the Gran Challenge. We apply a similar approach to follow a given trajectory than the winning vehicle Stanley [45].

III. Vehicle Description

Our vehicle, called SmartTer (Smart all Terrain), is a standard Smart car that has been enhanced for fully autonomous driving in both urban and non-urban environments. The model is a Smart fortwo coupé passion of year 2005, which is equipped with a 45 kW engine. This model has been chosen because it gathers several advantages:

- *compact and light* Such characteristics allow us to easily transport the vehicle on a trailer to the testing area[1] and fits in our lab's mechanical workshop. Furthermore, its light weight yields fair locomotion performance in rough terrain.

[1]Because the car has been deeply modified, it is not allowed to drive on public streets.

- *power steering* The power steering motor can deliver enough torque to steer the car. So, it is possible to "steer by wire" with minor modification.
- *auto gearshift* No additional modification is required to switch gears while the car is driving.
- *easy access to the CAN bus* Important sensory information such as steering wheel angle and wheels velocities are directly accessible.

All these features facilitate the process of modifying such a vehicle for autonomous driving. The fully equipped SmartTer is depicted in Fig. 1 and 2.

Fig. 1. SmartTer front view

Fig. 2. SmartTer side view

A. Vehicle modifications

In order to enhance the original model for autonomous driving, several modifications have been performed on the car. This section describes the mechanical and electrical changes that were necessary.

• Wheels with better grip and larger diameter have been mounted. This yields to an higher ground clearance and much better traction in rough terrain.
• A 24V power generator has been installed in order to power all the electronic devices and additional actuators. The generator is driven by a belt and pulley that is directly connected to the engine output axis (which is situated under the trunk, at the rear of the car). Two batteries placed in the trunk act as an energy buffer. They have a total capacity of 48Ah and are continuously recharged when the engine is running.
• To provide a clean interface to the vehicle, we integrated an automotive ECU designed for highly reliable real-time applications. This ECU has four CAN interfaces as well as a wide variety of analog and digital I/O. In the vehicle, it is sitting between the computers on the one hand and the vehicle CAN bus and our actuators on the other hand. In this ECU, we implemented a state machine that allows us to enable different modes of operation (STOP, PAUSE and RUN) via wired buttons as well as by a wireless remote control. Besides these emergency buttons, the ECU handles timeouts in the command coming from the computers and ensures a safe vehicle state whenever those commands are missing. As this system is in aspects of both hardware and software a hard-real-time system.
• The power steering system applies a torque M_{add} on the steering column that allows to minimize the effort required by the driver to steer the wheels (see Fig. 3). This task is fulfilled by the driving assistance unit (DA unit), which minimize the torque sensed in the steering column by applying an appropriate voltage to the power steering motor. The gain of the DA controller is set based on the car's velocity, which is broadcasted on the CAN bus of the vehicle. In order to use the power steering motor for "steering by wire", a specific electronic board has been designed and inserted in the vehicle's control loop. The Vehicle CAN bus was disconnected from the DA and routed to a computer (Rack0 in Fig. 3) so that the steering angle a_s can be read and used by our own controller. An additional CAN bus, called Computer CAN, allows to feed the DA unit with the minimal set of CAN messages required for the proper operation of the unit i.e. engine on and car velocity messages. The same bus is used to send commands to a CAN to analog module which "fakes" the torque voltage needed by the DA unit. Finally, the steering angle is controlled with a PID controller running on the control computer which minimizes the steering angle error $e = a_t - a_s$, where a_t is the desired steering angle. A switch mounted next to the steering wheel, enables the selection of manual or controlled mode.
• A system of cable and pulleys is used to activate the break pedal. The servo motor that pulls the cable is placed under the driver's seat and is commanded using the Computer CAN.
• A dedicated electronic board has been developed to enable the use of a computer to set the gas command. The command voltage, originally provided by a potentiometer embedded in the gas pedal, is simply generated by a CAN to analog device. This device receives commands through the Computer CAN bus. A button mounted inside the car allows us to choose between normal or controlled mode.

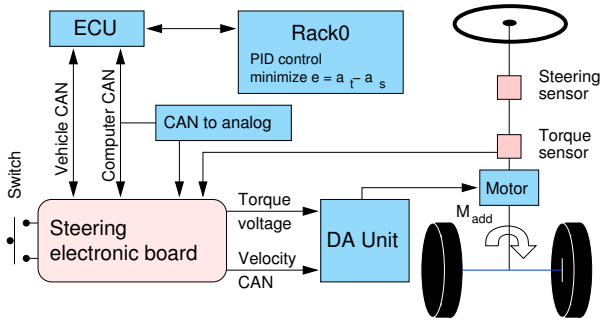


Fig. 3. Steer by wire system

B. Sensors

The sensors used for outdoor applications must meet strong requirements such as mechanical robustness, water/dust proofness and limited sensitivity to sun light. Thus, in comparison with indoor applications, the choice is limited and the selection of optimal sensors must be done carefully. In this section, all the sensors that have been mounted on the car are described.
• *Three navigation laser scanner sensors* (SICK LMS291-S05, outdoor version, rain proof, low sensitivity to sun light) These sensors are used mainly for obstacle avoidance and local navigation. One sensor is placed at the lower front slightly looking down and the two others on the roof, looking to the sides and slightly down. This 'A' shape configuration enables a large field of view and is well adapted to all kind of terrains. The three sensors are visible in Fig. 1 and Fig. 4 depicts a closer view of the sensors mounted on the roof.

Fig. 4. Sensors mounted on the roof

• *Rotating 3D scanner* This is a custom made sensor that is mounted on the roof (see Fig. 5). In order to acquire 3D scans of the environment, two SICK LMS291-S05 are mounted sideways on a plate rotating around the vertical axis. A signal is triggered each time the plate performs a full turn. That way it is possible to know its angular position at each time (using the rotational velocity of the motor and the elapsed time since the last trigger). The data and power lines of the two SICK

sensors go through a slip ring, which is mounted along the rotation axis. The 3D scans are mainly used to compute a consistent 3D digital terrain model of the environment.

Fig. 5. Rotating scanner and omnidirectional camera

• *Omnidirectional camera* (Sony XCD-SX910CR, focal length 12mm, with parabolic mirror and dust protection) The setup is mounted in front of the rotating scanner (see Fig. 5). It enables the acquisition of panoramic images that are used to supply texture information for the 3D digital terrain maps. The images are also exploited to detect artificial object in the scene.
• *Monocular camera* (Sony XCD-SX910CR, focal length 4.2mm) This camera is placed in the car, behind the wind shield. Like the omnidirectional camera, it is used to detect artificial object in the scene.
• *Differential GPS system* (Omnistar Furgo 8300HP) This device provides the latitude, longitude and altitude together with the corresponding standard deviation and the standard NMEA messages with a frequency of 5 Hz. When geostationary satellites providing the GPS drift correction are visible from the car, the unit enters the differential GPS mode (high precision GPS). When no correction signal is available, the device outputs standard precision GPS.
• *Car sensors* The measurements taken by the car sensors are reported with a frequency of 100 Hz and are accessible via the CAN bus of the vehicle. The car provides motor RPM, temperatures, steering wheel angle, wheel velocities, gas pedal position, and some further status information.
• *Optical gyroscope* (KVH DSP3000) This fibre optic gyroscope can measure very low rotation rates with a frequency of 100 Hz. It is possible to use it as a heading sensor for a comparably long period of time by integrating the angular rate. Contrary to compasses, the integrated heading is not sensitive to earth magnetic field disturbances. Finally, this unit offers much better accuracy than mechanical gyro and is not sensitive to shocks because it contains no moving parts.
• *Inertial measurement unit* (Crossbow NAV420, waterproof) This unit provides sensor data with a frequency of 100 Hz that contains the measurements from 3 accelerometers, 3 gyroscopes, a 3D magnetic field sensor and a GPS receiver. The internal digital signal processor of the unit combines the embedded sensors to provide the filtered attitude of the vehicle (roll, pitch, heading to true north) and the position (latitude, longitude and altitude). However, this sensor is not well adapted for a ground vehicle driving at low speed because of a bad signal/noise ratio of the inertial sensors. Furthermore, the earth magnetic field can be strongly distorted when the vehicle drives next to iron structures. This causes large error on the estimated heading. For better accuracy and robustness, we implemented our own localization algorithm, which is presented in section IV. The algorithm combines the measurements taken by the IMU, the differential gps, the car sensors and the optical gyroscope.

C. Computational power and software architecture

The system consists of four compact PCI computer racks communicating trough a gigabit Ethernet link. All the racks have the same core architecture which is a Pentium M processor running at 2GHz, equipped with 1.5GB of RAM, Gigabit and Fast Ethernet, two RS232 serial ports, USB ports and a 30GB hard disk. Each rack is dedicated to specific tasks and acquires measurement from different sensors as it is depicted in Fig. 6.

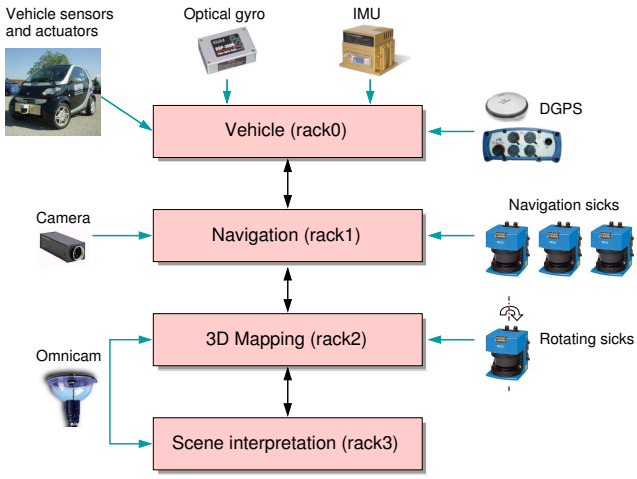


Fig. 6. Architecture of the system

The computer racks run the Linux operating system and the software architecture is based on both GenoM [9] and Carmen [25] robotic software architectures. The functional modules running on different computers exchange data using the Inter Process Communication (IPC) [35]. To guarantee that the time stamp associated to each data packet is globally consistent, the CPU clocks are synchronized using the Network Time Protocol Daemon (NTPD). In order to reduce communication delays, the architecture has been designed in such a way that it minimizes the amount of transmitted data.

The *Vehicle rack* is endowed with a CAN interface that is used to access the vehicle CAN bus. The measurements of the car sensors are continuously read and the car commands such as the vehicle velocity and steering angle are passed to the ECU. The other sensor i.e. the DGPS, the IMU and the optical gyro are connected to the rack through RS232 serial

ports. The main tasks of the Vehicle rack are to keep track of the vehicle position and to control its motion (steering, breaking, velocity control, etc.).

The *Navigation rack* acquires range data from the three navigation scanners through high speed RS422 serial ports. It is endowed with a firewire interface (IEEE 1394) which allows to grab images from the navigation camera. The main task of this computer is to plan a safe path to the goal using the sensor measurements (images and range data) and to provide the motion commands to the Vehicle rack.

A 3D map of the traversed environment is updated on the *3D Mapping rack* using the measurements acquired by the rotating 3D scanner (RS422 board). Like on the Navigation rack, the scanner data is acquired through RS422 ports. The index signal is detected using a multi-purpose IO board and the motor speed is set using an RS232 interface. For more realistic rendering, the texture information acquired by the omnidirectional camera is mapped on the 3D model as it is depicted in Fig. 7

Fig. 7. 3D scan with texture. One recognize a tree on the left hand side, yellow street markings (-x- shape), a pink box (next to the street markings) and persons (on the right, with arms extended)

Finally, the scene analysis is performed on the *Scene interpretation rack*. The artificial objects are extracted from the textured 3D maps and raw omni-cam images and their representation and location are stored in memory as the vehicle moves along the path.

IV. Localization

Our localization algorithm is based on the inverse form of the Kalman filter, i.e., the information filter. This filter has the property of summing information contributions from different sources in the update stage. This characteristic is advantageous when many sensors are involved, which is the case in our system. The localization is done in two steps, namely the state prediction and the state update.

A. State Update

The car state is updated using the measurements taken by several complementary sensors.

- *Differential GPS* We use the WGS-84 standard to convert the GPS coordinates in Cartesian coordinates (x, y, z) expressed in a local navigation frame n. The heading to true north ψ is also output by the unit and is available in the RMC message. The measurement model for the GPS is

$$z_{gps} = \begin{bmatrix} x_{gps} \\ y_{gps} \\ z_{gps} \\ \psi_{gps} \end{bmatrix}_n = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}_n + v_{gps} \tag{1}$$

  In order to reject the erroneous fixes caused by multi-path and satellite constellation change, we use the following gating function [39]

$$z^T(k) \cdot S^{-1} \cdot z(k) \leq \gamma, \tag{2}$$

  where S is the innovation covariance of the observation. The value of γ is set to reject innovations exceeding the 95% threshold.
- *Car sensors* For localization, we use the velocity $\dot{x}_{odo}$ of the car from the CAN bus. Unlike in the case of a flight vehicle, the motion of a wheeled vehicle on the ground is governed by nonholonomic constraints. Under ideal conditions, there is no side slip and no motion normal to the ground surface: the constraints are $\dot{y}_{odo} = 0$ and $\dot{z}_{odo} = 0$. In any practical situation, these constraints are often violated. Thus, as in [6], we use zero mean Gaussian noise to model the extent of constraint violation. The measurement model for the odometry is then expressed as

$$z_{odo} = \begin{bmatrix} \dot{x}_{odo} \\ 0 \\ 0 \end{bmatrix}_b = \left[C_b^n\right]^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_n + v_{odo}, \tag{3}$$

  where C_b^n is the matrix for transforming velocities expressed in the car's frame b into the navigation frame n. The observation noise covariance is obtained using

$$R_{odo} = C_b^n \cdot diag\left\{\sigma_{enc}^2, \sigma_{vy}^2, \sigma_{vz}^2\right\}\left[C_b^n\right]^T, \tag{4}$$

  where σ_{enc}^2 is the variance of the car velocity and σ_{vy}^2,σ_{vz}^2 are the amplitude of the noise related to the constraints.
- *Optical gyroscope* The measurement model for the optical gyro is

$$z_{opt} = \psi_{opt} = \psi + b_{opt} + v_{opt}, \tag{5}$$

  where b_{opt} is the angular offset between the heading to true north ψ and the actual measurement of the gyro.
- *Inertial measurement unit* For the reasons mentioned before, we disabled the GPS and used the unit in angle mode: roll, pitch and heading to magnetic north. The measurement model for this sensor is

$$z_{imu} = \begin{bmatrix} \phi_{imu} \\ \theta_{imu} \end{bmatrix}_n = \begin{bmatrix} \phi \\ \theta \end{bmatrix}_n + v_{imu} \tag{6}$$

$$\psi_{imu} = \psi + b_{imu} + v_{himu}, \tag{7}$$

where b_{imu} is the angular offset between ψ and the heading measurement of the IMU.

B. Prediction model

We apply a standard prediction model for the car which has the following form

$$\mathbf{x}_{k+1} = \begin{bmatrix} F_x & \dots & 0 \\ \vdots & F_y & \\ & & F_z \quad \vdots \\ 0 & & \dots \quad I_{5x5} \end{bmatrix} \cdot \mathbf{x}_k + w_k. \tag{8}$$

The state vector $\mathbf{x}$ contains the position and velocity expressed in the navigation frame n, the orientation of the vehicle represented by the three angles roll ϕ, pitch θ and yaw ψ and the two biases b_{imu} and b_{opt}:

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \phi & \theta & \psi & b_{imu} & b_{opt} \end{bmatrix}^T \tag{9}$$

The position of the vehicle at time $k+1$ is predicted using the position and velocity at time k. This takes the form of a first order process written as

$$F_{x,y,z} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}_k \tag{10}$$

where h denotes the sampling period ($h = 10\,\text{ms}$). All the other elements of the state vector are predicted as simple Gaussian processes. The covariance matrix Q_k associated to the state prediction process is represented as

$$Q_k = G_k \cdot q_k \cdot G_k^T \tag{11}$$

where q_k is a diagonal matrix containing the variances of the elements of the state vector

$$q_x = diag\left\{ \begin{array}{cccccccc} \sigma_x^2 & \sigma_y^2 & \sigma_z^2 & \sigma_\phi^2 & \sigma_\theta^2 & \sigma_\psi^2 & \sigma_{b_{imu}}^2 \sigma_{b_{opt}}^2 \end{array} \right\} \tag{12}$$

Finally, the matrix mapping the noise covariance q_k to the process covariance Q_k is written as

$$G_k = \begin{bmatrix} g_x & \dots & 0 \\ \vdots & g_y & \vdots \\ & g_z & \\ 0 & \dots & diag_{5x5}(h) \end{bmatrix}_k \tag{13}$$

where

$$g_{x,y,z} = \begin{bmatrix} h^2/2 \\ h \end{bmatrix} \tag{14}$$

V. Map Building

A. Map Representation

To represent the 3D data acquired with the rotating laser scanners, we use Multi-Level Surface (MLS) maps [47]. These maps can be regarded as an extension to elvation maps [2, 14, 36, 31, 29, 18]. The idea here is that each cell in a 2D grid can contain many representations of 3D objects called *surface patches*. A surface patch consists of a mean μ and a variance σ, as well as a *depth value* d. Here, μ and σ define a Gaussian distribution that reflects the uncertainty of the measured height of an object's surface. The depth value d represents the length of a vertical interval starting at μ and pointing downwards. The motivation behind this is to represent flat objects, such as street, ground etc., and non-flat objects such as buildings in the same framework. A surface patch of a building will usually have a large depth value, while the depth of a street patch is in general 0. Figure 8 illustrates different possible surface patches in an MLS map.

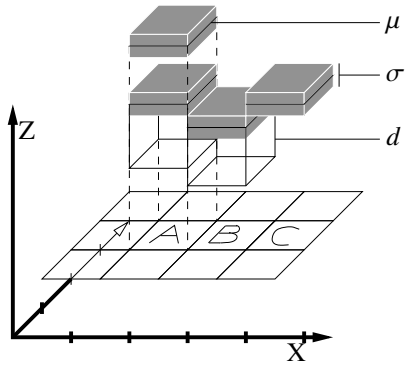


Fig. 8. Example of different cells in an MLS Map. Cells can have many surface patches (cell A), represented by the mean and the variance of the measured height. Each surface patch can have a depth, like the patch in cell B. Flat objects are represented by patches with depth 0, as shown by the patch in cell C.

B. Traversability Analysis and Feature Extraction

One main goal of the MLS map representation is the ability to classify the terrain in which the robot moves. This classification is important to use the map for path planning. Another design goal for the MLS maps was the possiblity to match local MLS maps to one big map, without relying on the raw point cloud data. This matching process is usually performed using the iterative closest point algorithm (ICP). The map matching using ICP has been shown to be very efficient when applying it to subsets of features rather than to the entire data set [28, 33]. Therefore, we first classify the surface patches into the three classes 'traversable', 'non-traversable' and 'vertical'. Then we subsample each of these classes and look for corresponding patches in the other map. This will be described in the next section. For the patch classification, we define vertical patches as those having non-zero depth values. A patch is considered as traversable if it is flat (the depth is 0) and the distance between its height the height of the neighboring patches does not exceed $10cm$. All other flat patches are classified as non-traversable.

C. Map Matching

To calculate the alignments between two local MLS maps calculated from individual scans, we apply the ICP algorithm. The goal of this process is to find a rotation matrix R and a translation vector $\mathbf{t}$ that minimize an appropriate error function. Assuming that the two maps are represented by a set of Gaussians, the algorithm first computes two sets of feature points, $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_{N_2}\}$. In

a second step, the algorithm computes a set of C index pairs or *correspondences* $(i_1, j_1), \ldots, (i_C, j_C)$ such that the point $\mathbf{x}_{i_c}$ in $\mathcal{X}$ corresponds to the point $\mathbf{y}_{j_c}$ in $\mathcal{Y}$ for $c = 1, \ldots, C$. Then, in a third step, the error function e defined by

$$e(R, \mathbf{t}) := \frac{1}{C} \sum_{c=1}^{C} (\mathbf{x}_{i_c} - (R\mathbf{y}_{j_c} + \mathbf{t}))^T \Sigma^{-1} (\mathbf{x}_{i_c} - (R\mathbf{y}_{j_c} + \mathbf{t})), \quad (15)$$

is minimized. Here, Σ denotes the covariance matrix of the Gaussian corresponding to each pair $(\mathbf{x}_i, \mathbf{y}_i)$. In other words, the error function e is defined by the sum of squared Mahalanobis distances between the points $\mathbf{x}_{i_c}$ and the transformed point $\mathbf{y}_{j_c}$. In the following, we denote this Mahalanobis distance as $d(\mathbf{x}_{i_c}, \mathbf{y}_{j_c})$.

In principle, one could define this function to directly operate on the Gaussians when aligning two different MLS maps. However, this would result in a high computational effort, especially if the maps are very big and many Gaussians are stored. Additionally, we need to take care of the problem that the intervals corresponding to vertical structures vary substantially depending on the view-point. Moreover, the same vertical structure may lead to varying heights in the surface map when sensed from different locations. In practical experiments, we observed that this introduces serious errors and often prevents the ICP algorithm from convergence. To overcome this problem, we separate Eq. (15) into three components each minimizing the error over the individual classes of points. These three terms correspond to the three individual classes, namely surface patches corresponding to vertical objects, traversable surface patches, and non-traversable surface patches.

Let us assume that $\mathbf{u}_{i_c}$ and $\mathbf{u}'_{j_c}$ are corresponding points, extracted from vertical objects. The number of points sampled from every interval classified as vertical depends on the height of this structure. In our current implementation, we typically uniformly sample four points per meter. The corresponding points $\mathbf{v}_{i_c}$ and $\mathbf{v}'_{j_c}$ are extracted from traversable surface patches, $\mathbf{w}_{i_c}$ and $\mathbf{w}'_{j_c}$ are extracted from not traversable surfaces. The resulting error function then is

$$e(R, \mathbf{t}) = \underbrace{\sum_{c=1}^{C_1} d_v(\mathbf{u}_{i_c}, \mathbf{u}'_{j_c})}_{\text{vertical cells}} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{v}_{i_c}, \mathbf{v}'_{j_c})}_{\text{traversable}} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{w}_{i_c}, \mathbf{w}'_{j_c})}_{\text{non-traversable}}. \quad (16)$$

In this equation, the distance function d_v calculates the Mahalanobis distance between the lowest points in the particular cells. To increase the efficiency of the matching process, we only consider a subset of these features by sub-sampling.

D. Loop Closing

Usually, the map matching process described in the previous section works well in cases where the robot travels only a short distance. However, for longer distance the accumulated local matching errors may be so large, that the overall map becomes inconsistent. This becomes visible especially when the robot returns to areas where it has been before, which is usually called *loop closing*. In our case, this matching error is bounded due to the high accurate GPS based global localization described before. However, a smaller matching error still remains and is visible in the maps. Therefore, we apply a global pose estimation technique similar to the one presented in [30]. For the details of this technique we refer to [47]. We only note that it is based on a non-linear minimization of the difference between the 3D transformation parameters $(x, y, z, \varphi, \vartheta, \psi)$ resulting from the robot poses and those given by local pose constraints between overlapping local maps. The local pose constraints are obtained by applying ICP matching between overlapping local maps.

VI. Path Planing

In order to acquire data about the environment, the car needs to drive through its environment and visit the different locations. This can be done by manually driving the car or in a more challenging way by autonomous navigation. A realistic approach is to provide a route description to the car and let it run autonomously along that route. However, it is not sufficient for safe navigation to only follow a predefined route since obstacle might block the route and the car has to plan an admissible trajectory around them.

The current version of our planning system follows the ideas of Kelly and Stentz [15] and is closely related to the approach of Thrun *et al.* [45]. The idea is to generate variations of the originally specified route. The robot then evaluates the different trajectories and selects the best admissible one given a cost function. The trajectories are evaluated according to traversability, curvature, and alignment with the specified route. The chosen trajectory is then sent to a low level controller, that keeps the car on the selected trajectory. The controller itself does not change the speed of the vehicle, it only adapts the steering angle of the car. The bigger the error between the car and the trajectory it should follow, the more the car tries to steer towards the given trajectory. We started with a controller that was also applied by Thrun et al. [45] which is given by the control low

$$\alpha(t) \quad = \quad \phi(t) + atan\left(\kappa \cdot \frac{x(t)}{v(t)}\right), \quad (17)$$

where α refers to the new steering command, $\phi(t)$ to the difference of the current steering angle and the orientation of the trajectory the car should follow. $x(t)$ refers to the current distance between the position of the robot and the trajectory and $v(t)$ describes the velocity of the car. The control gain κ influences, how intensive the car steers back to the trajectory. A too high value leads to oscillation and a too small value lead to a comparably slow convergence rate to the reference trajectory. We determined the gain through experiments and obtain good results for $\kappa \in [0.2, 0.4]$.

This controller follows the given trajectory but can lead to small overshoots in curves and to slightly shaky steering commands of the car. To overcome this problem, we do not use the raw velocity information but apply a post filtering in a Kalman filter fashion. This leads to smoother velocity estimates and helps to stabilize the steering command. We furthermore do not compute $\phi(t)$ only based on the closest point on the route but rather averaging of a few poses in front of the car. This compensates for slightly un-smooth input trajectories and reduces the risk of slight overshoots in curves.

VII. Simulation Techniques

The need for a sophisticated simulation environment for our autonomous vehicle mainly originated in two facts. First, the complex software architecture developed by several researchers at different labs had to be tested as a complete system as early as possible to verify the appropriateness of interfaces, interaction protocols, and data rates. Second, as the development of software and hardware was conducted in parallel from the early beginning of the project on, there was an inherent need for physically plausible data sets to test the algorithms, especially in the area of 3d mapping and navigation.

As a consequence, we have built a 3d simulation environment for our autonomous vehicle, its main sensors, and the outdoor terrain. An example for such a simulation is depicted in Figure 9. The developed system is based on the Gazebo simulator [16], which is part of the Player/Stage project. Gazebo uses the Open Dynamics Engine [38] to yield physically plausible simulations in three dimensions by taking into account friction, forces, and rigid body dynamics. It includes a wide variety of pre-build models for robotics applications and is relatively easy to extend. We developed a number of plug-in models for our autonomous smart car and its primary sensors which are three static laser range finders, two rotating laser range finders, an GPS and inertial sensor.

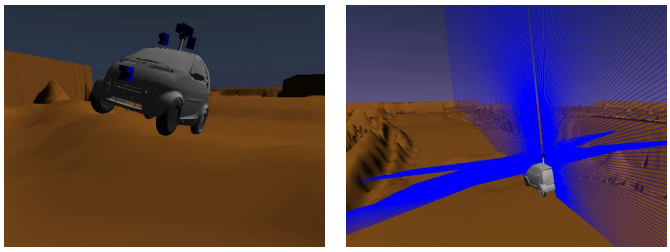


Fig. 9. The simulated autonomous Smart car in outdoor terrain (left). The simulation includes two rotating laser range finders mounted on top of the vehicle (right). The laser beams are visualized in blue.

Instances of these models can be configured and inserted into a simulated world using a simple XML-based description language. At the same time, the plug-in models implement the necessary interfaces and IPC-based communication protocols to our navigation and mapping system so they can readily be exchanged for real hardware components. There are no simulator specific message types build into the overall system to ensure that the architecture is clearly focused on the real application domain.

A. The Simulated Environment

In general, there are several different ways how the simulated environment can be specified. For testing basic navigation capabilities, the simulator supports a flat ground plane and simple geometric objects as obstacles, see the left image of Figure 10. More complex and realistic ground surfaces can be defined in terms of elevation maps or surface maps. By using the interface to the surface map data structure, one can run simulation experiments on terrain that has been traversed with the real vehicle. The right image on Figure 10, for example, depicts the simulated Smart car driving on a surface model that has automatically been constructed from real data acquired at an outdoor test site.

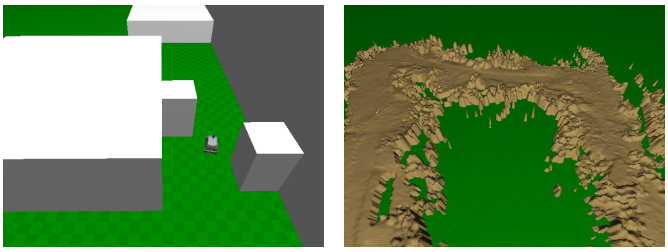


Fig. 10. The simulated environments can be composed of simple geometric objects (left) or can be automatically build from 3d laser range finder data gathered by the real vehicle (right).

B. Experiences with the Simulator

There are often controversial discussions, whether the achievements possible through simulation are worth the efforts necessary for developing and maintaining the simulator itself. From our experience, one should not spend an excessive amount of time on optimizing system parameters in simulation since the real system typically differs substantially in these aspects. On the other hand, the development of the 3d mapping capabilities, the navigation system, and especially the combination of both in one control loop was clearly facilitated by the simulation system.

For the 3d mapping algorithms, the main benefits lay in the possibility to set up simple geometric environments and to compare the mapping results with the known ground truth. Additionally, by simulating a moving vehicle with its physical properties we were able to get an intuition on how accurate the localization system has to be for achieving dense and accurate maps. It was also relatively easy to compare the results for different sensor placements, configurations, and data rates.

For the development of the navigation system, the availability of simple geometric environments was less important as this could be simulated more easily and faster by a simple planar simulation directly build into the navigation module. Far more important was the possibility to test the navigation algorithm in a dynamic setting together with the real local traversability maps calculated online. This could neither have been achieved by replaying real log files nor by using less realistic simulation.

VIII. Experiments

A. Localization

Our approach to localization has been extensively tested and proved to be accurate and reliable in an urban environment. A typical result obtained during the validation phase is depicted in Fig. 11. The figure represents the estimated trajectory of the car overlayed on the ortho-photo of the EPFL campus.

During the experiment, the car drove on areas were GPS was not available or of bad quality (close to buildings, underground, along narrow alleys bordered with trees, etc.).

Fig. 11. Overlay of the estimated trajectory on the ortho-photo of EPFL. The zones where the GPS was not available are highlighted. The total traveled distance is 2350m.

However, the localization algorithm was able to cope with GPS faults and provided accurate positioning estimation, such as depicted in Fig. 12.

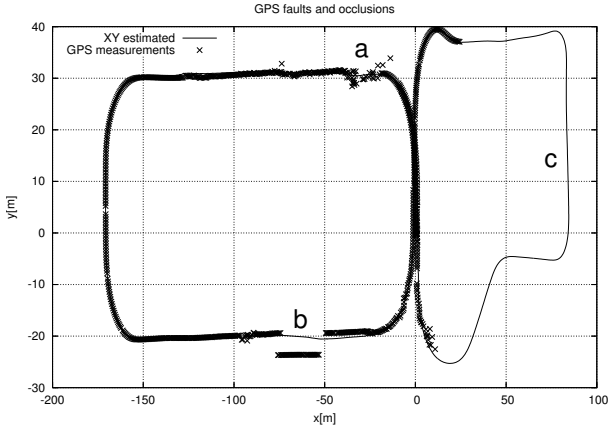


Fig. 12. The graph represents a part of the trajectory depicted in Fig. 11. In this urban environment the GPS signal is disturbed by many objects (trees, buildings, etc.) and GPS faults are of high amplitude. The localization algorithm was able to reject erroneous GPS fixes and to provide accurate estimations. The labels a,b and c mark areas where GPS is of poor quality (a, b) or unavailable (c).

The uncertainty associated to the pose estimation mainly depends on the GPS fixes quality. As depicted in Fig. 13, the standard deviation is low when differential GPS is available ($\sim$ 3 cm) but increases as soon as fixes become unavailable (up to 60 cm).

B. Mapping

To acquire the data, we steered the robotic car over an military test sight. On its path, the robot encountered three nested loops. The goal of these experiments is to demonstrate that our representation yields a significant reduction of the memory requirements compared to a point cloud representation, while still providing sufficient accuracy. Additionally, they show that

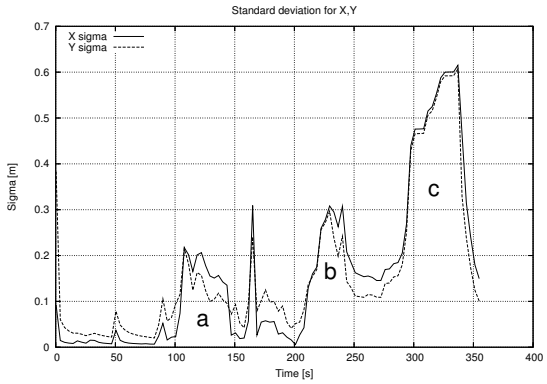


Fig. 13. Standard deviation along the north (x) and west axis (y) for the trajectory depicted in Fig. 12. The standard deviation increases when GPS quality is poor and decreases as soon as it gets better. The labels a,b and c corresponds to the zones marked in Fig. 12

our representation is well-suited for global pose estimation and loop closure. Furthermore we show that our representation is easy to apply to our simulator.

In the experiment we acquired 312 local point clouds consisting of 22,700,000 data points. The area scanned by the robot spans approximately 250 by 200 meters. During the data acquisition, the robot traversed three nested loops with a length of approximately 1,200m. Figure 14 shows a top view of the resulting MLS map with a cell size of 50cm x 50cm and 3 cutouts with a visualized smart. The yellow/light grey surface patches are classified as traversable. It requires 17.15 MBytes to store the computed map, where 36% of 200,300 cells are occupied. Compared to this the storage of the 22,700,000 data points requires 544,8 Mbytes.

IX. Conclusion

In this paper, we presented our approach to mapping of large-scale areas using an autonomous car. We first described out modifies Smart car and setup. Then, we presented out approach to localization which is based on an information filter that merges the information obtained by a variety of different sensors. We furthermore presented our compact map model that is suitable to model outdoor environment in an appropriate way. We showed how to construct a model given a set of smaller map build on the fly. We describe our approach to consistently merge the individual map into a global representation. Our approach has been implemented and tested using a real car equipped with different types of sensors. All experiments presented in this paper, show the result of real world data obtained with this robot.

Acknowledgment

This work has partly been supported by EC under contract number FP6-IST-027140-BACS, FP6-004250-CoSy, and FP6-2005-IST-5-muFly as well as by the German Research Foundation (DFG) within the Research Training Group 1103 and under contract number SFB/TR-8. The authors would like to thank Mike Montemerlo for providing extremely helpful hints in the context of the navigation module.

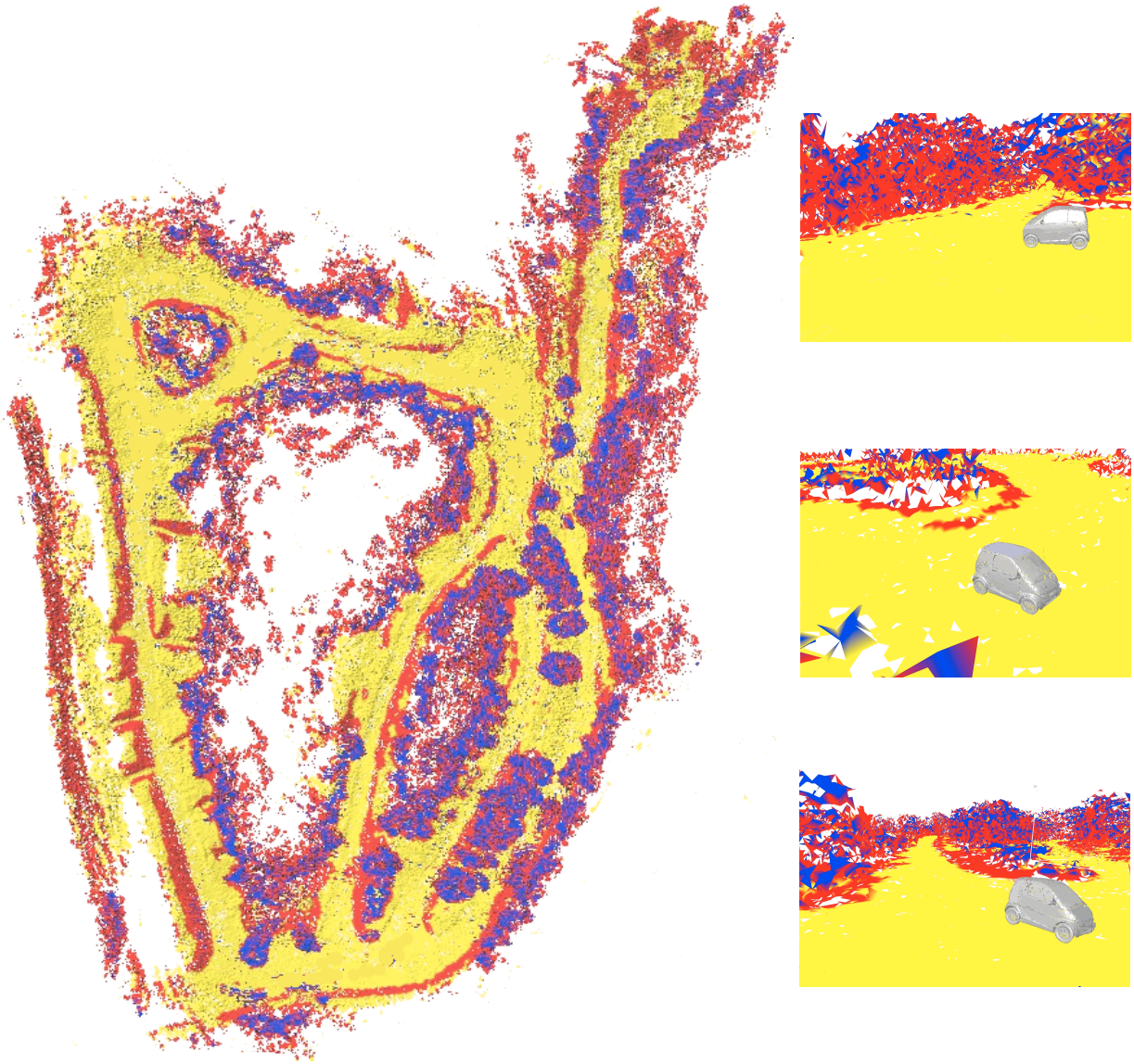


Fig. 14. The left handed image shows a top view of the resulting MLS map of a military test sight with a cell size of 50cm x 50cm. The area scanned by the robot spans approximately 250 by 200 meters. During the data acquisition, the robot traversed three nested loops with a length of approximately 1,200m. On the right hand side three cutouts with a visualized smart are depicted. The yellow/light grey surface patches are classified as traversable.

References

[1] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. Avenue: Automated site modeling in urban environments. In *Proc. of the 3rd Conference on Digital Imaging and Modeling*, pages 357–364, 2001.

[2] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. R. L. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer Society Press*, 22(6):18–22, 1989.

[3] L.B. Cremean, T.B. Foote, J.H. Gillula, G.H. Hines, D. Kogan, K.L. Kriechbaum, J.C. Lamb, J. Leibs, L. Lindzey, C.E. Rasmussen, A.D. Stewart, J.W. Burdick, and R.M. Murray. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*, 2006. Submitted for publication.

[4] DARPA. Darpa gran challenge rulebook. Website, 2004. http://www.darpa.mil/grandchallenge05/Rules 8oct04.pdf.

[5] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[6] G. Dissanayake, S. Sukkarieh, and H. Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, 17(5), 2001.

[7] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[8] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[9] S. Fleury, M. Herrb, and F. Ingrand. GenoM. http://softs.laas.fr/openrobots/tools/genom.php.

[10] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60:5–24, 2004.

[11] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2443–2448, Barcelona, Spain, 2005.

[12] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.

[13] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[14] M. Hebert, C. Caillas, E. Krotkov, I.S. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 997–1002, 1989.

[15] A. Kelly and A. Stentz. Rough terrain autonomous mobility, part 1: A theoretical analysis of requirements. *Journal of Autonomous Robots*, 5:129–161, 1998.

[16] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. Technical report, USC Center for Robotics and Embedded Systems, CRES-04-002, 2004.

[17] P. Kohlhepp, M. Walther, and P. Steinhaus. Schritthaltende 3D-Kartierung und Lokalisierung für mobile inspektionsroboter. In *18. Fachgespräche AMS*, 2003. In German.

[18] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury and; M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *Int. Journal of Robotics Research*, 21(10-11):917–942, 2002.

[19] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4), 1991.

[20] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proc. SIGGRAPH*, pages 131–144, 2000.

[21] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[22] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

[23] C. Martin and S. Thrun. Online acquisition of compact volumetric maps with mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, 2002. ICRA.

[24] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

[25] M. Montemerlo, N. Roy, S. Thrun, D. Hähnel, C. Stachniss, and J. Glover. CARMEN – the carnegie mellon robot navigation toolkit. http://carmen.sourceforge.net, 2002.

[26] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[27] H.P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, Robotics Institute, 1996.

[28] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[29] C.F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, 2000.

[30] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor estimates. In *ICRA*, 2006.

[31] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot. 3-d modelling and robot localization from visual and range data in natural scenes. In *1st International Conference on Computer Vision Systems (ICVS)*, number 1542 in LNCS, pages 450–468, 1999.

[32] K. Pervölz, A. Nüchter, H. Surmann, and J. Hertzberg. Automatic reconstruction of colored 3d models. In *Proc. Robotik*, 2004.

[33] P. Pfaff and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping. In *In Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, pages 165–176, 2005.

[34] Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley Publishing Inc., 1989.

[35] R. Simmons. IPC – inter process communication. http://www.cs.cmu.edu/afs/cs/project/TCA/www/ipc/index.html.

[36] S. Singh and A. Kelly. Robot planning in the space of feasible actions: Two examples. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1996.

[37] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[38] Russell Smith. Open dynamics engine. Website, 2002. `http://www.q12.org/ode/ode.html`.

[39] S. Sukkarieh, E.M. Nebot, and H. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle application. *IEEE Transactions on Robotics and Automation*, 15(3), 1999.

[40] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[41] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, 2000. IEEE.

[42] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[43] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7-8):693–704, 2004.

[44] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery Montemerlo, C. Deepayan, and W. Burgard. A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–442, 2003.

[45] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. To appear.

[46] R. Triebel, F. Dellaert, and W. Burgard. Using hierarchical EM to extract planes from 3d range scans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[47] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[48] C. Urmson. *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2005.

[49] Carl Wellington, Aaron Courville, and Anthony Stentz. Interacting markov random fields for simultaneous terrain modeling and obstacle detection. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[50] O. Wulf, K-A. Arras, H.I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *ICRA-04*, pages 4204–4209, New Orleans, apr 2004. IEEE.

[51] C. Ye and J. Borenstein. A new terrain mapping method for mobile robot obstacle negotiation. In *Proc. of the UGV Technology Conference at the 2002 SPIE AeroSense Symposium*, 1994.

[W5] S. Kolski, D. Furgeson, C. Stachniss, and R. Siegwart. Autonomous driving in dynamic environments. In *Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

Autonomous Driving in Dynamic Environments

Sascha Kolski*, Dave Ferguson†, Cyrill Stachniss*, and Roland Siegwart*
*Swiss Federal Institute of Technology (ETHZ)
Zurich, Switzerland
{kolskis, scyrill, rsiegwart}@ethz.ch
†Carnegie Mellon University (CMU)
Pittsburgh, PA, USA
dif@cmu.edu

***Abstract*— Autonomous vehicles are being used increasingly often for a range of tasks, including automated highway driving and automated parking. These systems are typically either specialized for structured environments and depend entirely on such structure being present in their surroundings, or are specialized for unstructured environments and ignore any structure that may exist. In this paper, we present a hybrid autonomous system that recognizes and exploits structure in the environment in the form of driving lanes, yet also navigates successfully when no such information is present. We believe this approach is more flexible and more robust than either of its sub-components alone. We demonstrate the effectiveness of our system on both marked roads and unmarked lots under the presence of dynamic objects, such as pedestrians or other vehicles.**

I. Introduction

Every year, thousands of people are killed in road accidents, with millions more injured. The vast majority of these accidents are due to human error, with roughly 5% caused by vehicle defects [1]. Such staggering findings motivate the use of driver assistant systems and fully automated vehicles to increase driver and passenger safety.

Driver assistant systems can help drivers to identify dangerous vehicle states and traffic scenarios and reduce the risk of accidents. These driver assistant systems are widespread in all categories of vehicles and range from anti-lock brakes to radar based adaptive cruise control. The development of these systems has been accelerated by integrated drive-by-wire components such as electronic gas pedals, brakes, and steering systems.

The development of such components has also hastened the arrival of autonomous passenger vehicles. In 1997, the NavLab vehicles travelled 'no hands' across the United States, requiring only accelerator and brake pedal interaction from the driver [2]. In 2005, 23 autonomous vehicles started a race across the Nevada desert in the DARPA Grand Challenge race [3], with 5 of them finishing the 211.1 Km distance.

Most of these systems depend on environmental structure like driving lanes or dense sets of GPS points. However, in many common driving scenarios neither of these sources of information will be available, for example, when leaving a road and entering a parking lot.

Autonomous navigation in unstructured environments is an active research area in field robotics, and a number of effective approaches have been developed that address this task [4]–[7]. A common technique is to maintain a map of the environment and use this to plan safe paths to a desired goal location. As the vehicle traverses the environment, it updates its map and path based on its observations. Such an approach works well when dealing with reasonably small areas, but storing and planning over maps of the entire environment is impractical when traversing long distances. Further, without taking into account non-spatial information such as road markings, these approaches are unable to ensure that the vehicle stays within its lane (or even on the road) when navigating through highway or urban environments.

In this paper we present a hybrid navigation system that combines the benefits of existing approaches for driving in structured environments (e.g. roads) and unstructured environments (e.g. parking lots). When driving on detectable roads, the system uses visual lane detection and laser range data to generate a local map, which is processed by a local planner to guide the vehicle down the lane while avoiding obstacles. When driving in unstructured environments, the system employs a global map and planner to generate an efficient trajectory to a desired goal. The combined system is capable of navigating a passenger car to a given goal position without relying on road structures, yet it makes use of such structure when it is available. We also describe extensions to this approach capable of dealing with dynamic obstacles, such as pedestrians or other vehicles, that are commonly found in realistic driving scenarios.

We begin by briefly introducing our autonomous Smart vehicle and its onboard sensors. We then describe our system for navigating structured and unstructured environments, and go on to describe how this system can be used in environments containing dynamic obstacles. In Section IX we present results from runs performed in road and parking lot scenarios and we conclude with discussion.

II. Vehicle and Sensors

Our vehicle is a Smart fortwo passenger car that has been modified for autonomous operation. Firstly, we have interfaced the Smart's controller area network (CAN) bus to access data on the dynamic state of the vehicle, specifically the wheel speed and the steering angle. We have also added actuators to the brake pedal and interfaced the electronic gas pedal and power steering. Finally, a number of sensors (discussed below) have been added to provide vehicle and environmental information. A detailed description of the mechanical and architectural aspects of the vehicle can be found in [8].

Fig. 1. Our autonomous Smart car platform. There are three fixed laser range finders mounted on the front of the vehicle and on the sides of the roof, and two spinning laser range finders mounted together on the center of the roof. Inside the vehicle, mounted behind the windscreen, is an automotive camera used for lane detection.

A. Proprioceptive Sensors

As with many other passenger cars, the Smart is equipped with a variety of sensors which are linked using the vehicle's CAN bus. By interfacing this bus it is possible to access the sensor data and measure the vehicle's dynamic state precisely.

a) Wheel Encoders: The overall vehicle speed is derived from the four wheel encoders with a resolution of 0.5 revolutions/minute. The steering wheel angle is available with a resolution of $0.04°$.

b) IMU: We have added a 6 degree of freedom IMU to the Smart that is able to measure angular rates up to 100°/sec at a resolution of 0.025°. Lateral accelerations in all three dimensions can be measured up to 2g with a resolution of 0.01 m/s^2.

B. Exteroceptive Sensors

c) Differential GPS system: (Omnistar Furgo 8300HP, rain proof antenna) This system provides an accurate position estimate together with its standard deviation when satellites providing the GPS drift correction are visible from the car. When no correction is available standard GPS is provided.

d) Laser Range Finders: We use five SICK laser range finders for sensing the spatial structure of the environment. These are configurable laser range finders based on time of flight measurements, with angular resolutions of 1 or 0.5°, angular ranges of 180° and measuring ranges up to 80 meters. Three of these lasers are kept at fixed angles—one at the front of the vehicle and two on the sides of the roof—to quickly detect upcoming obstacles and difficult terrain, and two of the lasers are mounted to a spinning platform—on the center of the roof—to provide full 3D information about the vicinity of the vehicle.

e) Monocular Camera: An automotive gray-scale camera is mounted inside the vehicle at the top of the windscreen for observing the area in front of the vehicle and detecting lane information. The resolution of the camera is 750 $\times$ 400 pixels and it delivers information at 25 frames per second.

III. Position Estimation

The localization algorithm used in our system is based on the information form of the Kalman filter, the Information filter. This filter has the property of summing information contributions from different sources in the update stage. This characteristic is very interesting when many sensors are involved, which is the case in our system. Our sensor fusion scheme is based on [9] [10] and [11]. To accurately localize the vehicle, four different sensors are used: DGPS, IMU, optical gyro and vehicle sensors (wheel encoders and steering angle sensor). The combination of their measurements allows the estimation of the vehicle's 6 degrees of freedom i.e. the 3D position (x, y, z) and the attitude (roll, pitch, heading).

A detailed description of this aproach can be found in [8].

IV. Traversability Estimation

Reliable estimates of the traversable area in the vicinity of the vehicle are crucial for autonomous driving. We currently use the three static laser range finders on our vehicle to estimate the traversability of the area in front of the vehicle.

Given a laser range observation, we first compute the end points of the individual beams. We then add the 3D points to the cells of a local two-dimensional grid map according to the x, y-coordinate of the beam (horizontal position). We then parse the cells and compute the mean and variance of the z-values (vertical position) for each cell. The traversability classification of a cell is performed locally based on these values. When adding observations from multiple laser range finders into a single grid, it is often the case that false obstacles are detected by some of the lasers (described as phantom obstacles by Thrun et al. [12]). These false obstacles are caused by small errors in the pitch estimate of the pose of the vehicle. To remove these artifacts, we compute the traversability estimate individually for each scan and merge the independently-estimated traversability values into a common grid structure. We found that this yields good results when moving on streets as well as on unpaved roads. An example traversability map produced by this approach is shown in Figure 2.

V. Traversability Prediction

If an autonomous vehicle has onboard far-range sensors, then the information from these sensors can be used by the vehicle to enable smooth following of winding roads and early reaction to obstacles in the vehicle's route. The smaller the sensing range, the more extreme the obstacle avoidance movements of the vehicle must be. This is also true for following the current road lane. To generate smoother motion

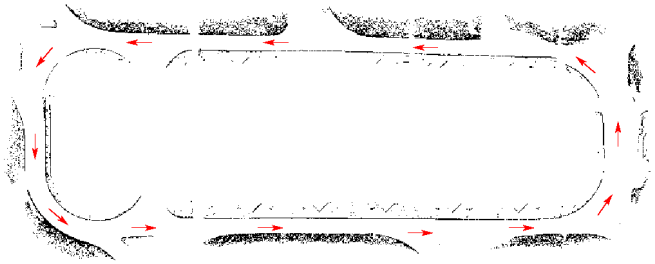


Fig. 2. A traversability map obtained using the three fixed laser range finders on our vehicle. Black areas are untraversable and the red/grey arrows illustrate the trajectory taken by the car.

of the vehicle using only our laser range finders, we perform a prediction of the traversable area at a far range by using the accurate near-range information provided by these sensors.

In order to perform the prediction, we compute a 1D pattern of average cost values. This pattern is generated by storing, for each cell in our 1D pattern, the average cost of all the already-observed cells in our local cost map that have their lateral offset position from the current route match the cell position in our 1D template. This process is illustrated by the image in the second row of Figure 3.

We then use this pattern to estimate unknown traversability cost values for cells far away from the vehicle along the current route (illustrated by the blue lines in the same figure). In this way, we obtain an estimate of the expected cost in the currently unobserved area. The image in the last row of Figure 3 provides an example result of this estimation. The area within the red rectangle is the measured cost map and the the area in the blue rectangle is the predicted cost based on the measured cost map.

This technique allows us to estimate the traversability of areas that have not been observed with the sensors. This prediction does not cope with unforseen obstacles but it does help the car to improve its estimate of the road projection. This is especially important if the localization is affected by GPS drift, since in this case it is not sufficient to simply follow the predefined route. With our approach we are able to robustly estimate the road and the traversable area, even when faced with GPS drift or outages.

VI. Driving in Structured Environments

When driving in structured environments such as roads or highways, it is important for safety that vehicles abide by traffic rules and stay in their own lanes. For autonomous vehicles, such structure is useful because it constrains the available actions of the vehicle and reduces the complexity of the navigation task. For instance, if an autonomous vehicle is traveling down a road, it knows it must stay within its current lane so the lane can be used as a guide for where the vehicle must travel to next. Such an approach can be coupled with a standard commercial navigation unit that provides higher-level guidance on when to turn down which street.

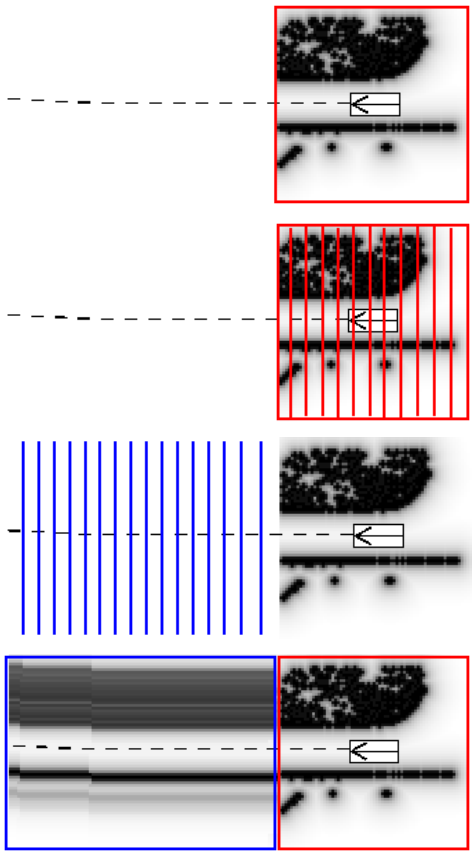


Fig. 3. The traversability prediction. The image in the first row shows a local cost map representing the traversable and non-traversable area. The dashed line illustrates the route. Based on this map and the route description, we compute a 1d cost pattern based on the cell labeled by the red lines in the image in the second row. We then use this pattern to predict the traversability in from of the car (illustrated by the blue lines in the third figure). Finally, we obtain a cost prediction for cells not observed to far by the robot (labeled by the blue rectangle in the last image).

The resulting combined system can autonomously navigate between arbitrary road locations.

However, to ensure safe navigation, it is not enough to just follow the current lane. The vehicle must be alert at all times and able to avoid other cars and objects that may unexpectedly place themselves in its path, such as cars pulling out from driveways or pedestrians crossing the street, for example. To achieve such behavior in our Smart, we construct a local map using the traversability estimation method described in the previous section and plan a collision-free path through this map. Both the map and the plan are updated frequently (at 20 and 10 Hz, respectively). With both the local obstacles and lane information encoded in the local map, the vehicle is able to plan trajectories that keep it within the current lane and also avoid any obstacles.

A. Lane Detection

To extract lane information, we use a monocular gray-scale camera designed for automotive use and a real-time lane detection algorithm running on a separate computer equipped

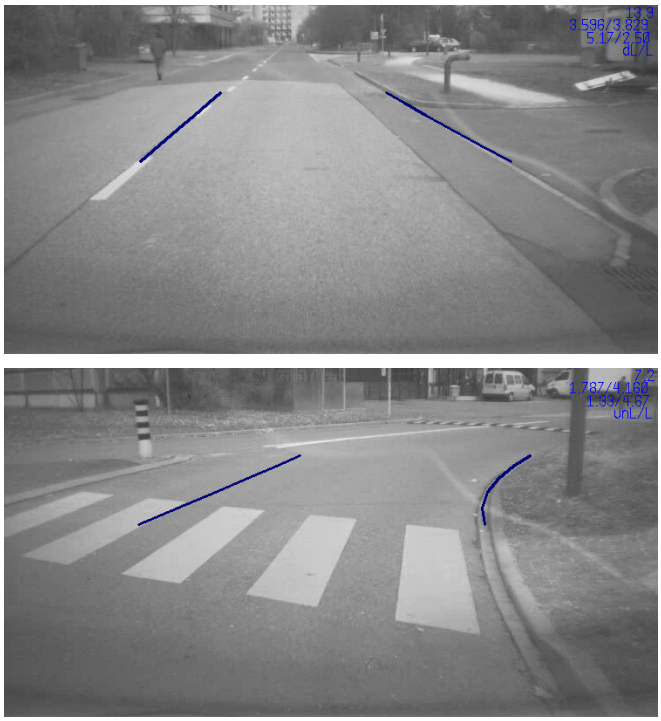


Fig. 4. Example results from our lane detection approach applied to images from a straight (top) and curved (bottom) section of road.

with a frame grabber. Our approach combines hypotheses from several lane detection algorithms, each designed to detect different types of lanes, such as the closest lane to the vehicle, straight lanes, or curved or symmetric lanes. These algorithms rely mainly on the spatial gradient of the image to extract their hypotheses. The results of the individual algorithms are then combined to determine the most probable lane. Example results from our lane detection algorithm are shown in Figure 4 and more details on the algorithm can be found in [13].

B. Local Planning

In order to follow the current lane safely and smoothly, we project a set of potential vehicle actions onto our traversability map and check the cost of these actions. For this, we use an approach similar to that used by the Stanford Racing Team in the Grand Challenge [12]. We take the centerline of the lane and use this to construct a set of possible trajectories for the vehicle to execute. These trajectories vary in their lateral offset from the nominal centerline path and provide a series of alternatives from which the best obstacle-free trajectory can be selected.

By exploiting the structure of the driving lane, this combined approach provides smooth, safe trajectories for the vehicle when it is operating on roads.

VII. Driving in Unstructured Environments

In unstructured environments where there is no lane information to guide or constrain the actions of the vehicle, we must use a more general approach for navigation. For instance, imagine our vehicle has arrived at its intended destination address and now wants to park in a specified goal location within the parking lot. To do this, we can still use the local planning component of our system, however we now need to compute a path for the planner to follow as we no longer have lane information to provide this for the vehicle. To generate these paths we use the Field D* algorithm, which has been incorporated into several fielded robotic systems [14]. This algorithm provides very low-cost 2D paths through grid-based representations of an environment and is able to repair these paths to account for new information as the vehicle observes obstacles during its traverse. These paths do not take into account the heading restrictions of the vehicle and instead approximate the least-cost path to the goal for a vehicle that can turn in place. Because Field D* does not encode the mobility constraints of the vehicle, it cannot be used alone for accurate trajectory planning for the vehicle. Consequently, we combine it with a local planner to provide feasible paths.

One way to do this is to use the Field D* path as the input to our local planner, which will then track this path to the goal. As the vehicle navigates through the environment, the global Field D* path is updated based on new information received through the onboard sensors, and the trajectories generated by the local planner are subsequently updated to reflect the new global path. This approach works well in static environments, where the Field D* path can be quite accurately tracked using the local planner. However, in dynamic environments such an approach may not be ideal, as discussed below.

VIII. Navigating in Dynamic Environments

Typical driving scenarios involve dynamic obstacles: there are usually pedestrians or other vehicles moving around within the environment that need to be avoided. These dynamic obstacles need to be accurately detected and reasoned about in order to produce safe paths for our vehicle to traverse. In the following two subsections we describe extensions to our navigation approach that enable us to model and reason about dynamic elements.

A. Mapping Dynamic Environments

To detect and predict the trajectories of moving objects, several approaches have been proposed in the robotics community. Feature-based approaches operate extract features from the raw data and then track these features to compute their motion parameters. Such approaches are suitable for a variety of sensor data, for example, vision, radar, and laser, and have been widely used [15]. However, these approaches typically require a priori knowledge of the features to track and are therefore only suitable for the detection of well defined classes of objects. Raw data-based approaches, on the other hand, detect motion from raw sensor data and do not depend on any model of the objects being observed. They are thus less accurate for predicting well-behaved, known object classes but perform well when confronted with a range of different dynamic elements.

Our vehicle uses a raw data-based scan alignment approach to detect moving objects in the environment. Based on work

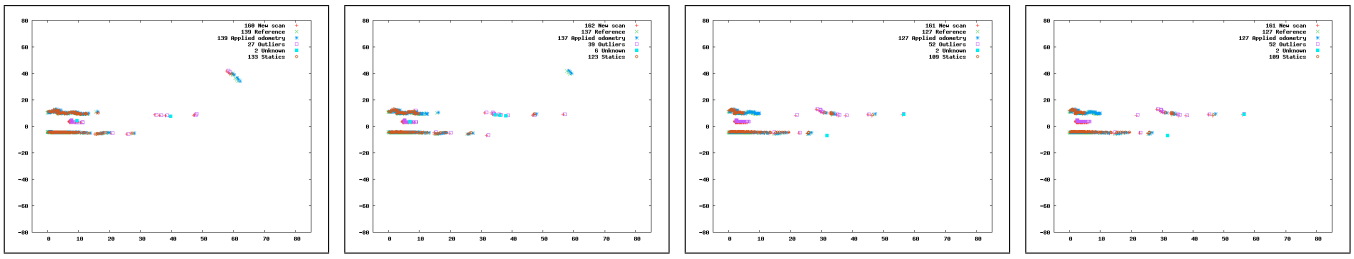


Fig. 5. Scan points taken during a test ride on campus. The static structure of the environment (curbs, buildings) is detected as static parts of the environment while the oncomming vehicle forms an L-shaped set of dynamic points.

introduced by Jensen [16], our algorithm extends the *iterative closest points* algorithm (ICP) [17]. The ICP algorithm aligns two sets of points by iteratively finding the set of points in one scan that are closest to a set of points in the other scan, and then computing a transformation that minimizes the distance between the two sets of points. Special care has to be taken to suppress outliers, which are points that are present in one scan, but not in the other, because they bias the alignment. The pose correction $d_x, d_y, d\Theta$ is computed as a weighted mean over all connected points. The link between a scan point (x_i, y_i) and a scan point (x_j, y_j) is expressed by a link variable $l_i = j$. With this the pose correction can be computed from the linked scan points and results in

$$d_x = \frac{1}{I} \sum_{i=I^*} (x_i - x_{l_i}) \tag{1}$$

$$d_y = \frac{1}{I} \sum_{i=I^*} (y_i - y_{l_i}) \tag{2}$$

$$d\Theta = \frac{1}{I} \sum_{i=I^*} (\phi_i - \phi_{l_i}) \tag{3}$$

Linking and correcting are repeated until the correction value is below a predefined threshold. The resulting transformation determines the displacement from the reference to the correspondence pose.

While in scan matching outliers are a disturbing factor and are filtered out in each iteration, they are very useful for the detection of dynamic obstacles. In our approach, the outliers found in each iteration of scan matching are collected and clustered. The resulting clusters are candidates for dynamic objects and are tracked to derive their motion parameters. Figure 5 provides an example illustrating this ability of this approach to filter static points from those in motion.

B. Planning in Dynamic Environments

When driving within road lanes, dynamic obstacles usually do not significantly interfere with the traverse of our vehicle because their behavior is well-defined. To ensure we don't collide with any of these obstacles, our local planner can estimate the trajectories of these other vehicles or pedestrians and then check that its intended trajectory does not intersect these objects at any point in time. Figure 6 shows a simple example of this reasoning. The local planner can then remove from contention any trajectories that intersect dynamic obstacles (or modify the velocity profile of the trajectory to avoid the obstacle). Since the local planner is generating a series of possible trajectories that span the current driving lane, at least one of these trajectories should still be obstacle-free if the dynamic obstacle is abiding by traffic rules.

For our unstructured driving scenario, the situation is complicated because the dynamic obstacles may interfere entirely with the global path being tracked. Thus, it may not be possible to track this path using our local planner. Instead, we may need to evaluate a more general set of possible local trajectories for the vehicle to execute, including some that do not follow the current path. For this, we use an approach that follows a large body of work on outdoor mobile robot navigation [4], which has the local planner project out a range of possible local trajectories and then evaluate each trajectory based on both the cost of the trajectory itself (in terms of curvature, terrain, distance, etc), as well as the cost of a global path from the endpoint of the local trajectory to the goal. Thus, rather than a single global path being planned from the current vehicle position to the goal, global paths are planned from each trajectory endpoint. Since Field D*, like most replanning algorithms, performs planning in a backwards direction out from the goal, computing these extra paths and their associated costs is very efficient (and often requires no extra planning at all).

Figure 7 shows an illustrative example of this combined approach. Here, a set of local arc-based trajectories are shown in red/gray, with the best trajectory shown in blue/black. Here, the best trajectory was selected based on a combination of the cost of the trajectory itself and the cost of a global path from the end of the trajectory to the goal (the goal is shown as a filled circle at the right of the figure). The global path from the end of the best trajectory to the goal is also shown in blue/black. In this example, a purely local planner would have selected the straight trajectory leading directly to the right, as this brings it closest to the goal in terms of straight-line distance. However, such a trajectory could cause it to get stuck behind the clump of obstacles in the middle of the map.

IX. Experiments

We have tested our system in both structured and unstructured environments. For structured environments, we had the vehicle drive down a road and record the resulting local maps.

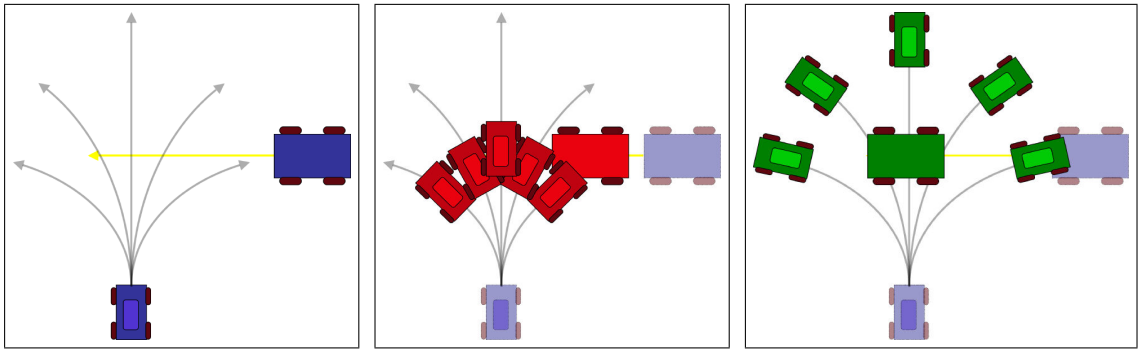


Fig. 6. Local planning amidst dynamic obstacles. If an agent (facing upwards) assumes the dynamic obstacle (traveling in from the right) is static when choosing its next action (potential actions shown as the arcs emanating out from the agent), it may select an action that will have it collide with the obstacle at some future point in time. Instead, it needs to estimate the position of the dynamic obstacle in the future and use these estimates to select an action that will avoid the obstacle at all times. The three images show the potential position of the agent based on its available actions, as well as the position of the dynamic obstacle, at three stages in time (the color of each agent reflects the time).

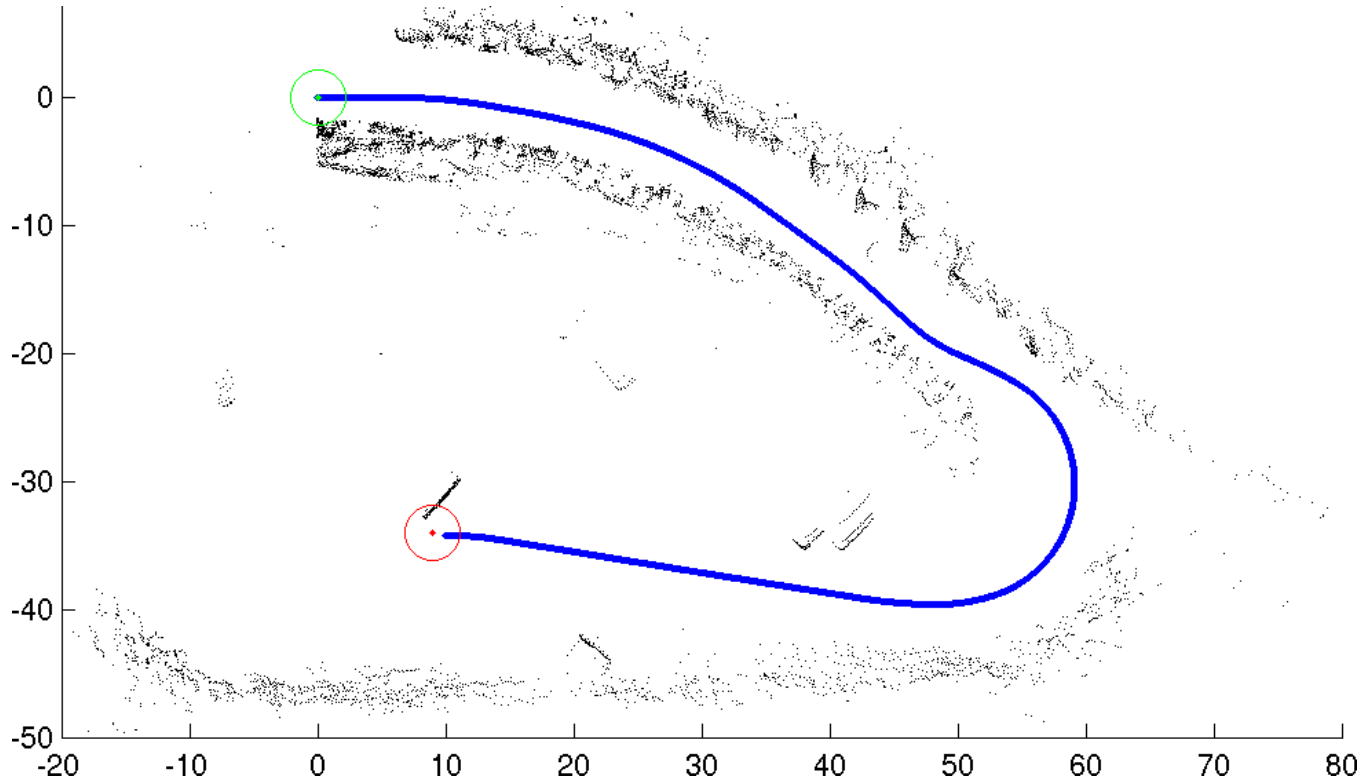


Fig. 8. Results from global planning and mapping in an unstructured environment. Shown here is the map created from the laser during an autonomous traverse from an initial position on a rural road to a goal position inside a large parking lot. Also shown is the path (in blue/black) traversed by the vehicle. The vehicle began from the position marked in green/gray at the top of the map, and navigated to the goal position marked in red/gray at the bottom.

Fig. 9. Snapshots from a video taken of the traverse in Figure 8.

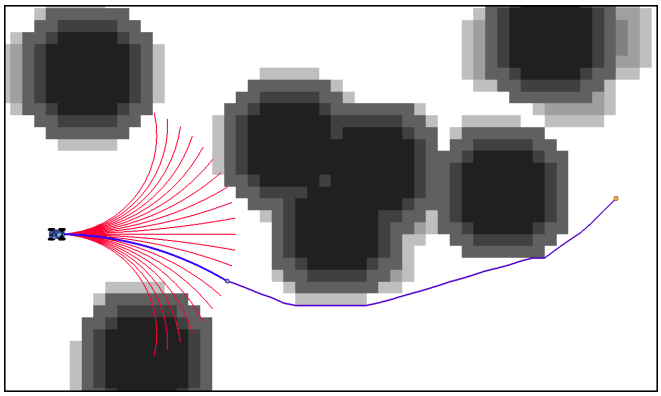


Fig. 7. **Global Planning in Unstructured Environments**. The vehicle projects a set of feasible local trajectories through the local map from its current position and orientation (trajectories for a single speed are shown in red/gray). The cost of each of these trajectories is computed based on the cost of the cells the trajectory travels through (darker areas are more expensive, with black cells representing obstacles). A global path is planned from the end of each trajectory to the goal (shown as a filled circle on the right side of the map) and the cost of this path is added to the cost of the trajectory. The best trajecotry is shown in blue/black, along with the global path from the end of this trajectory to the goal. The map here has been configuration-space expanded so that the vehicle can be treated as a single point during planning.

Figure 10 shows the combined cost map constructed from the series of local maps and highlights both obstacles and lane information. Since the laser range data does not contain any information about the lane markings, the vision-based lane detection system is necessary to keep the vehicle in its lane.

To test our vehicle in unstructured environments, we gave it a more complex task. We began on a road and tasked it with autonomously navigating to a goal location in a nearby parking lot. Because there were large shrubs between its initial position and its goal, it was forced to travel down the road until it observed an opening through which it could enter the parking lot. At this point it entered the parking lot and navigated to its goal location.

Figure 8 shows the resulting map built by the vehicle and the vehicle's traverse. Figure 9 shows a series of images taken from a video of the traverse. Overall the vehicle travelled about 140 meters in 62 seconds, i.e. at average speed of roughly 2.3 m/s.

The vehicle trajectory seen in Figure 8 shows its ability to navigate in a scenario given a sparse set of waypoints and a combination of global and local path planning techniques.

Together these experiments illustrate our vehicle's ability to navigate through both road and non-road environments. Our vehicle effectively avoids obstacles to reach a defined goal position without relying on an a priori model of the environment.

X. Conclusion

In this paper we have presented a hybrid approach for autonomous navigation in structured and unstructured environments. Our approach exploits any lane structure present in the environment and combines this with local obstacle information to guide the vehicle along safe trajectories. When

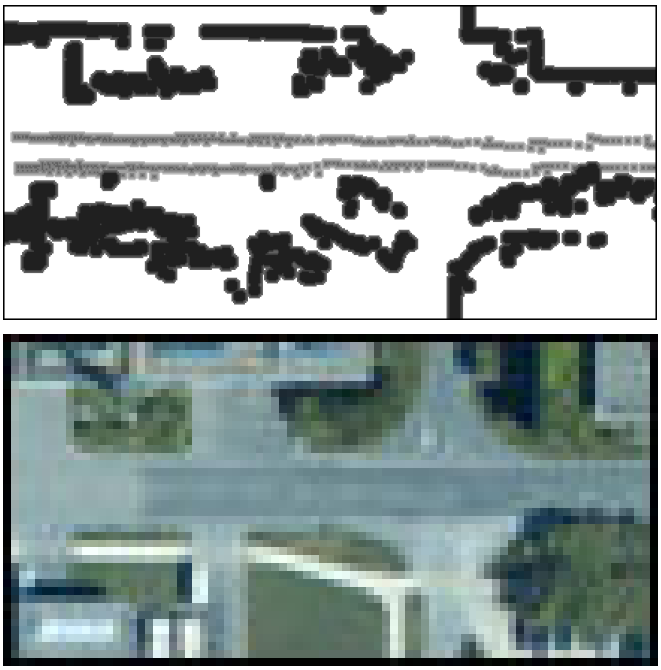


Fig. 10. Results from our lane detection and mapping in a structured environment. Data was gathered from roughly 100 meters of traverse down a road (traveling from left to right). The top image shows the combined local maps created by the vehicle during the traverse, with lane information shown as dark gray areas and obstacles shown in black. Notice that the obstacle information does not provide any real indication of the location of the lane or even road, and so does not suffice for safely guiding the vehicle. The bottom image shows a satellite map of the area.

no structure is detected, the approach falls back on a global planner that generates efficient paths for the vehicle to desired goal locations. We have provided results demonstrating the operation of the vehicle in both structured and unstructured environments.

Acknowledgements

We thank the European Commission for partially funding this work through the SPARC project (Secure Propulsion using Advanced Redundant Control) under IST-507859 and the BACS Project under contract number FP6-IST-027140-BACS. Dave Ferguson is partially supported by a U.S. National Science Foundation Graduate Research Fellowship.

References

[1] M. Shell, "Final report of the european esafety working group on road safety, online available," 2003. [Online]. Available: http://europa.eu.int/informationsociety/activities/esafety/indexen.htm

[2] C. Thorpe, T. Jochem, and D. Pomerleau, "The 1997 automated highway demonstration," in *1997 International Symposium on Robotics Research*, 1997.

[3] "Darpa grand challenge race website." [Online]. Available: http://www.darpa.mil/grandchallenge

[4] A. Kelly, "An intelligent predictive control approach to the high speed cross country autonomous navigation problem," Ph.D. dissertation, Carnegie Mellon University, 1995.

[5] A. Stentz and M. Hebert, "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol. 2, no. 2, pp. 127–145, 1995.

[6] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[7] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, "Recent progress in local and global traversability for planetary rovers," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[8] P. Lamon, S. Kolski, and R. Siegwart, "The smartter - a vehicle for fully autonomous navigation and mapping in outdoor environments," In Proceedings of CLAWAR 2006, Brussels, Belgium, 2006.

[9] P. Lamon and R. Siegwart, "3d position tracking in challenging terrain," Proceedings of the Field and Service Robotics FSR 2005, August 2005, 2005.

[10] E. N. G. Dissanayake, S. Sukkarieh and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications," *IEEE Transactions on Robotics and Automation, 2001*, 2001.

[11] E. N. S. Sukkarieh and H. Durrant-Whyte, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics and Automation, Jun 1999*, 1999.

[12] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge," *Journal of Field Robotics*, 2006, accepted for publication.

[13] M. Bellino, Y. Lopez de Meneses, P. Ryser, and J. Jacot, "Lane detection algorithm for an onboard camera," *SPIE proceedings of the first Workshop on Photonics in the Automobile*, 2004.

[14] D. Ferguson and A. Stentz, "Field D*: An Interpolation-based Path Planner and Replanner," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2005.

[15] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *IEEE International Conference on Robotics and Automation*, May 2003.

[16] B. Jensen, R. Philippsen, and R. Siegwart, "Motion detection and path planning in dynamic environments," in *Workshop Proc. Reasoning with Uncertainty in Robotics, International Joint Conference on Artificial Intelligence IJCAI'03*, 2003.

[17] P. J. Besl and N. D. Kay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.