Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation

Dieter Fox
Institute of Computer Science III
University of Bonn, Germany

Doctoral Thesis
December 1998

To my parents

Acknowledgements

The work described herein could not have been done without the support of numerous people. I would like to begin by expressing my appreciation to my advisor Armin B. Cremers and co-advisor Leslie P. Kaelbling. I am grateful for their suggestions and guidance.

This work owes much of its inspiration to Wolfram Burgard. In many hours of fruitful discussions he was always willing to listen and gave valuable suggestions. Sebastian Thrun had an enormous impact on my thinking. I especially appreciate his enthusiasm and analytical ability. As friendly critics, Wolfram's and Sebastian's comments helped me a lot in finding a scientific approach to the problems I addressed.

I would also like to thank all other members of the "Rhino Team", and especially Michael Beetz, Dirk Hähnel, Daniel Hennig, Dirk Schulz, and Walter Steiner, for their spirit of cooperation. They provided the framework necessary to develop and evaluate my ideas.

I am grateful to Joachim Buhmann and his group who have always been open for questions concerning the statistical background of this work.

Finally, I wish to thank my family and friends who made it possible for me to go through the tough times and who helped me enjoying the good times.

Contents

Chapter 1

Introduction

This dissertation explores fundamental issues in effective and safe navigation of mobile indoor robots. In particular, we will look at the problem of self-localization and aspects of collision avoidance during the process of self-localization. A successful realization of this process has to address several sub-problems.

Consider, for example, a man with a city map trying to find a particular place in an unknown city. In order to use this city map, the man must first determine his position within the city. To do so, he can look around and compare his observations with the information represented in the map. Here, such an observation can be for example a road sign or a landmark. In most cases, a single observation is not enough to uniquely determine one's location. This is mainly due to the fact that several places often look the same. Thus, the problem of *ambiguities* is an inherent part of the problem of localization. In order to disambiguate between the different possible positions, one has to keep these possible locations in mind and collect additional information at other locations in the environment. In many cases it is sufficient to undirectedly move around until enough observations have been made. This could be the case e.g. if the next street sign is found. However, if the environment possesses only relatively few features that enable one to unambiguously determine one's location, it can be much more efficient to first determine directions in which to move in order to *actively disambiguate* between the possible locations. For example, the man in the foreign city could look for a church steeple in order to uniquely determine his location. Additionally, the person often is confronted with the following problem: Due to changes at some places, the city looks different from the representation of the map. We call this the problem of *dynamic* environments. Nevertheless, once the person knows where he is, he still must keep track of his position as he moves on. In most cases, the task of *tracking* can be efficiently achieved e.g. by continuously comparing the observations with the places expected in the city map. However, as soon as the person loses his position again, he has to *relocalize* himself, which often requires similar steps to those performed on arrival in the city.

This example shows that knowledge about one's position within one's environment is a precondition for making use of a map. Autonomous indoor mobile robots face the same problems as the person when performing their tasks. Such tasks include, e.g., office delivery, surveillance, personal assistance, entertainment, manipulation of objects, and so on. Effective navigation is a basic precondition for successful mobile robot applications. In order to make use of a map of the environment and to carry out navigation plans, a successful application presupposes mobile robot localization. Therefore, localization is one of the fundamental problems in mobile robotics. Over the last few years, there has been a tremendous scientific interest in algorithms for estimating a robot's location from sensor data. We are particularly interested in methods that do not depend on any modification of the environment such as the placement of active beacons. As can be seen from the example, a successful system for autonomous mobile robot localization should address the following aspects.

- During the process of self-localization, ambiguous situations must be represented.
- Ambiguities can be resolved by a sequence of observations at different locations.
- The map of the environment can be used to find actions helping to uniquely determine the robot's location.
- Once the location of the robot is uniquely determined, tracking this location can be achieved more efficiently.
- Due to possible changes in dynamic environments, the map does not always match the environment.
- Localization includes the ability to detect failures of the position estimation and to relocalize the robot, when necessary.

In this thesis we will introduce an approach to mobile robot localization which addresses all these aspects of localization. Our approach to position estimation is a fine-grained grid-based implementation of Markov localization. The paradigm of Markov localization is based on a general framework for state estimation and applies probabilistic representations for the robot's location, the outcome of actions, and the robot's observations. The basic principles of our technique can be summarized as follows.

Global Localization: By global localization we mean the ability to estimate the position of a robot without knowledge of its initial location and the ability to relocalize a robot if its position is lost. In contrast with most existing position estimation techniques, our implementation of Markov localization can represent multi-modal probability distributions. Since Markov localization estimates the location of the robot in the entire environment, it is able to detect situations in which the position of the robot is lost. Therefore, our technique meets all requirements for global localization.

Robust Localization: Most localization techniques rely on a static model of the environment. Therefore, robust localization in dynamic environments requires the ability to estimate the position of a robot even when a significant number of the observations cannot be matched with the map. E.g. the appearance of typical indoor environments may change significantly due to furniture that is moved around, opened or closed doors, or people that surround the robot. In order to reliably localize a mobile robot even in such dynamic environments, our approach introduces a technique which filters sensor data. These filters are designed to eliminate the damaging effect of sensor data corrupted by unmodeled dynamics.

Active Localization: The efficiency especially of global localization can be improved by actively disambiguating between different possible locations. Actively controlling the actuators of a robot in order to localize it is especially rewarding whenever the environment possesses relatively few features that enable it to unambiguously determine its location. Key open issues in active localization are *"where to move"* and *"where to look"* so as to best localize the robot. In order to derive means for determining the best action with respect to localization, we will introduce a decision-theoretic extension of Markov localization. The guiding principle of our approach is to control the actuators so as to minimize future expected uncertainty. In our extension, uncertainty is measured by the entropy of future position probability distributions. By choosing actions to minimize the expected future uncertainty, our approach is capable of actively localizing a mobile robot from scratch.

Safe Localization: Active localization raises the problem of how to safely control a robot especially during the process of active global localization. Most existing approaches to safe navigation rely on a purely sensor-based, reactive collision avoidance. In order to overcome the limitations of this paradigm, we combined our method for position estimation with such a reactive collision avoidance technique. The resulting hybrid approach to collision avoidance differs from previous approaches in that it considers the dynamics of the robot and avoids collisions with invisible obstacles even if the robot is uncertain about its position.

All techniques introduced in this thesis will be extensively validated on the robot RHINO. The experiments are conducted in two different environments. One environment, a part of our computer science department, is depicted in Figure 1.1 (a). It includes all features of typical office environments such as people walking by, furniture that is moved around in the offices, and several places that look the same to the robot's sensors such as e.g. symmetric corridors. The other environment is a crowded museum, in which RHINO served as an autonomous museum tour-guide robot for extended periods of time (see Figure 1.1 (b)). This unstructured environment stresses additional requirements for robust localization such as accuracy and the ability to handle situations in which much of the robot's sensory information is corrupted by the presence of people. The experiments conducted in these two environments show that our implementation of Markov localization in combination with the extensions proposed in this thesis meets all requirements we have for a technique for autonomous mobile robot localization.

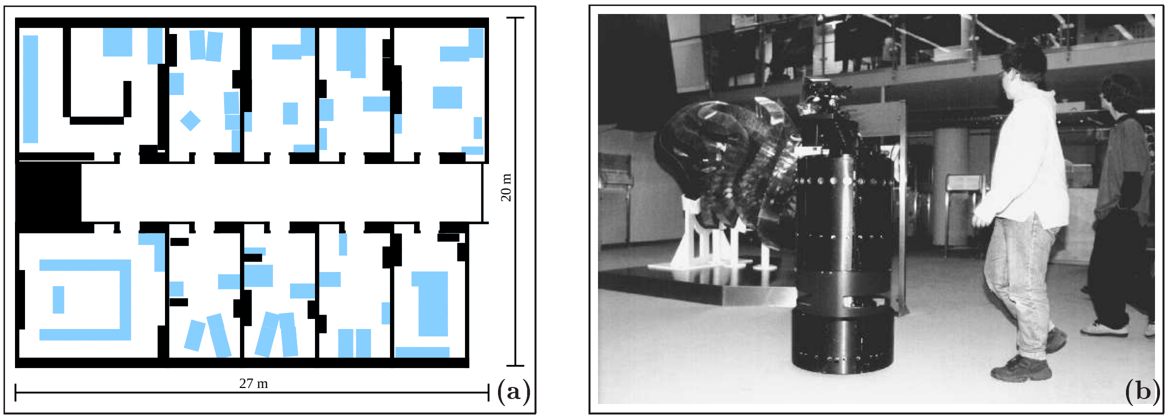


Fig. 1.1. (a) Map of our department and (b) RHINO as it gives a tour through the exhibition of the *Deutsches Museum Bonn*.

The remainder of this thesis is organized as follows. In the next chapter we will introduce the basics of Markov localization and an implementation of this general scheme. Than we will present the two indoor environments in which all our techniques will be tested throughout this thesis. Our extension for robust position estimation even in highly *dynamic* environments will be introduced in Chapter 3. A further extension of Markov localization is introduced in Section 4. This enhancement significantly increases the efficiency of position estimation by *actively* controlling the actuators of the robot. After introducing our hybrid approach to collision avoidance in Chapter 5, we will discuss the software architecture of the RHINO system with an emphasis on the integration of localization into robot control. Finally, we will conclude in Chapter 7.

Chapter 2

Markov Localization

2.1 Introduction

To navigate reliably in indoor environments, a mobile robot must know where it is. Over the last few years, there has been a tremendous scientific interest in algorithms for estimating a robot's location from sensor data. In the context of mobile robots, the general problem of localization can be stated as follows.

Given: A model of the environment such as a grid-based geometric description of obstacles or a topological map of the environment.

Task: Estimate the location of the robot within the environment based on observations. These observations typically consist of a mixture of odometric information about the robot's movements and information obtained from the robot's proximity sensors or cameras.

The majority of existing approaches to position estimation fall into the category of *local* techniques. Such approaches to localization aim at compensating for the odometric error occurring during robot navigation based on sensor data. They usually require that the initial location of the robot is known, and are only capable of *tracking* the location of a robot. Recently, several researchers proposed a new localization paradigm, called Markov localization [NPB95; SK95; KCK96; BFHS96a]. Markov localization is based on a probabilistic framework, which is expressive enough to represent situations in which the position of the robot is not uniquely determined. Therefore, Markov localization enables robots to localize themselves even under *global* uncertainty.

Our implementation of Markov localization uses position probability grids introduced in [BFHS96a]. This approach uses a fine-grained grid-based discretization of the state space. Such a representation has the advantage that it

- provides an accurate estimation of the position of a robot,

- makes no assumption about the outline of the environment and thus can be applied even in unstructured and non-rectangular environments, and
- integrates raw data from proximity sensors and is therefore able to exploit arbitrary geometric features of the environment.

Position probability grids are applied successfully in various indoor environments. With the optimizations introduced in this chapter, position probability grids are able to localize a robot from scratch and to efficiently track its position as it moves around.

In this chapter we introduce the general framework of Markov localization and our grid-based implementation of this technique. In Section 2.2 we will discuss the different aspects under which approaches to position estimation can be analyzed. A short overview of different techniques used for tracking the robot's position is presented in Section 2.3. In Section 2.4 we give an outline of the general framework of Markov localization. After discussing other approaches based on the paradigm of Markov localization, we present our implementation of Markov localization in Section 2.6. Two example applications of our implementation are presented in Section 2.7. These examples motivate the extensions of Markov localization introduced in the remainder of this work. Finally, we will discuss the general properties of our approach in Section 2.8.

2.2 A Taxonomy of Position Estimation Approaches

For the purpose of this thesis we categorize different approaches to localization along the following dimensions.

Local → Global

The first dimension depends on the localization task that can be handled by the methods. While *local* approaches assume that the position of the robot is specified within a small area, *global* approaches are more powerful because they are able to determine the location of a robot without knowledge of its initial position. Engelson called the problem of global localization the "kidnaped robot problem," suggesting that a robot should be able to localize itself even if some external force carried it to an unknown location [Eng94].

In order to deal with uncertain sensor information, most approaches to position estimation use a probabilistic representation of the position of a robot. In this context, the robot's position is modeled by a random variable and the state space of this variable consists of the locations considered for position estimation. We can classify these techniques for mobile robot localization along the local/global dimension by the expressiveness of the probability distributions they use to represent the robot's location: While one class of approaches only considers unimodal Gaussian probability distributions, other techniques aim at representing arbitrary distributions for the position of a robot.

The majority of existing localization approaches belongs to the first class. Most approaches of this class use Kalman filtering to integrate sensor information over time (see Section 2.3). They only consider a small fraction of the state space and are not able to represent ambiguous situations. Therefore, they are not able to localize a robot globally. Note, that a unimodal normal distribution would suffice for global localization if the robot would be able to detect *unique* landmarks. However, this cannot be assumed in the general case. Thus, local approaches usually require that the initial location of the robot is known and are designed to compensate for odometric error occurring during navigation. The strength of these techniques lies in their efficiency and accuracy.

Recently, several researchers proposed a new localization paradigm, called Markov localization [NPB95; SK95; KCK96; HK96; BFHS96a]. These approaches are able to probabilistically represent ambiguous situations and therefore enable robots to localize themselves under global uncertainty. We will discuss these approaches in more detail in the remainder of this chapter. Global approaches have two important advantages over local ones: First, they allow robots to operate with a higher degree of autonomy, since the initial location of the robot does not have to be specified. Second, even if the initial location is known, global approaches provide an additional level of robustness, due to their ability to recover from localization failure.

Static → Dynamic Environments

A second dimension along which localization methods can be grouped is concerned with the nature of the environment which they can master. The majority of existing approaches is based on the assumption that, according to the robot's sensors, the only aspect that may change over time is the robot's own location. This is typically only the case if the environment is static or if the dynamics of the environment cannot be perceived through the robot's sensors. Unfortunately, this is not true for most indoor environments, because their appearance can change significantly caused, e.g., by furniture that is moved around, opened / closed doors, or people that surround the robot. Thus, an important goal of research on mobile robot navigation is the development of methods that can localize a robot in dynamic environments.

Passive → Active Approaches

A third dimension is given, if we consider whether the localization method provides means to control the robot so as to best localize it. Passive localization exclusively addresses the estimation of the robot's position based on an incoming stream of sensor data. It rests on the assumption that neither robot motion, nor the pointing direction of the robot's sensors can be controlled. While passive approaches have the advantage that they do not interfere with the robot's primary tasks such as office delivery, they can be extremely inefficient. In most indoor environments, several places look the same to the robot's sensors. Therefore, especially during the process of global localization, it might be necessary to guide a robot to remote locations in order to disambiguate between the different possible positions of the robot. Active position estimation goes beyond the passive paradigm. It assumes that during localization, the localization routine has partial or full control over the robot, providing the opportunity to increase the efficiency and the robustness of localization.

The main goal of this thesis is to introduce an approach to position estimation which, according to this taxonomy, falls into the area of global, active localization in dynamic environments. The aspects of global localization will be addressed in this chapter. In Chapter 3, we will enhance our approach to global localization in order to make it applicable even in highly dynamic environments. Furthermore, we will show in Chapter 4, that the framework of Markov localization is well suited to provide means for active controlling the robot so as to best localize it.

2.3 Local Approaches to Position Estimation

Local approaches to mobile robot localization only consider a small fraction of the state space of the robot and are designed to compensate odometric error occurring during navigation. The fraction of all possible locations considered for position estimation is usually a small area centered around the estimated position of the robot. Good overviews of this class of techniques are given in [BEF96; CW90].

In order to estimate the position of a robot, Weiß and colleagues store angle histograms constructed out of laser range-finder scans taken at different locations in the environment [WWvP94]. The position and orientation of the robot during navigation is calculated by maximizing the correlation between histograms of the new measurements and the stored histograms. This maximum, together with odometric information, is used to initialize the match of the next scan with the corresponding stored angle histograms. Yamauchi applies a similar technique to estimate the position of a robot: He uses hill-climbing to match local maps built from ultrasonic sensors against previously built maps [Yam96]. The maps are built based on evidence grids [Mor88]. As in [WWvP94], the location of the robot is represented by the position yielding the best match.

In contrast to these methods, many existing approaches to position tracking use *Kalman filtering*, a technique well suited for integrating sensor information over time. This method was first proposed by Kalman [Kal60] and can be used to estimate the state of an arbitrary linear dynamic process. Each variable describing the state of the process to be modeled is represented by a Gaussian distribution. The parameters of this distribution are updated whenever a control is applied to the system and whenever new measurements from the sensors arrive. The two kinds of updates in Kalman filtering are often referred to as *prediction* and *correction*. In the prediction phase, the change of the state due to control actions is modeled. The correction phase corrects the state description by combining the estimation of the state with incoming sensor data (see e.g. [May90] for an introduction to Kalman filtering). When applied to position tracking for mobile robots, the process to be estimated is the x-y-α position of the robot within the environment. Here, the width of the Gaussian distribution represents the local uncertainty of the estimated position. Whenever the robot moves, the estimated position is shifted according to the distance measured by the robot's odometry. New sensory input is incorporated into the position estimation by matching the percepts against the world model. While the existing applications of Kalman filtering to position estimation are similar in how they model the motion of the robot, they differ mostly in how the likelihood of observations at the different locations in the environment is determined. In the remainder we will discuss some applications of Kalman filters to position tracking for mobile robots.

Leonard and Durrant-Whyte match beacons extracted from sonar scans with beacons predicted from a geometric map of the environment [LDW91a]. These beacons consist of planes, cylinders, and corners based on *regions of constant depth*. To update the position estimation, Cox matches distances measured by infrared range-finders against a line segment description of the environment [Cox91]. Gutmann and Schlegel extended this work to world models consisting of reference scans from laser range-finders stored during exploration [GS96]. They apply a technique introduced in [LM94] to match the scans against the world model.

Schiele and Crowley compare different strategies to track the robot's position based on occupancy grid maps and ultrasonic sensors [SC94]. They compute a local grid map using the most recent sensor readings. Such a local map is matched against the global grid map to produce a position and orientation estimate which is combined with the previous estimate using a Kalman filter. The experiments reported in this paper show that matching local occupancy grid maps against a global grid map results in good localization performance. Similar results have been achieved when extracting features from both maps before matching them. In a similar work, Shaffer et al. compare the robustness of two different matching techniques against different sources of noise [SGS92]. These techniques are based on laser range-finders: The feature-based estimation first extracts features from a sensor scan and matches them against features predicted from the map. An iconic estimator matches endpoints of the scan against line segments extracted from a map. They conclude by proposing a combination of the techniques in order to combine the benefits of both.

As noted above, all implementations of Kalman filtering for position estimation rest on the assumption that the position of the robot can be represented by a single Gaussian distribution. Several implementations have shown that this assumption is reasonable for robust and efficient position tracking. Unfortunately, this assumption is violated if the position of a robot has to be estimated from scratch, i.e. without knowledge of the starting position of the robot: Due to local similarities in the environment the position of the robot can only be determined uniquely, if enough sensor readings are collected at different places in the environment. During the process of global localization, the position estimation technique has to be able to represent ambiguous probability distributions. While this problem could in principle be solved with Kalman filters by including *any* possible starting location of the robot into the state variable, we will introduce a more general and expressive technique for state estimation in the next section. Several illustrative examples for ambiguous situations during global position estimation will be given in Section 2.7.

2.4 Basics of Markov Localization

Markov localization provides a general probabilistic framework well-suited for globally estimating the global position of a mobile robot based on observations and actions. We use the term "Markov localization" because it is a direct application of state estimation within the framework of "Partially observable Markov decision processes" (POMDP). POMDPs have their origins in the domain of operations research where they are used for making optimal decisions under uncertainty in both, acting and sensing. When applying POMDPs to a planning problem, the state of the world in which an agent acts is modeled by one or several random variables and the performance of the decision making agent is measured by the utility of the states of the world. This utility can be given, e.g., by the monetary wealth of a company or the distance of a mobile robot from its goal location. Problems concerning the complexity of POMDP-planning will be discussed shortly in Section 4.2. One of the main preconditions for making decisions is knowledge about the state of the world. Unfortunately, in most realistic environments the state of interest cannot be observed directly but can only be estimated based on uncertain sensory information. Hence, an important component of POMDPs is the *state estimator*, which is responsible for estimating the state of the world based on incoming sensor data and on the actions taken by the agent.

Markov localization is a special case of such a state estimator: Here, the agent is a mobile robot and the state of the world is the position of the robot within its environment[1]. To be more specific, let L_t denote the random variable representing the state of the robot at time t. $Bel(L_t = l)$ denotes the robot's belief that it was at location l at time t. This belief represents a probability distribution over the entire space of L. We use the term Bel instead of P for this particular probability density in order to emphasize that this corresponds to the state to be estimated. Here, l can be either a location in x-y-α space (where x and y are Cartesian coordinates and α is the robot's orientation) or a node in a topological representation of the environment. The state represented by L_t is updated

[1] We will keep our notation for probabilistic terms close to the notation used in [RN95].

whenever new sensory input is received or the robot moves. Without loss of generality we assume that at every discrete point i in time a measurement S_i is perceived and an action A_i is performed (the time index is incremented after the action). The task of Markov localization is to update the belief $Bel(L_t)$ for any location l in the environment as the robot moves.

In general, the state at time t has to be conditioned on *all* data available so far: $Bel(L_t \mid S_1, \ldots, S_t, A_1, \ldots A_{t-1})$ represents this belief given all previous actions A_i and percepts S_i. The computation of such a conditional probability grows exponentially with the number of conditioning variables. Even worse: the number of the conditioning variables A_i and S_i grows linearly in time. Before introducing the Markov localization algorithm, we discuss two important independence assumptions, which are necessary to derive an efficient rule for updating the state variable L as new input is received.

2.4.1 Independence Assumptions

The following two assumptions describe properties which an environment has to (at least approximately) fulfill in order to be applicable to Markov localization.

1. **Independence of actions**

   - *General case:* The state L_t at time t solely depends on the previous state L_{t-1} and the last action A_{t-1} performed by the agent or, in other words, all states and actions prior to time $t-1$ do not affect the state at time t once L_{t-1} is known. This assumption, as summarized in Eq. (2.1), is also called Markov assumption since it expresses the Markovian property of the environment.

     $$P(L_t \mid L_1 \ldots, L_{t-1}, A_1 \ldots, A_{t-1}, S_1 \ldots S_{t-1}) \;=\; P(L_t \mid L_{t-1}, A_{t-1}) \tag{2.1}$$

   - *Markov localization:* Knowledge about the position at time $t-1$ and about the motion command executed by the robot at time $t-1$ is sufficient to predict the position of the robot at time t. This assumption is reasonable in the context of mobile robot localization, since all actions and positions prior to $t-1$ provide no additional information about the current position of the robot.

2. **Independence of percepts**

   - *General case:* The percept S_t depends only on the state of the world at time t. Once we know L_t, all other measurements, prior states, and actions have no influence on the prediction of S_t.

     $$P(S_t \mid L_1 \ldots, L_t, A_1 \ldots, A_{t-1}, S_1 \ldots, S_{t-1}) \;=\; P(S_t \mid L_t) \tag{2.2}$$

     Such a conditional independence assumption has been successfully applied in different areas of probabilistic reasoning (see e.g. [Pea88]).

- *Markov localization:* All percepts depend on the position of the robot within the environment. E.g. given the position of the robot, the distance measured by a proximity sensor only depends on the distance to the obstacles in the environment and does *not* depend on the distance measured by another proximity sensor. Apart from robot localization, this assumption is frequently applied in map building for mobile robots (see e.g. [Mor88; Thr99b]).

Please note that the same independence assumptions are made by the localization techniques based on Kalman filtering. Both assumptions can be summarized as follows.

All relevant aspects of the world are modeled in the state variable(s) L and the conditional probabilities $P(L_t \mid L_{t-1}, A_{t-1})$ and $P(S_t \mid L_t)$.

In Chapter 3 we will discuss situations in which these assumptions are violated in the case of Markov localization and will propose a technique for filtering sensor measurements corrupted by non-modeled aspects of the environment.

2.4.2 Markov Localization Algorithm

In this section we derive an efficient algorithm for updating the position estimation as the robot moves and senses. The quantity of interest is a probability distribution representing the robot's belief of being at a location l. The belief $Bel(L_t)$ has to be conditioned on all previous actions a_i and percepts s_i. Please note, that we write the percepts s_i and actions a_i in small letters, because they are observed values for possible percepts and actions. By applying Bayes' rule we get Eq. (2.3) describing the belief of being at a location l at time t.

$$Bel(L_t = l \mid s_{1,\ldots,t}, a_{1,\ldots,t-1}) = \frac{P(s_t \mid L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1})\; P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})}{P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})} \tag{2.3}$$

For the reader's convenience we will omit the conditioning variables $s_{1,\ldots,t-1}, a_{1,\ldots,t-1}$ in the beliefs Bel whenever possible. This yields

$$\boxed{Bel(L_t = l) = \frac{P(s_t \mid L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots,t-1})\; P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})}{P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})} \quad (2.4)}$$

In the following we will show how to compute the different terms of this equation. The updated versions of Eq. (2.4) will be set in frames as well.

Sensor model

In Eq. (2.4), $P(s_t \mid L_t = l, s_{1,\ldots t-1}, a_{1,\ldots,t-1})$ describes the probability of observing s_t given the robots belief at time t and all previous percepts. With the assumption "independence of percepts" introduced in Eq. (2.2), this term can be rewritten as $P(s_t \mid L_t = l)$. Please note that this probability implicitly represents the map of the environment and is often referred to as the *sensor model*. Such a percept s_t can be e.g. a distance measured by a proximity sensor or an abstract feature such as a door-way observed in the environment. Since the model of the sensor is assumed to be static, we will write $P(s_t \mid l)$ instead of $P(s_t \mid L_t = l)$ in order to emphasize the independence of the time t. This yields the following updated version of Eq. (2.4).

$$\boxed{Bel(L_t = l) \quad = \quad \frac{P(s_t \mid l) \;\; P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})}{P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})} \qquad (2.5)}$$

The application of the sensor model to update the belief of the robot corresponds to the *correction phase* of the estimation update in Kalman filtering.

Action model

The term $P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ in Eq. (2.4) describes the estimate for being at location l immediately *after* action a_{t-1} has been executed and *before* any sensor measurement at time t has been perceived. Conditioning this term on the previous state L_{t-1} results in Eq. (2.6). This transformation is justified by the fact that $P(L_{t-1} = l')$ sums up to one over all l'. Now we can apply the Markov independence assumption defined in Eq. (2.1) and get Eq. (2.7).

$$
\begin{aligned}
& P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \\
= & \sum_{l'} P(L_t = l \mid L_{t-1} = l', s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \; P(L_{t-1} = l' \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \qquad (2.6) \\
= & \sum_{l'} P(L_t = l \mid L_{t-1} = l', a_{t-1}) \; P(L_{t-1} = l' \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) \qquad (2.7)
\end{aligned}
$$

The term $P(L_t = l \mid L_{t-1} = l', a_{t-1})$ is also called *action model* since it describes the change of the state due to actions. In Markov localization, this model describes the probability that the robot is at location l upon executing control a_{t-1} at a position l'. In most applications of Markov localization such control actions are measured by the robot's wheel encoders. In order to model uncertainty of the odometry, most implementations describe this conditional probability by a bounded Gaussian distribution centered at the predicted position of the robot (see [BFHS96a]).

Replacing $P(L_{t-1} = l' \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ in Eq. (2.7) by the belief at time $t-1$ yields Eq. (2.8). Here we also express the fact that $P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ only depends on the previous state L_{t-1} and on the action performed immediately before time t.

$$P(L_t = l \mid L_{t-1}, a_{t-1}) \;=\; \sum_{l'} P(L_t = l \mid L_{t-1} = l', a_{t-1})\; Bel(L_{t-1} = l'). \tag{2.8}$$

Again, we omitted the conditioning variables $s_{1,\ldots,t-1}, a_{1,\ldots,t-1}$ in the belief $Bel(L_{t-1} = l')$. Now we can rewrite Eq. (2.5) as

$$\boxed{Bel(L_t = l) \;=\; \frac{P(s_t \mid l)\;\; P(L_t = l \mid L_{t-1}, a_{t-1})}{P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})} \qquad (2.9)}$$

The update of the belief according to the action model corresponds to the *prediction phase* of the estimation update in Kalman filtering.

Normalization

The denominator $P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1})$ in Eq. (2.9) is a normalizer which ensures that $Bel(L_t)$ sums up to one over all locations. Again, we condition this term on the state L_t (see Eq. (2.10)). By replacing the summands according to the sensor and the action model we get Eq. (2.11).

$$\begin{aligned}
P(s_t \mid s_{1,\ldots,t-1}, a_{1,\ldots,t-1}) &= \sum_l P(s_t \mid L_t = l, s_{1,\ldots,t-1}, a_{1,\ldots t-1})\; P(L_t = l \mid s_{1,\ldots,t-1}, a_{1,\ldots t-1}) && (2.10)\\
&= \sum_l P(s_t \mid l)\;\; P(L_t = l \mid L_{t-1}, a_{t-1}) && (2.11)
\end{aligned}$$

In the remainder we will denote this normalizer by $P(s_t \mid L_t)$ in order to emphasize that this value describes the probability of observing s_t given the complete belief at time t.

Efficient Computation

Given the resulting Markov update formula in Eq. (2.12) we now are able to derive an incremental algorithm for updating the belief $Bel(L)$ as the robot moves or new sensory input is obtained.

$$\boxed{Bel(L_t = l) \;=\; \frac{P(s_t \mid l)\; P(L_t \mid L_{t-1}, a_{t-1})}{P(s_t \mid L_t)} \qquad (2.12)}$$

According to this equation, we can integrate motion and perception independently. At each point in time, the belief is first updated using the action model. This yields the probability $P(L_t \mid L_{t-1}, a_{t-1})$ which corresponds to the belief immediately before the percept s_t is made. The sensor measurement s_t is then integrated into the belief by applying the sensor model. Table 2.1 illustrates the different steps of this update rule. Again, we omitted the conditioning variables where possible.

For each location l update the belief Bel whenever ...

... **an action a is executed:**

- apply *action model*:

$$P(L_t \mid L_{t-1}, a_{t-1}) \;\leftarrow\; \sum_{l'} P(L_t = l \mid L_{t-1} = l', a) Bel(L_{t-1} = l') \tag{2.13}$$

... **new sensory input s is perceived:**

- apply *sensor model*:

$$Bel(L_t = l) \;\leftarrow\; P(s \mid l)\; P(L_t \mid L_{t-1}, a_{t-1}) \tag{2.14}$$

- *normalize*:

$$Bel(L_t = l) \;\leftarrow\; \frac{Bel(L_t = l)}{P(s \mid L_t)} \tag{2.15}$$

Tab. 2.1. The Markov localization algorithm

The belief $Bel(L_0)$ reflects the knowledge about the starting position of the robot. If the position of the robot relative to its map is entirely unknown, $Bel(L_0)$ is initialized by a uniform distribution. If the initial position of the robot is known with absolute certainty, then $Bel(L_0)$ is a Dirac distribution centered at this position. In both cases the belief update equations Eq. (2.13) and Eq. (2.14) are sufficient to refine the robot's belief upon sensing and moving.

2.5 Applications of Markov Localization

The framework of Markov localization is general enough to allow the use of arbitrary sensors and world models. Different variants of Markov localization have been developed and employed successfully at several sites [NPB95; SK95; KCK96; HK96; BFHS96a]. While all implementations of Markov localization rely on the update cycle presented in Table 2.1, the existing implementations can be distinguished particularly by the discretization of the state space and world model they rely on: In [NPB95; SK95; KCK96; HK96], a *topological* representation of the belief state is used, where each possible location l corresponds to a node in a topological map of the environment. Due to the nature of this representation, the percepts s are abstract features extracted from proximity sensors (e.g. a percept s can be the detection of a T-junction between two hallways within an office building). In contrast to these techniques, our implementation is based on a fine-grained, *geometric* variant of Markov localization. The methods proposed in [Thr98; Thr99a; OHD97] apply Markov localization in the context of machine learning. While we will introduce our approach in Section 2.6, the other approaches are discussed in this section.

2.5.1 Approaches Based on Topological Maps

Nourbakhsh et al. applied a technique for localization closely related to Markov localization during the *1994 AAAI-Mobile Robot Competition and Exhibition* [NPB95]. They use topological maps of the environment where the different nodes correspond to distinctive places in hallways such as openings or junctions and the connections between these places. Possible percepts s of the robot are the detection of a “closed door”, an “open door”, or an “open hallway”. The estimated position of the robot within the topological map is updated using a “state-set progression technique”: Whenever a perception is made, the robot is assumed to have moved at least one node within the map. Metric information about distances between the distinctive places is not used. The belief state is updated using certainty factors for the likelihood of detecting the features at the different locations in the environment. Here, actions are not considered explicitly and the task of localization is simplified by the assumption that the orientation of the robot is known.

The approach proposed by Simmons and Koenig is also based on topological maps [SK95; KS98]. In contrast to [NPB95] they additionally utilize metric information coming from the wheel encoders to compute state transition probabilities. Corridors are modeled by chains of states, where each state is one meter long. Uncertainty about the length of corridors is represented by several parallel chains of different length. In [KS98] they proposed a method for amending the accuracy of the length of corridors based on the Baum-Welch Algorithm [RJ86]. Here, corridors are assumed to be straight and perpendicular so that the orientation of the robot can be described by the four compass directions. Rooms and junctions are discretized into states of size $1 \times 1\text{m}^2$. The percepts of the robot are very similar to those in [NPB95]. Due to the uncertainty of ultrasonic sensors they do not map each sensor scan directly to a percept but first build local occupancy maps based on [Mor88] from several sensor scans. These maps are used to extract abstract features. The update of the belief state is performed according to the Markov update algorithm given in Table 2.1. Robot navigation is modeled within the framework of partially observable Markov decision processes: based on the belief estimated via Markov localization, heuristics are used for generating navigation plans for the robot.

Kaelbling et al. use an approach which is also based on the Markov equations to update the belief state [KCK96]. The main focus of their work lies in using the framework of POMDPs to generate close-to-optimal behavior of the robot. The discretization of the environment and the percepts of the robot are closely related to those in [SK95]. In [KCK96], they experimentally compare their approach to handle position uncertainty with the techniques proposed in [NPB95; SK95] (see also Section 4.2). Originally, the map was purely topologic, but in recent work Shatkey and Kaelbling [SK97] introduced a method for learning topological maps with weak odometric information using an extension of the Baum-Welch algorithm.

Hertzberg and Kirchner apply Markov localization to estimate the position of a robot within a system of sewerage pipes [HK96]. Due to the specific nature of the environment no metric information is provided in the topological graph describing the sewerage pipes. Nodes within the graph correspond to junctions and inlets. Here too, belief is updated according to the equations presented in Section 2.4.

2.5.2 Approaches Focusing on Learning the Sensor Model

The approaches presented in the previous section rely on an explicit topological model of the environment. As mentioned above, this model is "connected" to Markov localization by the probability of making percepts at the different locations within the map. The approaches proposed by Thrun [Thr99a; Thr98] and Oore et al. [OHD97] do not rely on such an explicit map of the environment: the environment is modeled *implicitly* in the sensor model $P(s \mid l)$, which is learned from observations s.

Thrun applies the framework of Markov localization for learning the sensor model $P(s \mid l)$ from sample data with neural networks [Thr99a; Thr98]. After learning, the environment is exclusively represented by the weights of the neural networks. The networks are trained to minimize the expected localization error on a sequence of position-percept pairs, where the percepts are given by a scan of sonar measurements and simple visual features. Each neural net is trained as a detector for a specific abstract percept extracted from the sensor values. Thrun shows that the learned detectors perform better than previously handcrafted visual feature detectors such as those used in e.g. [KW90]. Another important result of his work is that the kind of feature extracted from the environment strongly depends on the uncertainty in the position estimation assumed during learning: When the robot is extremely uncertain about its position it selects different landmarks than if it knows its position with high accuracy. In this work, the environment (a corridor) is modeled as one-dimensional. While this assumption is reasonable for hallway navigation, its applicability to arbitrary two-dimensional office environments might be difficult, especially due to the increase in the number of necessary training data.

Oore et al. use neural networks to learn the probabilities of distances s measured by sonar sensors at the different places l within the environment [OHD97]. The environment (an office) is discretized into small patches and the networks are trained in a supervised (pairs of measured distances and positions of the robot) and unsupervised version (distances and robot motion between percepts) of the learning algorithm. While experimental results show good performance in a $5 \times 5\text{m}^2$ office it remains unclear how the approach scales for larger environments.

2.6 Position Probability Grids

In contrast to the approaches discussed in the previous section, we use a fine-grained geometric discretization to represent the position of the robot (see [BFHS96a]). More specifically, L is represented by a fine-grained, regularly spaced grid, where the spatial resolution is usually between 10 and 15 cm and the angular resolution is usually 1 or 2 degrees. This grid-based approach is able to provide accurate position estimates. Whereas the coarse discretization of the topological approaches requires the definition of abstract features such as openings, doorways or other types of landmarks, our high resolution grids allow us to integrate raw (proximity) sensor readings. Hence our approach is able to exploit arbitrary geometric features of the environment.

The disadvantage of our grid-based method lies in the huge state space which has to be maintained. For a mid-size environment of size $30 \times 30\text{m}^2$, an angular grid resolution of $2°$, and a cell size of $15 \times 15\text{cm}^2$ the state space consists of $7,200,000$ states. Please notice that in the basic Markov localization algorithm given in Table 2.1, each of these locations has to be updated for any incoming sensor data and any registered motion command. In order to deal with such huge state spaces, two optimizations play a major role in making position probability grids work online. First, our approach applies a fast model for proximity sensors and, second, it uses a technique for selectively updating the belief state, thus focusing on the relevant part of the state space only. In the following two sections we will describe these techniques in more detail.

2.6.1 Fast Model of Proximity Sensors

To facilitate reading, the identifiers introduced in this section are summarized in Appendix A.

As mentioned above, the likelihood $P(s \mid l)$ that a sensor reading s is measured at position l has to be computed for any position l in each update of the position estimation (see Table 2.1). Therefore, it is crucial for online position estimation to have a model of proximity sensors which allows a fast computation of this likelihood. Moravec proposed a method to compute this quantity for sonar sensors based on occupancy grid maps [Mor88]. For each location l one has to compute a generally non-Gaussian probability density function over a discrete set of possible distances measured by the sensor. In a first implementation of our approach we used a method similar to Moravec's approach (see [BFHS96b]). Unfortunately, this method is computationally too expensive to be applied in our case: first, it is infeasible to compute this density online and a pre-computation for representing the complete densities for all possible states would by far exceed the memory of typical computers and especially those generally found on mobile robots.

The key idea of our fast sensor model is to compute the likelihood of a proximity measurement solely based on the distance to the next obstacle in the map. This distance can be extracted by ray-tracing from occupancy grid maps or CAD-models of the environment. In particular, we consider a discretization $d_1, \ldots, d_n$ of possible distances measured by a proximity sensor. In our discretization the size of the ranges $\Delta_d = d_{i+1} - d_i$ is equal for all i and d_n corresponds to the maximal range of the proximity sensor[2]. Let $P(d_i \mid l)$ denote the probability of measuring distance d_i if the robot is at location l. In order to derive this probability we consider the following two cases that can cause a certain distance to be measured by a proximity sensor (see also [Hen97]).

a.) **Known obstacle is detected**: If the sensor detects an obstacle represented in the world model, the resulting distribution is given by a Gaussian distribution with mean at the distance to this obstacle in the map. To be more specific, let o_l denote the distance to the next obstacle in the map given that the proximity sensor is fired when the robot is at position l. Eq. (2.16) describes the probability $P_m(d_i \mid l)$ of measuring distance d_i if the sensor detects this obstacle.

$$P_m(d_i \mid l) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{(d_i - o_l)^2}{2\sigma^2}} \tag{2.16}$$

The standard deviation σ of this distribution represents the uncertainty of the measured distance and depends on

- the granularity of the discretization of L, which represents the robot's position,

[2]In all our experiments d_n is set to 500cm.

- the accuracy of the world model, and
- the accuracy of the sensor.

Each of these three factors increases the uncertainty. We assume that the first two sources of uncertainty result in a Gaussian distribution of the distance to the next obstacle in the world model given that the robot is at position l. Given the distance to the next obstacle, we still have to consider the inaccuracy of the sensor when detecting this obstacle. This is done by convolving the distribution for the distance to the obstacle with another Gaussian distribution representing the accuracy of the sensor. Let σ_m denote the standard deviation modeling uncertainty in the position and the map and let further σ_s represent the sensor inaccuracy, respectively; then the resulting standard deviation σ is given by the following equation. $\sigma = \sqrt{\sigma_m^2 + \sigma_s^2}$ (see e.g. [HEK93]).

Figure 2.1 gives an example of such a Gaussian distribution for sonar and laser sensors. Here, the distance o_l to the next obstacle is 230cm, the standard deviation σ_m of the map is 16.8cm, and the standard deviation σ_s of the sonar sensors and laser range-finders is 16.0cm and 4.8cm, respectively[3]. For illustrative purpose we additionally plotted a Gaussian distribution with standard deviation σ_m, which represents the uncertainty solely due to the position discretization and the map. When comparing this dashed line with the other two curves, the high accuracy especially of the laser sensor becomes evident.

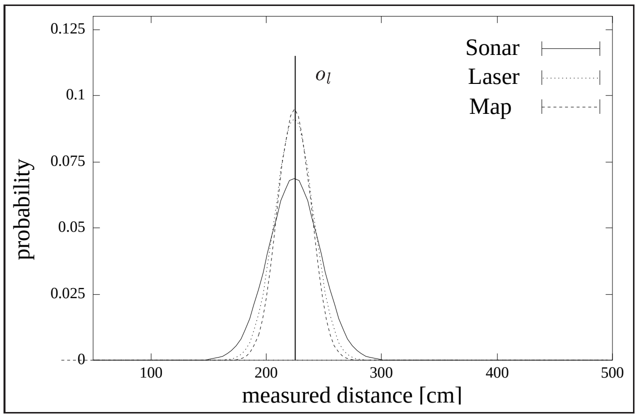


Fig. 2.1. Probability of measured distances if obstacle in distance o_l is detected.

b.) **Unknown obstacle is detected**: Typically, the world model is static and does not contain all relevant aspects of the environment. E.g. dynamic obstacles such as people are not represented in such static models. Especially due to these limitations there is

[3]These values are based on an approximation of real data (see next section).

a non-zero probability that the sensor is reflected by an obstacle not represented in the world model. Assuming that these obstacles are equally distributed over the entire environment, the probability $P_u(d_i)$ of detecting an unknown obstacle at distance d_i is independent of the location of the robot and can be modeled as a geometric distribution. This distribution results from the following observation. A distance d_i is measured if the sensor beam is *not* reflected by an obstacle at a shorter distance $d_{j<i}$ *and* is reflected at distance d_i. The resulting probability is

$$P_u(d_i) \;=\; c_r(1 - \sum_{j<i} P_u(d_j)). \tag{2.17}$$

Here, c_r is the probability that the sensor is reflected by an obstacle at a range Δ_d of the sensor discretization and $P_u(d_0)$ is initialized with zero. Such a distribution for sonar and laser measurements is depicted in Figure 2.2 (.) Here, we set the parameters to $c_r = 0.008$ and $\Delta_d = 4.0$cm for sonars, and $c_r = 0.005$ and $\Delta_d = 4.0$cm for lasers. The reader may notice that the high probability of measuring 500cm is due to the fact that this distance represents the probability of measuring *at least* 500cm, which is the maximal range of the sensors.

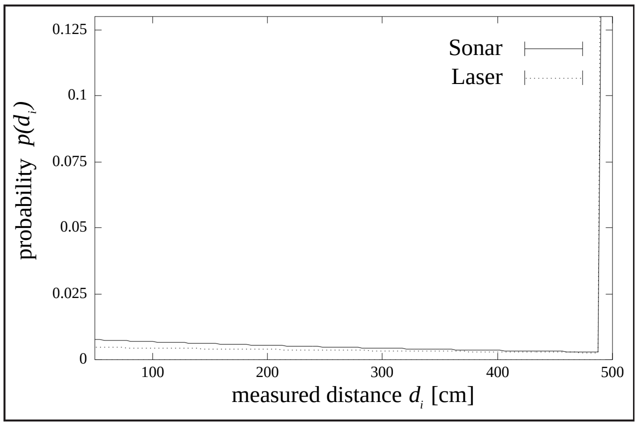


Fig. 2.2. Probability of measured distances due to unknown obstacles.

The reader may notice that sensor noise due to specular reflection or cross-talk is not explicitly considered in this sensor model. However, it is extremely difficult to decide whether a certain sensor reading is caused by specular reflection at map obstacles or by an unmodeled obstacle. Therefore we decided to include this kind of noise into the sensor model for unknown obstacles.

A sensor beam can only be reflected either by a known or by an unknown obstacle. Therefore, only *one* of these two cases can occur at a certain point in time. Nevertheless, we obtain a mixture of these two distributions on average.

Combining the Two Cases

In order to combine these two cases, it is not sufficient to sum up the distributions P_m and P_u, because they are not independent of each other. Therefore we have to consider the influence between the two cases. The combined probability $P(d_i \mid l)$ of measuring distance d_i if the robot is at location l can be generated by the following exclusive-or consideration. A sensor measures a distance d_i, if

a.) the sensor beam is

  1.) **not** reflected by an unknown obstacle before reaching distance d_i

  $$a_1 = 1 - \sum_{j<i} P_u(d_j),$$

  2.) **and** reflected by the known obstacle at distance d_i

  $$a_2 = c_d \; P_m(d_i \mid l)$$

b.) **or** the beam is

  1.) reflected **neither** by an unknown obstacle **nor** by the known obstacle before reaching distance d_i

  $$b_1 = 1 - \sum_{j<i} P(d_j \mid l)$$

  2.) **and** reflected by an unknown obstacle at distance d_i

  $$b_2 = c_r.$$

The parameter c_d in topic a_2 of this enumerations denotes the probability that the sensor detects the next obstacle in the map. The combined probability is summarized in Eq. (2.18), which can easily be transformed into Eq. (2.20).

$$\begin{aligned}
P(d_i \mid l) &= p(a_1 \wedge a_2 \;\vee\; b_1 \wedge b_2) && (2.18)\\
&= 1 - (1 - P(a_1 a_2)) \cdot (1 - P(b_1 b_2)) && (2.19)\\
&= 1 - (1 - (1 - \sum_{j<i} P_u(d_j)) \; c_d \; P_m(d_i \mid l))) \cdot (1 - (1 - \sum_{j<i} P(d_j)) \; c_r) && (2.20)
\end{aligned}$$

As can be seen here, the probability of a distance d_i only depends on probabilities for smaller distances $d_{j \leq i}$. Thus we can iteratively compute the whole distribution by starting at distance $d_0 = 0$cm. In order to consider the maximal range of the sensor we have to cut the distribution at the distance d_n:

$$P(d_n \mid l) \;=\; 1 - \sum_{j<n} P(d_j \mid l) \qquad (2.21)$$

To demonstrate the feasibility of our sensor model we collected several million real proximity measurements. For each measurement we additionally stored the distance o_l to the next obstacle in the map. From these data we were able to estimate the probability of measuring a certain distance d_i if the distance o_l to the next obstacle in the map is given. The dotted line in Figure 2.3 (a) depicts this probability for sonar measurements if the distance o_l to the next obstacle is 230cm. Again, the high probability of measuring 500cm is due to the fact that this distance represents the probability of measuring *at least* 500cm.

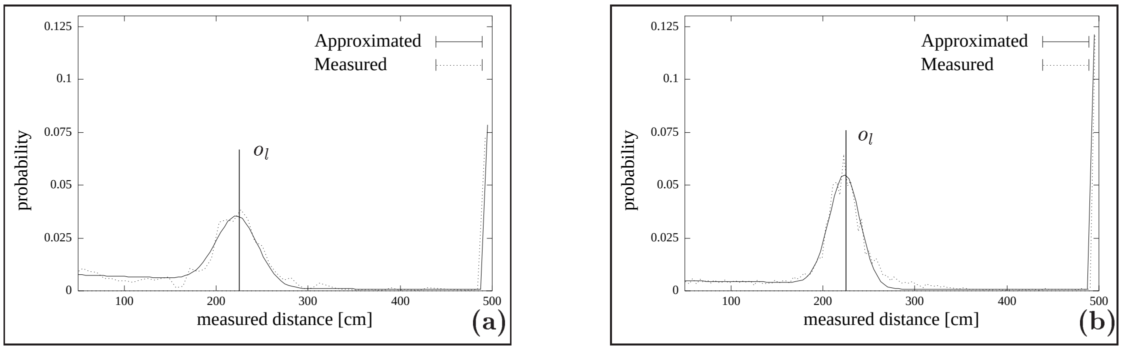


Fig. 2.3. Measured and approximated probabilities of (a) sonar and (b) laser measurements given the distance o_l to the next obstacle.

The solid line in the figure represents the distribution obtained by adapting the parameters of Eq. (2.20) so that the resulting distribution approximates the real probabilities. The corresponding measured and approximated probabilities for the laser sensor are plotted in Figure 2.3 (b). Note that the higher accuracy of the laser sensor is represented by a smaller standard deviation of the Gaussian distribution (see also Table 2.2). The probability of measuring a certain distance depends on the nature of the environment, the discretization of the robot's location, and the accuracy of the sensor. While we adapted the parameters of our sensor model "by hand" it is desirable to learn the parameters based on data collected in a certain environment. Hennig shows how to refine such a sensor model using an approach closely related to the expectation maximization algorithm [Hen97]. The parameters used in this example are summarized in Table 2.2.

Sensor	c_r	Δ_d	σ_s	c_d
Sonar	0.008	4.0 cm	16.0 cm	0.75
Laser	0.005	4.0 cm	4.8 cm	0.75

Tab. 2.2. Parameters for approximation of sonar and laser measurements.

The preceding figures illustrated distributions for only *one* expected distance o_l. The probabilities for *all* possible distances o_l to an obstacle for sonar and laser sensors are depicted in Figure 2.4 (a) and (b) and Figure 2.5 (a) and (b), respectively. In these figures, the distance o_l is denoted by expected distance. The similarity between the measured and the approximated distributions shows that our sensor model results in a good approximation of the real distribution. Please note the strong correspondence even between the probabilities of max range distances.

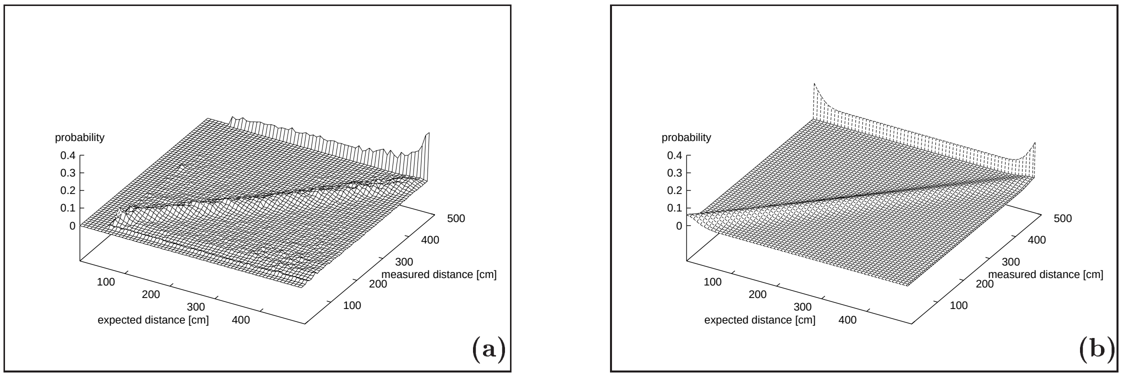


Fig. 2.4. (a) Measured and (b) approximated probability of sonar measurements.

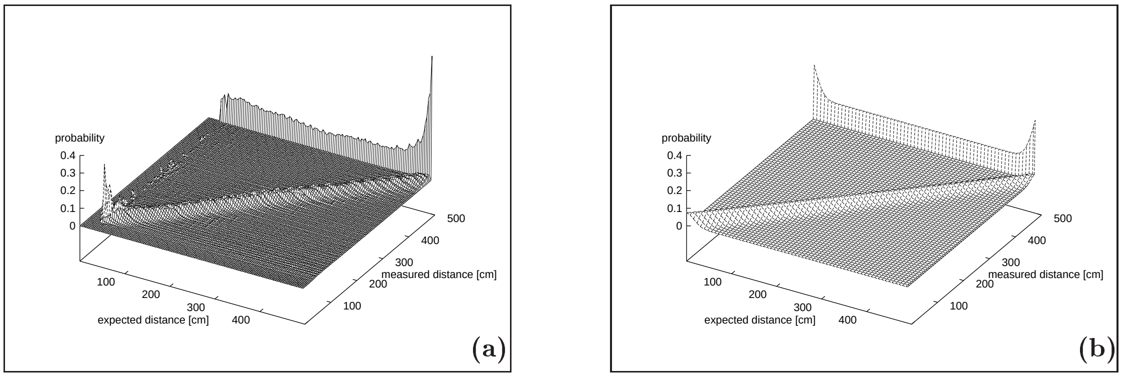


Fig. 2.5. (a) Measured and (b) approximated probability of laser measurements.

In this section we showed that the probability $P(d_i \mid l)$ of measuring a distance d_i if the robot is at location l is dominated by the distance o_l to the closest obstacle in the map. Based on this observation and the sensor model presented in this section, the probability $P(d_i \mid l)$ can be pre-computed for each location l and stored in two tables. The first, three-dimensional table `DistTab`, contains the pre-computed distances o_l to the next obstacle if the robot is at a location l and fires a sensor into a certain direction (these distances can be stored in one byte per location). The second, two-dimensional table `ProbTab`, comprises the probability $P(d_i \mid o_l)$ as shown in Figure 2.4 (b) and Figure 2.5 (b). Based on these tables, the probability of measured distances can be retrieved efficiently by the following nested table lookups: $P(d_i \mid l) = \texttt{ProbTab}[d_i][\texttt{DistTab}[l]]$.

2.6.2 Selective Update

An obvious disadvantage of the position probability grid approach to Markov localization is the size of the state space. The second optimization of our approach is a technique for a *selective* update of the belief state. The key idea of this approach is to exclude unlikely positions from being updated. For this purpose, we introduce a threshold[4] θ and approximate $P(s \mid l)$ for cells with $Bel(L_{t-1} = l) < \theta$ by $\tilde{P}(s)$. The quantity $\tilde{P}(s)$ is given by the average probability of measuring the feature s given a uniform distribution over all possible locations[5]. This leads us to the following update rule for a sensor measurement s (compare Eq. (2.14) and Eq. (2.15)):

$$Bel(L_t = l) \quad \longleftarrow \quad \begin{cases} \frac{P(s|l)}{P(s|L_t)} \cdot P(L_t = l \mid L_{t-1}, a) & \text{if } Bel(L_{t-1} = l) > \theta \\ \frac{\tilde{P}(s)}{P(s|L_t)} \cdot P(L_t = l \mid L_{t-1}, a) & \text{otherwise} \end{cases} \tag{2.22}$$

Please note that $\tilde{P}(s)$ differs from the normalizer $P(s \mid L_t)$, defined in Eq. (2.9): while $\tilde{P}(s)$ is computed for the a priori belief state, $P(s \mid L_t)$ reflects the probability of measuring s given the belief at time t. According to Eq. (2.22), all positions with probability below θ are updated with the same value $\alpha_t = \tilde{P}(s)/P(s \mid L_t)$. Thus we can rewrite the selective update scheme by

$$Bel(L_t = l) \quad \longleftarrow \quad \begin{cases} \alpha_t \cdot \frac{P(s|l)}{\tilde{P}(s)} \cdot P(L_t = l \mid L_{t-1} = l', a) & \text{if } Bel(L_{t-1} = l) > \theta \\ \alpha_t \quad \cdot \quad P(L_t = l \mid L_{t-1} = l', a) & \text{otherwise} \end{cases} \tag{2.23}$$

Now consider a partitioning of the state space L into n partitions $\pi_1, \ldots, \pi_n$. One partition π_i is called *active* at time t if it contains locations l with probability above θ; otherwise we call such a partition *passive* because the probabilities of all cells are below the threshold. With the selective update scheme represented in Eq. (2.23), we don't have to consider each location in a passive partition individually, but it suffices to keep track of the α-values only once for the whole partition. To do so, we store for each passive partition a value β_i. Whenever the probabilities of all locations in a partition fall below the threshold θ, β_i is initialized with 1 and is updated as follows: $\beta_i(t+1) = \alpha_t \cdot \beta_i(t)$. As soon as a passive partition becomes active again, we only have to multiply the β-value into the individual probabilities. The same idea holds for the movement of the robot: By keeping track of the motion performed since a partition became passive, it suffices to incorporate the accumulated motion whenever the partition becomes active again. In order to check whether a passive partition has to be activated again, we store the probability P_{max} of the most likely position in the partition. As soon as $P_{max} \cdot \beta_i(t) \geq \theta$ becomes true, the partition is activated again because it contains at least one position with probability above the threshold.

[4] In our current implementation θ is set to 1% of the *a priori* position probability.

[5] This is a conservative approximation, since the probability of the observations is at unlikely positions usually below the average.

By applying the selective update, the position probability grid approach is able to efficiently track the position of a robot once its position has been estimated uniquely. Furthermore, the ability to detect localization failures and to relocalize the robot remains inherent in our modified version of Markov localization. This can be seen if we compare the update of active and passive cells on incoming sensor data. According to Eq. (2.23), the difference lies in the ratio $P(s \mid l)/\tilde{P}(s)$. To see the influence of this ratio, we computed these values for our model of sonar and laser sensors. In this case $\tilde{P}(d_i)$ is obtained through Eq. (2.24) by assuming that all distances d_j to known obstacles are expected with probability $1/n$.

$$\tilde{P}(d_i) \quad = \quad 1/n \sum_j P(S = d_i \mid D = d_j) \tag{2.24}$$

An example of this ratio given an expected distance o_l of 230cm, is depicted in Figure 2.6.

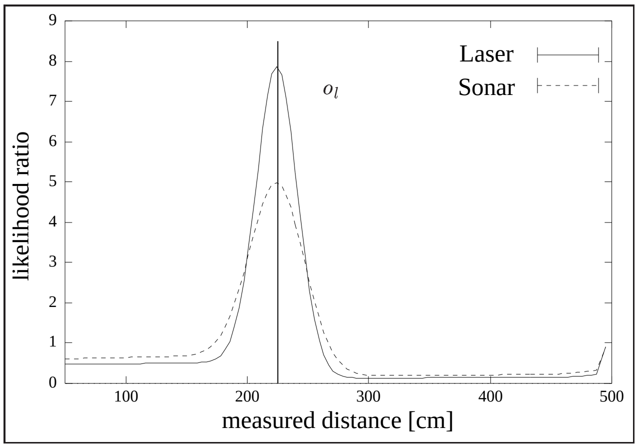


Fig. 2.6. Ratio $\frac{P(d_i|l)}{\tilde{P}(d_i)}$ for sonar and laser measurements for expected distance o_l.

Now the ability of the selective update to detect localization failures becomes obvious: If the position of the robot is lost, then the likelihood ratios for the distances measured at the active locations become smaller than one on average. Thus the probabilities of the active locations decrease while the probabilities in the passive partitions increase due to normalization until they are activated again. Once the true position of the robot is among the active locations, the robot can re-establish the correct belief.

In our current implementation we partition the state space L such that each part π_i consists of all locations with equal orientation. In extensive experimental tests we did not observe evidence that the selective update scheme impacts the robot's behavior in any noticeable way. In all our experiments we observed that the probabilities of the active locations sum up to at least 0.99. During global localization, the certainty of the position estimation increases and the density typically concentrates on the real position of the robot. If the position of the robot is known with high certainty, each scan comprising 180 laser measurements is typically processed in less than 0.1 seconds using a 200MHz Intel PentiumPro. In order to localize a robot in real-time, our implementation of position probability grids does not integrate *all* sensor data, but integrates only a random subset of the available data[6]. The percentage of data that is integrated into the belief depends on the required computation time and increases with the certainty of the position estimation.

The advantage of the selective update scheme is that the computation time required to update the belief state adapts to the certainty of the position estimation. However, we use a fixed representation of the state space, so that the space requirements of our implementation stay constant even when only a minor part of the state space is updated. To overcome this disadvantage, Burgard et al. [BDFC98] introduced a novel implementation of the position probability grid approach. This novel implementation is based on an octree representation of the state space. Here, only the relevant parts of the belief state have to be represented in the computer's memory. Another advantage of this implementation is that the discretization of the state space can be changed at runtime: If the robot is certain about its position, we use a very fine-grained discretization in order to enhance the accuracy of the position estimation (see [BDFC98] for more details).

2.6.3 Reduction of the Representational Complexity

The main advantage of Markov localization is its ability to represent arbitrary belief states. This ability is the key precondition for estimating the position of a mobile robot from scratch i.e. without knowledge about its starting position. In the remainder of this work we will introduce several extensions of Markov localization. These techniques consider the belief $Bel(L)$ when making decisions such as which percepts to filter or which action to perform next. In order to make these extensions work in real time, even the belief state reduced to the active locations as introduced in the previous section is too large. Thus we need a more compact representation of the belief state.

We generate such a representation by approximating L by a set L_m of local maxima m_i. Each local maximum is described by its position μ_i and the robot's belief of being there. We denote this belief by w_i because it can also be seen as the "weight" of the maximum. The positions μ_i of the maxima are computed at runtime, by first scanning the belief state for positions l_i for which the probability $Bel(L = l_i)$ exceeds a threshold θ_m and which are

[6]Please note that our laser range-finders generate about six scans of 360 distance measurements per second and the 24 sonar sensors are fired about four times per second

maximal within a small region $R(l_i)$[7]. Each l_i serves as a seed for a local maximum m_i. The position μ_i of such a local maximum is the expected location of the robot given that it is in the region around l_i. This location is determined by averaging over the region $R(l_i)$:

$$\mu_i = \frac{\sum_{l \epsilon R(l_i)} Bel(L = l) \cdot l}{\sum_{l \epsilon R(l_i)} Bel(L = l)}. \tag{2.25}$$

When summing up the locations l in Eq. (2.25), we treat the x-y-α coordinates of the locations independently. With this technique, the average distance between the global maximum and the true location of the robot is typically below 10cm for a grid resolution of 15cm (see Section 3.6 and [GBFK98]). Finally, the belief of being at a local maximum m_i is computed by summation over the region: $w_i = \sum_{l \epsilon R(l_i)} Bel(L = l)$.

This compact representation is somewhat justified by the observation that in practice, $Bel(L)$ is usually quickly centered on a small number of hypotheses and approximately zero anywhere else[8]. Please note that in our experiments the belief states are represented by less than 10 such local maxima when being applied for the extensions introduced in the remainder of this work. In most cases, the likelihoods of being in one of the regions associated with these local maxima summed up to more than 0.99.

2.7 Indoor Environments and their Challenges

In this section we will introduce two indoor environments in which our techniques for mobile robot localization will be tested throughout this thesis. Since the emphasis of this work is on the extensions of Markov localization, we will give here only examples for the application of position probability grids in these environments. At the end of each example, the specific challenges posed by the corresponding environment are discussed. Solutions to these challenges are the main focus of this thesis.

All experiments in this work are conducted using RHINO, an RWI B21 mobile robot. RHINO and its sensors are depicted in Figure 2.7. Please note that only the proximity sensors of this robot are applied for position estimation: Two laser range-finders providing 360 distances covering the surrounding of the robot in a height of approximately 30cm, and 24 ultrasound sensors mounted at a height of approximately 110cm. The reactive collision avoidance technique introduced in Chapter 5 additionally makes use of the infrared sensors, the tactiles and the camera system, if possible.

[7] In our current implementation θ_m is set to 100 times the average probability of being at a location l and the size of the region $R(l)$ is typically around $l_{x,y} \pm 30$cm and $l_\alpha \pm 4°$.

[8] Before this concentration of the belief state is established, the techniques introduced in the following chapters do not apply.

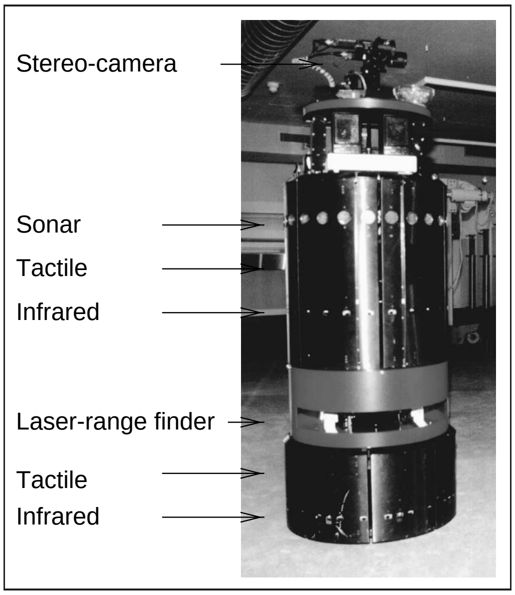


Fig. 2.7. Sensors of the mobile robot RHINO.

2.7.1 Office Environments

Our department at the University of Bonn is a representative example of an office environment. It provides all typical features of such environments: People walking by, offices are frequently refurnished, and several places that look the same to the robot's sensors such as e.g. symmetric corridors. Figure 2.8 depicts an outline of this department. Please note that even though the figure depicts all the furniture within the offices, only the walls and a small subset of the furniture can be detected with the robot's sonar sensors (the laser range-finders were not applied in this example). In this case only the black obstacles are detectable with RHINO's sonar sensors.

In this example we will illustrate the ability of our implementation of Markov localization to *globally* estimate the position of a mobile robot from scratch. The task of the method is to estimate the position of the robot while it moves through the environment. Here, the initial position of the robot is completely unknown and information about the position of the robot can only be derived from the robot's sonar sensors. The distances to the next obstacles in the map necessary for our sensor model introduced in Section 2.6.1 are extracted from the map depicted in Figure 2.8. The path of the robot during this example run is shown in Figure 2.9 (a). The robot starts in the corridor and first moves up and down the corridor by passing positions `1`, `2`, and `3`. Then it moves into `Room A` and `Room B` via position `4` and finally stops at position `5`.

Figure 2.9 (b) shows an outline of the environment along with the belief $Bel(L)$ as the robot passes position `1`. Here, the belief is projected on the $xy-$coordinates and

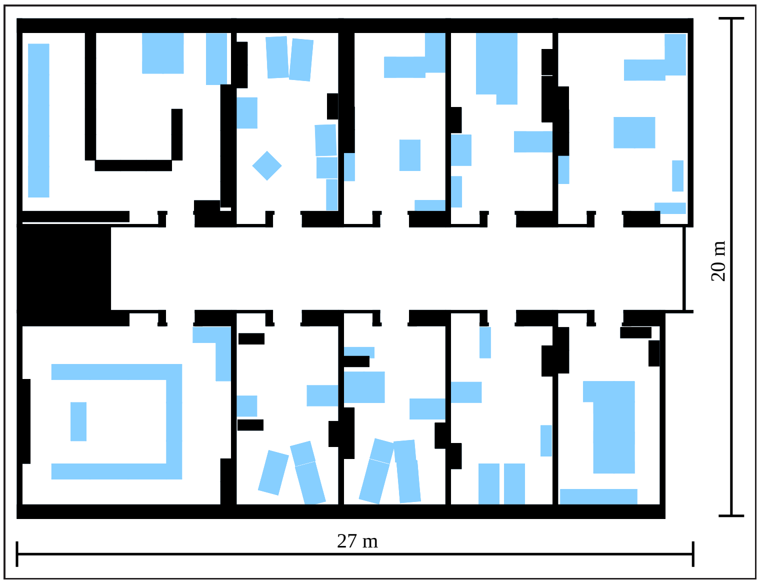


Fig. 2.8. Map of the department (only black obstacles can be detected by the sonar sensors).

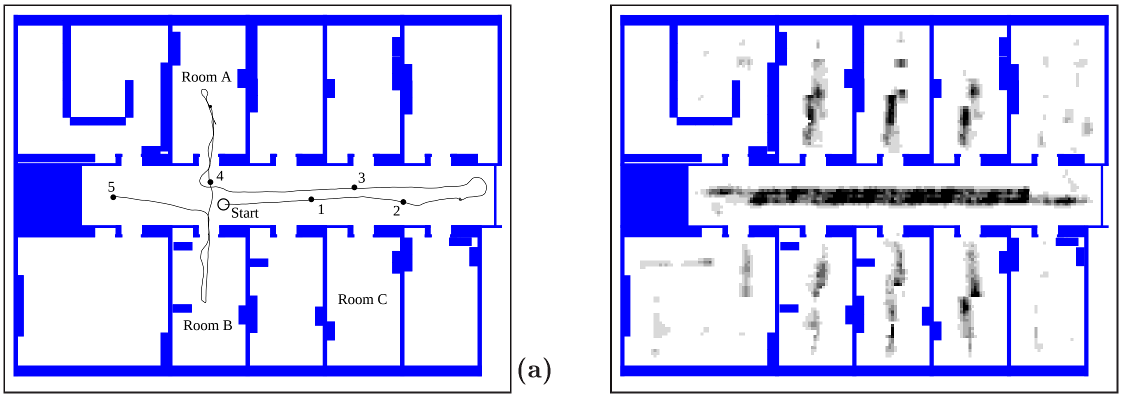


Fig. 2.9. (a) Environment and path of the robot and (b) belief $Bel(L)$ at position 1.

darker values represent higher probabilities. In this early stage of the global position estimation the probability distribution is spread out over most areas of the environment due to local symmetries. Note that such an ambiguous situation cannot be represented by the implementations of Kalman filter techniques.

After some more meters of robot motion the belief is still not concentrated on a single position, but the robot already detected that it is in the corridor (see Figure 2.10 (a)). As the robot reaches position 3 it has scanned the end of the corridor with its sonar sensors and hence the distribution depicted in Figure 2.10 (b) is concentrated on two local maxima. While the maximum labeled I represents the true location of the robot, the second maximum arises due to the symmetry of the corridor (position II is rotated by

180° relative to position I).

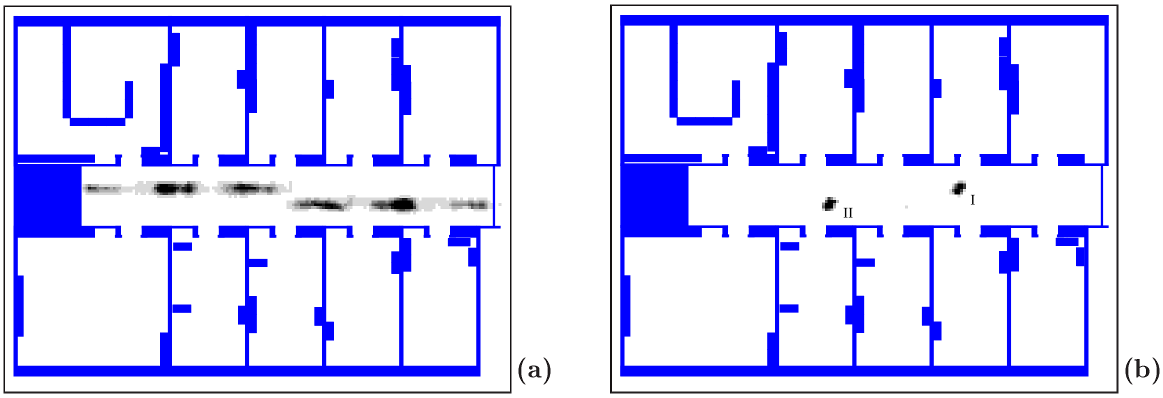


Fig. 2.10. Belief $Bel(L)$ (a) at position 2 and (b) at position 3.

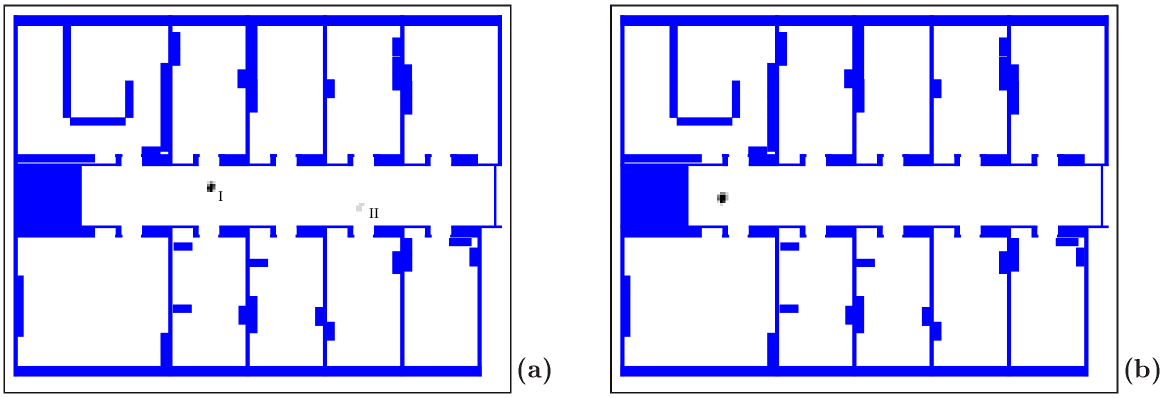


Fig. 2.11. Belief $Bel(L)$ (a) at position 4 and (b) at position 5.

Before reaching position 4, the robot moved into Room A. The different furniture in Room A and Room C helped the robot to disambiguate between positions I and II so that the probability of being at the correct position I is now higher than the probability of being at position II (see Figure 2.11 (a)). After having passed Room B the robot reaches the final position 5 and now uniquely determined its position correctly. The final belief of the robot is shown in Figure 2.11 (b).

Regarding this example, there are two main challenges posed by such an environment with respect to localization:

1. **Symmetry**: Different places in office environments typically look the same to the robot's sensors. Thus, in order to allow a robot to localize itself from scratch, it is important to apply a localization method which is expressive enough to handle situations in which the position is not determined uniquely. Markov localization is such a technique as we saw in our example run. In this example we guided the robot

into the offices in order to allow it to uniquely determine its position. To design mobile robots with a high degree of autonomy, it is desirable that robots do this on their own: Explore their environment in order to determine their location.

2. **Dynamics**: Apart from people walking by, especially the refurnishing of offices and opened / closed doors can significantly change the appearance of the environment. This raises the question of how to make a localization method robust against such dynamics effects.

These two challenges fall into the focus of this work: The problem of dynamic environments will be addressed in Chapter 3 and in Chapter 4 we will introduce a method for actively guiding a robot to places that help it to uniquely localize itself.

2.7.2 Deutsches Museum Bonn

In a recent attempt to move away from office-type environments into more difficult ones, the problem of accurate position estimation was a major prerequisite for successful mobile robot navigation. In particular, we recently installed our mobile robot RHINO in the *Deutsches Museum Bonn* (German Museum Bonn, Figure 2.12 (a)), where it served the function of an interactive robotic tour-guide. The robot's task was to engage visitors and guide them through the exhibition, providing verbal explanations for the various exhibits (see Figure 2.12 (b)). An overview of the different techniques applied in this project is given in [BCF$^+$98a; BCF$^+$98b] and the software architecture of the RHINO system with an emphasis on the integration of position estimation into mobile robot control will be discussed in Chapter 6.

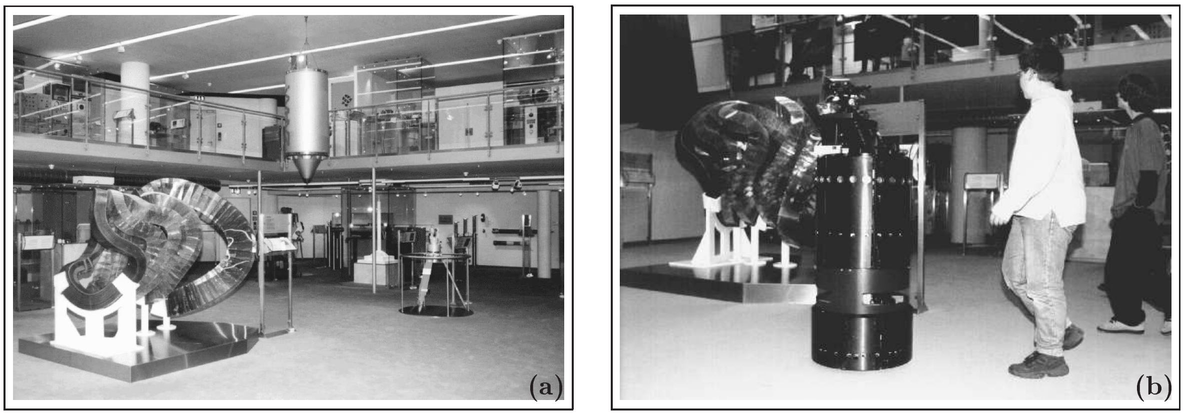


Fig. 2.12. (a) The exhibition of the *Deutsches Museum Bonn* and (b) RHINO as it gives a tour.

To ensure maximal safety in robot navigation, we applied RHINO's laser range-finders for position estimation during the tour-guide project. Figure 2.13 depicts an outline of the museum. Please note that only the black obstacles could be detected reliably with the robot's laser range-finders (sonar sensors were not applied in this environment). The grey obstacles were either not on the height of the laser sensors or made of glass. Thus, position estimation in the museum was based solely on these black obstacles.

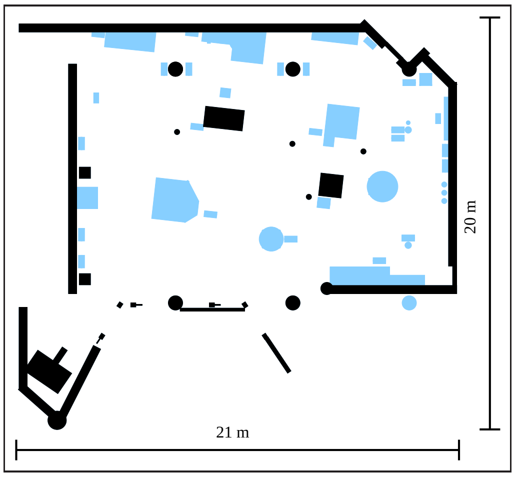


Fig. 2.13. Outline of the museum (only black obstacles are used for position estimation).

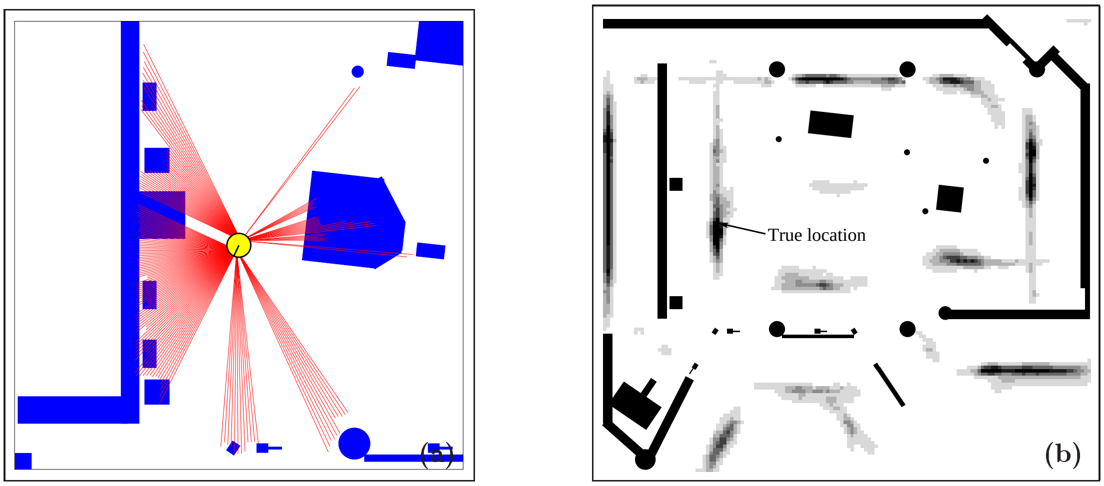


Fig. 2.14. (a) Scan of the laser range-finders and (b) position probabilities after integrating this scan into the belief.

As an example, consider the situation when the robot is placed in the museum and its position is completely unknown. Such a situation is represented by a uniform distribution of the position probabilities. Figure 2.14 (a) depicts a scan of the laser range-finders taken at the start position of the robot. Here, max range measurements are omitted and the relevant part of the map is shaded in grey. The belief after having integrated this sensor scan into the position estimation is plotted in Figure 2.14 (b) along with a map of the environment (only objects used for localization are depicted). Several local maxima in the distribution show that the position of the robot has not yet been uniquely determined.

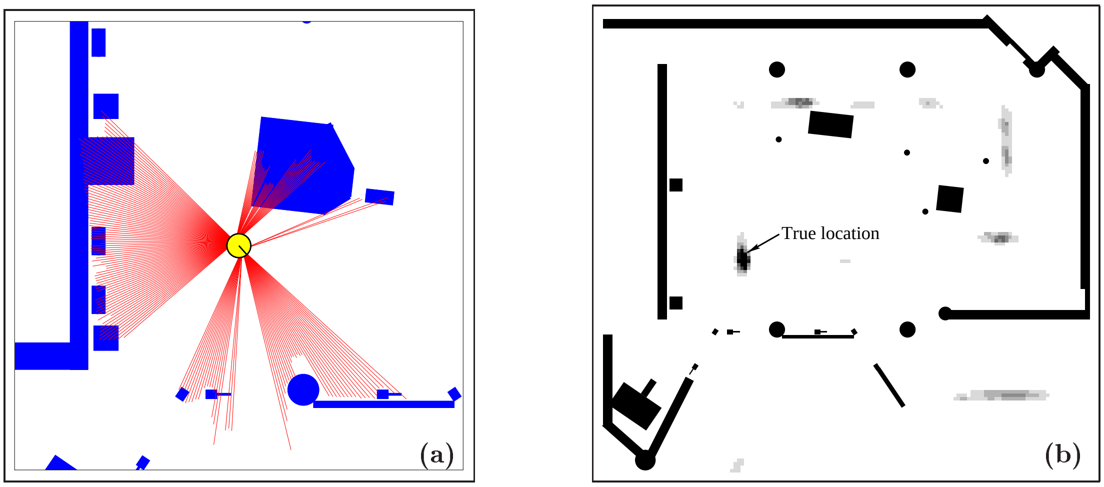


Fig. 2.15. (a) Second scan of the laser range-finders taken at another position and (b) position probabilities after integrating this scan into the belief.

Now the robot moves about 2 meters and integrates the next scan depicted in Figure 2.15 (a) into the position estimation. The updated belief is plotted in Figure 2.15 (b). As can be seen here, the certainty in the position estimation increases and the global maximum of the belief already corresponds to the true location of the robot. After integrating another scan into the belief the robot finally perceives the sensor scan shown in Figure 2.16 (a). Now the position is uniquely determined and the robot is highly certain of being at the true location (see Figure 2.16 (b)).

Once the position of the robot is uniquely determined, the estimation error is typically below 10cm as the robot moves through the museum. All computation is carried out in real-time, while the robot moves. Localizing the robot from scratch requires less than two minutes in the museum. As the certainty in the robot's position grows, our selective update scheme introduced in Section 2.6.2 allows us to process each sensor scan in less than 0.1 seconds, using a 200MHz Intel PentiumPro.

What made localization in the museum specifically challenging was the nature of the environment:

1. **Lack of features**: The problem of localization was particularly difficult in this museum due to the lack of features detectable by the robot's sensors and due to the large open space in the center portion of the environment.

2. **Sensor blockage**: The large number of people surrounding the robot made localization a challenging problem, since they often blocked a major fraction of RHINO's

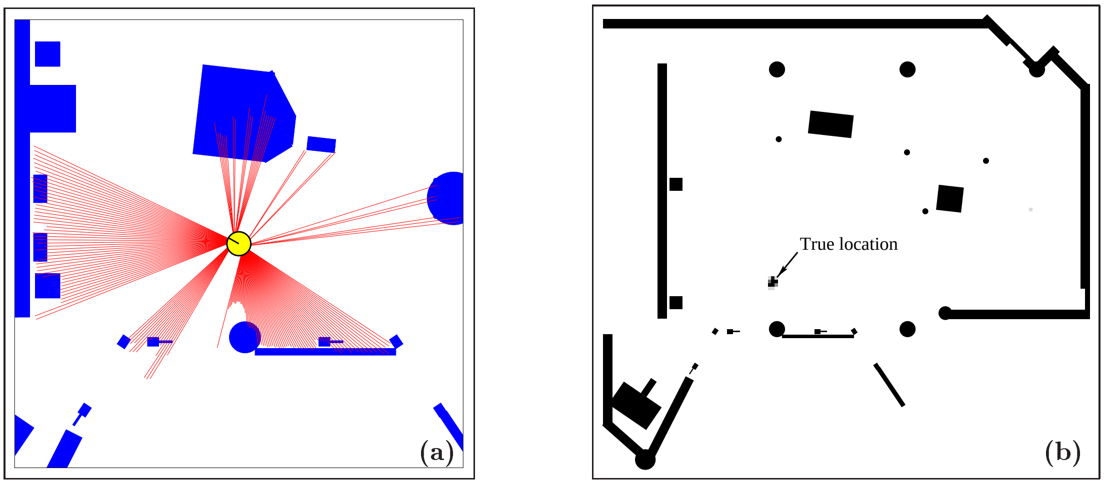


Fig. 2.16. (a) Fourth scan of the laser range-finders and (b) position probabilities after integrating this scan into the belief.

laser readings for extended durations of time.

3. **Invisible obstacles**: Even though RHINO applies four different sensor systems for obstacle detection (c.f. 2.7), various obstacles in the museum were virtually undetectable for the robot, such as e.g. glass cages, put up to protect exhibits, or metal plates on which exhibits were placed. While this seems to be a challenge for collision avoidance and not for localization, we will show in this thesis, that *accurate* position estimation is a precondition for safe navigation in such environments.

While the first problem is solved by position probability grids introduced in this chapter, the other two challenges are not addressed by Markov localization. Our solution to the problem of localization even when the robot's sensors are blocked will be introduced in Chapter 3. In order to ensure safe navigation even in environments with invisible obstacles, we developed a method for robot control which integrates sensor and map-based information into reactive collision avoidance. This technique will be discussed in Chapter 5.

The reader may notice that, although all environmental models used in this thesis are CAD-maps, position probability grids are able to localize a mobile robot even in learned occupancy grid maps. In [BFHS96b] we give examples where the robot successfully localizes itself in maps built from sonar data.

2.8 Discussion

In this chapter we introduced the paradigm of Markov localization to estimate the position of a mobile robot based on sensor data and its implementation using position probability grids. This approach applies probabilistic representations for the robot's location, the outcome of actions, and the robot's percepts.

Compared to *local* approaches to position tracking, which assume that the starting location of the robot is known, our approach inherits the benefits of a general technique for state estimation: It can represent multi-modal probability distributions and therefore is able to represent ambiguities. This is an important prerequisite to *globally* (re-)localize a robot from scratch. In work beyond the scope of this thesis we conducted an experimental comparison between our technique and a local approach to position estimation. In this comparison position probability grids have shown to be more robust against noise in the proximity sensors and the odometry of mobile robots (see [GBFK98] for exact results). On the other side, due to the discretization of the state space, our technique is less accurate than techniques based on scan matching: While the accuracy of our approach is typically higher than 10cm, scan matching techniques in combination with laser range-finders can reach arbitrary accuracies relative to the matched objects (see e.g. [GS96; AV98; VTG96]). However, in [BDFC98] we proposed a new implementation of position probability grids which is able to adapt the discretization of the state space in order to allow higher accuracy, if needed.

In contrast to the other implementations of Markov localization, our technique uses a fine-grained grid-based discretization of the state space. Due to this representation, our implementation has advantages over the applications based on topological descriptions of the environment:

- It provides an *accurate* estimation of the position of a robot.
- It can integrate *raw data* from proximity sensors and is therefore able to exploit arbitrary geometric features[9] of the environment.

The obvious disadvantage of position probability grids over the topological approaches is the size of the state space L that has to be maintained. Especially two techniques made it possible to efficiently deal with the resulting huge state space:

1. The *fast model* of proximity sensors allows to retrieve the probability $P(d_i \mid l)$ of sensor measurements by two subsequent table lookups.
2. The *selective update* scheme concentrates the computation on likely states only while still keeping the ability of the approach to recover from localization failures.

With these optimizations, position probability grids are able to localize a robot from scratch and to efficiently track its position as it moves around.

The test environments presented in this chapter raised warrants for further improvements of our implementation of Markov localization. In the subsequent chapters we will

[9] Of course these features must be detectable by the proximity sensors.

address the following questions. First, how can we attain robust position estimation even in dynamic environments? Here, the dynamics can result from reconfiguration of the furniture in the environment or from people surrounding the robot. Second, how can we design a localization procedure, which allows a mobile robot to efficiently localize itself from scratch without the need of a human supervisor? And finally we want to address the problem of how to utilize our localization technique in order to achieve safe navigation even in the presence of obstacles which cannot be detected by the robot's sensors.

Chapter 3

Localization in Dynamic Environments

3.1 Introduction

In the previous chapter we introduced position probability grids as an implementation of Markov localization. Over the last years, this approach has been shown to be robust in different kinds of environments. Occasional changes of the environment such as opened / closed doors or people walking by generally do not affect the performance of the position estimation. However, it typically fails to localize a robot if too many aspects of the environment are not covered by the world model. This is the case, for example, in densely crowded environments such as the *Deutsches Museum Bonn* during the tour-guide project, where the robot is naturally accompanied by crowds of people (see Figure 3.1). The same problem can be caused in office environments: If a major part of the furniture within the environment has been moved, the robot might lose track of its location if the world model is not updated according to these changes.

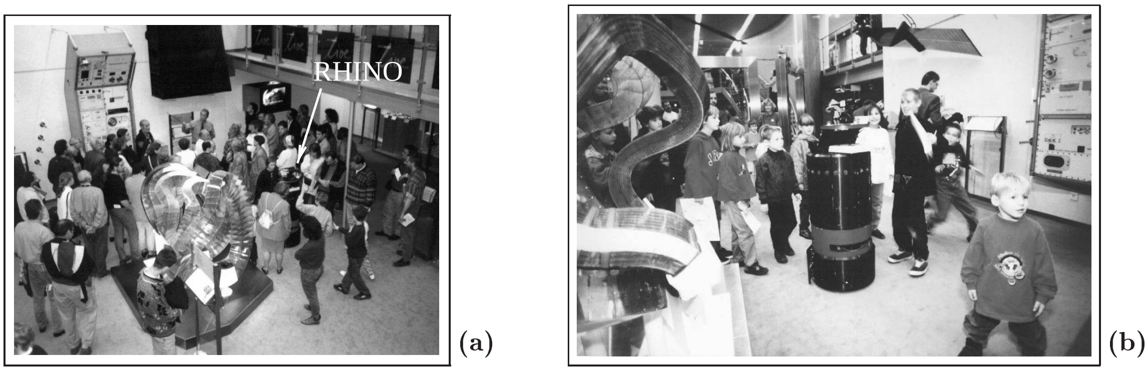


Fig. 3.1. RHINO surrounded by visitors in the *Deutsches Museum Bonn*.

In general, one can follow two kinds of approaches in order to deal with the problem of position estimation in dynamic environments. First, one can continuously update the model of the environment according to the dynamics. Second, one can rely on a partial static model and filter out data caused by dynamic effects. Unfortunately, it is extremely difficult, if not even impossible, to update a global world model so that it consistently keeps track of all changes in the environment. To avoid the problem of simultaneous map building and localization we follow the second approach to localize a mobile robot even under such difficult circumstances. Instead of continuously updating the world model, our approach relies on a model which represents only the static aspects of the environment and we extend the basic Markov localization scheme by *filtering* sensor data. These filters are designed to eliminate the damaging effect of sensor data corrupted by external (unmodeled) dynamics. In this chapter we propose and compare three such filters. The first filter is a general method for filtering sensor data in dynamic environments. It selects only those readings that increase the robot's certainty, which is measured by the entropy of the belief $Bel(L)$. The other two filters are especially designed for proximity sensors, as they attempt to filter such readings which with high probability are caused by obstacles *blocking* the objects represented in the model of the environment.

In an experimental study, our extended Markov localization is compared to the purely Markovian implementation of position probability grids introduced in Chapter 2. These experiments are conducted using data gathered in the *Deutsches Museum Bonn*. These data are an ideal test bed for localization in dynamic environments, as the robot often was surrounded by dozens of people which closely followed it around, thereby blocking most of its sensors for extensive periods of time.

The remainder of this chapter is organized as follows. In the next section we discuss related work and then analyze the violated independence assumption made by Markov localization in dynamic environments. Section 3.4 and 3.5 introduce the different filters for detecting corrupted sensor measurements. The advantage of our filter technique is validated experimentally in Section 3.6. Finally the different aspects of our approach are discussed in Section 3.7.

3.2 Related Work

The problem of mobile robot localization in environments where a major part of the sensor measurements is corrupted due to changes in the environment has found very little attention in the literature. One approach to deal with such situations could be to adapt the kind of sensors to the disturbances at hand: e.g. cameras pointed towards the ceiling as applied in [KW90] cannot perceive most of the changes that occur in typical office environments, and therefore make such techniques applicable even in environments that appear notoriously dynamic. However, such techniques depend on highly structured ceilings in order to enable accurate position estimation.

Another remedy to this problem would be to broaden the state estimation to simultaneously estimate the position of the robot and the state of the world. Approaches for concurrent map building and position estimation as proposed in [LM97; TFB98; LDW91b], unfortunately, require too much computational resource to be applied online. Furthermore, these approaches only seem to be applicable for mapping aspects of the environment that do not change very frequently such as e.g. walls, doors, or furniture. Obviously, crowds of humans surrounding the robot do not fall into this category of obstacles.

Yamauchi et al. estimate the position of a mobile robot based on sonar sensors by matching a local map against previously stored evidence grids of different places in the environment [Yam96; YLar]. The position yielding the best match is used as the estimated location of the robot. While this approach can deal with situations in which furniture has been (re-)moved in an office environment, it remains unclear whether it can handle situations in which a major part of the robot's surrounding changes.

In [GBFK98] Gutmann et al. compared a previous version of our extended Markov localization technique to the approach introduced in [LM94]. This approach uses Kalman filters to match scans of laser range-finders into a previously built map. First results generated with data from our museum tour-guide project show that their technique is robust against dynamics within the environment by filtering whole scans before matching them into the world model. However, this approach does not have the ability to globally (re-)localize a mobile robot in ambiguous environments.

3.3 Violation of Independence Assumption

In this section we discuss the reason why Markov localization fails to estimate the position of a robot in extremely dynamic environments. The assumption of independence of percepts as described in Eq. (2.2) is necessary for the correct derivation of the Markov localization algorithm presented in Section 2.4.2. This assumption states that the sensor measurements observed at time t solely depend on the corresponding state L_t of the world. In other words: Knowledge about the percepts and actions that occurred prior to t provide no additional information about the percepts at time t. This assumption is only justified if all relevant aspects of the world are modeled in the state variable L, which typically only contains the position of the robot. In the case of localization in densely populated environments, this assumption is clearly violated when using a static model of the world.

To illustrate this point, Figure 3.2 (a) shows an example situation where RHINO has been projected into the map of our department at its correct position. The lines indicate the current proximity measurements obtained from the robot's two laser-range finders (maximal range measurements are omitted). In this situation, the independence assumption holds for the individual readings: each measurement of the sensor scan solely depends on the position of the robot within the map. Knowledge about the other or previous readings provides no additional information to predict this certain measurement.

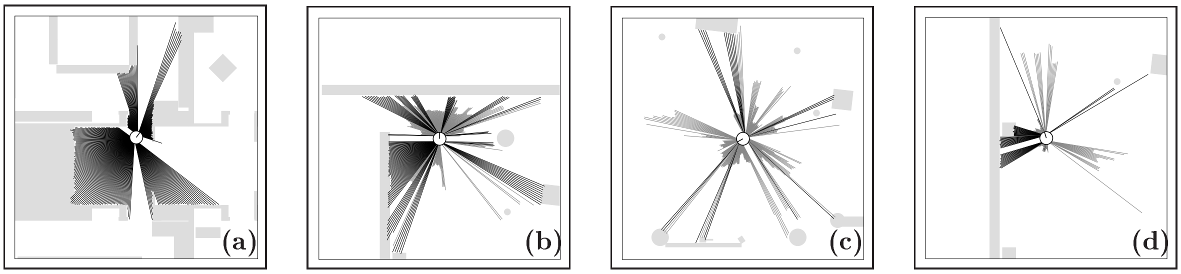


Fig. 3.2. Situations, in which the Markov independence assumption (a) holds or (b,c,d) is violated.

Figure 3.2 (b)-(d) give situations in which the sensor scan is corrupted by obstacles not represented in the world model: the different shading of the measurements indicates the two classes they belong to: the black values correspond to static obstacles that are part of the map, whereas others are caused by humans and thus violate the Markov assumption. The proximity of people usually increases the robot's belief of being close to modeled obstacles, which has the effect that the robot quickly loses track of its position when relying on *all* sensor measurements.

To reestablish the independence of percepts we could try to include the position of the robot *and* the position of people into the state variable L. Unfortunately, the computational complexity of state estimation increases exponentially in the number of dependent state variables to be estimated. Our approach to solve the problem of non-modeled aspects in the environment is to develop filters which select those readings of a complete scan which with high likelihood are *not* due to static obstacles in the map thus making the system more robust against such kind of noise. It should be noted that we will introduce one filter which is closely related to the problem of estimating the position of people.

3.4 Entropy Filter

The first filter used in our implementation is called *entropy filter* and works as follows. The *entropy* $H(L)$ of a belief over L is defined by

$$H(L) \quad = \quad -\sum_{l} Bel(L=l)\ \log Bel(L=l) \tag{3.1}$$

Entropy is a measure of uncertainty about the outcome of a random variable ([CT91]). The larger the entropy of the state estimation, the higher the robot's uncertainty as to where it is. The *entropy filter* measures the relative change of entropy upon incorporating a sensor reading into the belief $Bel(L)$. More specifically, let s denote the measurement of a sensor (in our case a single range measurement). The change of the *entropy* of $Bel(L)$ given s is defined as:

$$\Delta H(L \mid s) \quad := \quad H(L) - H(L \mid s) \tag{3.2}$$

$H(L \mid s)$ is the entropy of the belief $Bel(L \mid s)$ defined by the Markov update equations Eq. (2.14) and Eq. (2.15). While a negative change of entropy indicates that after incorporating s, the robot is less certain about its position, a positive change indicates an increase in certainty. The selection scheme of the entropy filter is as follows.

Exclude all sensor measurements s with $\Delta H(L \mid s) < 0$.

Thus, the entropy filter makes robot perception highly selective, in that it considers only sensor readings confirming the robot's current belief.

3.5 Filters for Proximity Sensors

While the entropy filter makes no assumptions about the nature of the sensor data and the kind of disturbances to expect in dynamic environments, we will now describe two filters especially designed for proximity sensors such as sonar sensors or laser range-finders.

3.5.1 Distance Filter

This filter is called *distance filter*, since it selects sensor readings based on their distance relative to the distance to the closest obstacle in the map. To be more specific, a measurement s is assumed to be corrupted, if it is *shorter* than the distance expected from the world model. This filter removes those sensor measurements s which with probability higher than θ (this threshold is set to 0.99 in all experiments) are shorter than expected.

To see, consider a discrete set of possible distances $d_1, \ldots, d_n$ measured by a proximity sensor. Let $P_m(d_i \mid l)$ denote the probability of measuring distance d_i if the robot is at position l and the sensor detects the next obstacle in the map. The distribution P_m describes the sensor measurement *expected* from the map. This distribution is assumed to be Gaussian with mean at the distance o_l to the next obstacle. See Section 2.6.1 for a more detailed discussion of this sensor model. The dashed line in Figure 3.3 represents P_m for a laser-range finder and a distance o_l of 230cm. Given this distribution for the expected measurement, we are interested in the probability that a measured distance d_i is *shorter* than the expected one. This probability is equal to the probability that the expected sensor reading is *longer* than the measured one. Thus we can compute $P_{\text{short}}(d_i \mid l)$, the probability of interest, by summation over all distances d_j bigger than d_i:

$$P_{\text{short}}(d_i \mid l) \quad = \quad \sum_{j>i} P_m(d_j \mid l). \tag{3.3}$$

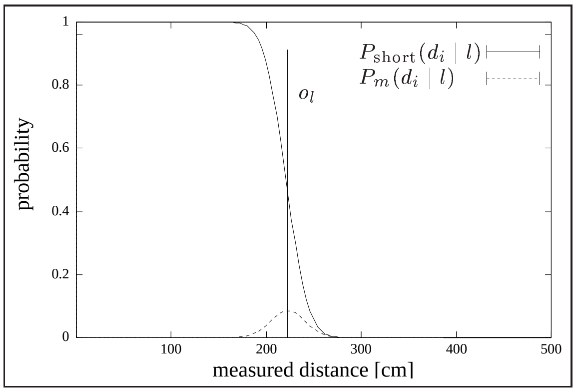


Fig. 3.3. Probability $P_m(d_i \mid l)$ of expected measurement and probability $P_{\text{short}}(d_i \mid l)$ that a distance d_i is shorter than the expected measurement.

Until now we have assumed that the position l of the robot is known with absolute certainty. In order to consider the belief state of the localization we have to average over all possible locations of the robot. Eq. (3.4) describes the probability that d_i is shorter than expected with respect to the belief $Bel(L)$.

$$P_{\text{short}}(d_i) \quad = \quad \sum_l P_{\text{short}}(d_i \mid l) Bel(L = l) \tag{3.4}$$

The selection scheme of the distance filter is as follows.

Exclude all sensor measurements d_i with $P_{short}(d_i) > \theta$.

This filter makes the implicit assumption that the shortness of a sensor reading corresponds to the probability that this sensor reading is caused by an unmodeled obstacle.

3.5.2 Blockage Filter

In this section we will derive a filter, which *explicitly* models the relation between the measured distance and the probability that this measurement is caused by an unmodeled obstacle. This filter is based on the estimation of the position of unknown obstacles in the environment relative to the robot. The model for proximity sensors applied by our grid-based Markov localization as described in Section 2.6.1 is dominated by the distance to the next obstacle in the map. In order to minimize the damaging effect of corrupted sensor readings this filter aims at detecting those sensor readings which are caused by an unknown obstacle *blocking* the next obstacle represented in the map. For the reader's convenience we will refer to unmodeled obstacles as humans, since in the case of a sufficiently complete map of the environment most unknown obstacles are humans.

To see how the blockage filter works, let us first assume that we can estimate $P(h_j \mid d_i, l)$, namely the probability that a human is at distance h_j, if the robot is at location l and the proximity sensor measures distance d_i. Given this quantity, we can compute $P_{\text{blocked}}(d_i \mid l)$, i.e. the probability that the closest obstacle in the map is blocked by a human if the robot is at location l and the sensor measures distance d_i. This probability is based on the probability that a human is closer to the robot than the next obstacle in the map:

$$P_{\text{blocked}}(d_i \mid l) \quad = \quad \sum_{j<o_l} P(h_j \mid d_i, l) \tag{3.5}$$

Here, o_l is the distance to the next obstacle in the map (see also Section 2.6.1 and Appendix A). By averaging over all possible locations of the robot, we get the probability $P_{\text{blocked}}(d_i)$ that a sensor measurement is caused by a human blocking a map obstacle:

$$P_{\text{blocked}}(d_i) \quad = \quad \sum_{l} P_{\text{blocked}}(d_i \mid l)\; Bel(L = l) \tag{3.6}$$

Based on this equation we can detect sensor readings which with high likelihood are caused by a human. The selection scheme of the blockage filter is as follows.

Exclude all sensor measurements d_i with $P_{\text{blocked}}(d_i) > \theta$.

In order to implement this filter, it remains to derive a probabilistic model for the distance of humans given a certain sensor measurement. This distribution is denoted by $P(h_j \mid d_i, l)$. In the following sections we will proceed as follows. First we will introduce the different probability distributions, on which $P(h_j \mid d_i, l)$ depends. We will show that this probability mainly depends on an extended sensor model, which describes the probability of measuring a certain distance if the robot is at location l *and* a human is at distance h_i. This extended model of proximity sensors will be introduced in the subsequent section.

Estimating the Distance to Humans

Let $P(h_j \mid d_i, l)$ denote the probability that a human is in distance h_j, if the robot is at location l and the proximity sensor measures distance d_i[1]. This probability can be computed using the Bayesian inversion equation:

$$P(h_j \mid d_i, l) \quad = \quad \frac{P(d_i \mid h_j, l)\; P(h_j \mid l)}{P(d_i \mid l)} \tag{3.7}$$

Here, $P(d_i \mid l)$ is the probability of measuring distance d_i if the robot is at location l. Please note, that this quantity is our model of the proximity sensor as introduced in Section 2.6.1. $P(h_j \mid l)$ denotes the probability that a human is at distance h_j if the robot is at location l.

[1] These and the following identifiers are summarized in Appendix A.

This quantity can be estimated as follows. According to our sensor model, we assume that on average a range Δ_d of the discretization is occupied with probability c_r by a human. By additionally assuming that a human is *not* at the position which is occupied by the closest obstacle in the sensor direction, we get the following probability distribution for humans relative to the robot.

$$P(h_j \mid l) = c_r(1 - P_{occ}(d_i \mid l)) \quad (3.8)$$

Here $P_{occ}(d_i \mid l)$ is the probability that the distance d_i is occupied by the closest obstacle if the robot is at location l. It is straightforward to extract this quantity from our geometric model of the environment.

To complete the derivation of Eq. (3.7), it remains to model the probability $P(d_i \mid h_j, l)$ of measuring distance d_i if the robot is at location l and a human is at distance h_j. Please note that this probability is closely related to our sensor model $P(d_i \mid l)$. The difference between our previous sensor model and this probability is that apart from knowing where the robot is, we additionally have to consider that a human stands in a distance h_j. In the next section we will show how to extend the sensor model $P(d_i \mid l)$ in order to incorporate knowledge about humans.

Extended Model of Proximity Sensors

If we know the position of the robot and the distance to a human along the sensor beam, then we can distinguish three different kinds of obstacles on the sensor beam. These obstacles are:

a.) The closest obstacle in the map,

b.) an unknown obstacle, and

c.) the human.

The first two kinds of obstacles have already been discussed in Section 2.6.1. The probability of measuring a distance d_i if a human stands at distance h_j will be denoted by $P_h(d_i \mid h_j)$. This probability can be modeled by a Gaussian distribution with mean at the distance to the human. The uncertainty of the sensor is modeled by the standard deviation σ_s of this distribution.

Example distributions for the three kinds of obstacles are depicted in Figure 3.4. The solid line with the continuously low values shows the probability $P_u(d_i)$ that the sensor measurement d_i is caused by an unknown obstacle. The dashed line represents $P_m(d_i \mid l)$, namely the probability of measuring a distance d_i if the robot is at location l. In this example we assume that the distance o_l to the next obstacle in the map is 230cm. Please notice that the parameters used for these two distributions are taken from the sensor model that has been adapted to data collected during the museum tour-guide project. Finally, the dotted graph depicts the probability $P_h(d_i \mid h_j)$ if a human stands in a distance of 120cm.

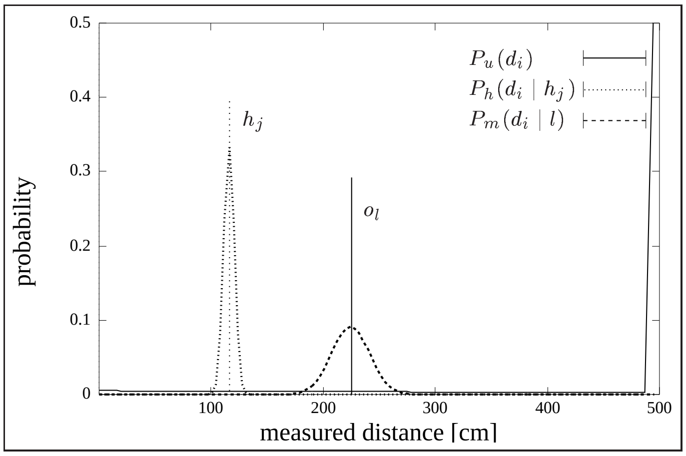


Fig. 3.4. Probabilities of measuring a distance d_i given unknown obstacles ($P_u(d_i)$), the closest obstacle in the map ($P_m(d_i \mid l)$), or the distance to a human ($P_h(d_i \mid h_j)$).

As can be seen in the figure, the standard deviation of the distribution $P_m(d_i \mid l)$ is larger than the standard deviation of $P_h(d_i \mid h_j)$. This is based on the following consideration: Both distributions model the uncertainty of the sensor by a standard deviation σ_s. But while P_h assumes exact knowledge about the distance h_j to the human, the distribution P_m suffers additional uncertainty σ_m due to the inaccuracy of the world model and due to the discretization of the position L. Thus, the standard deviation of P_m is given by $\sigma = \sqrt{\sigma_m^2 + \sigma_s^2}$ while the standard deviation of P_h is given solely by $\sigma = \sigma_s$.

The probability $P(d_i \mid h_j, l)$ of measuring distance d_i if we know that the robot is at location l and a human is at distance h_j depends on the three distributions P_u, P_m, and P_h. In order to correctly model the combined distribution, we must consider the dependencies between these distributions. As already done in the derivation of the sensor model $P(d_i \mid l)$, we can combine the different cases by an exclusive-or relation: The sensor measures a distance d_i if

a.) the sensor beam is

1.) reflected **neither** by an unknown obstacle **nor** by the human before reaching distance d_i

$$a_1 = 1 - \sum_{j<i}(1 - (1 - P_u(d_j))(1 - P_h(d_j \mid h_k))),$$

2.) **and** reflected by the known obstacle at distance d_i

$$a_2 = c_d \ P_m(d_i \mid l)$$

b.) **or** the beam is

1.) reflected **neither** by an unknown obstacle **nor** by the known obstacle **nor** by the human before reaching distance d_i

$$b_1 = 1 - \sum_{j<i} P(d_j \mid l)$$

2.) **and** reflected by an unknown obstacle at distance d_i

$$b_2 = c_r$$

c.) **or** the beam is

1.) reflected **neither** by an unknown obstacle **nor** by the map obstacle before reaching distance d_i

$$c_1 = 1 - \sum_{j<i}(1 - (1 - P_u(d_j))(1 - c_d \ P_m(d_j \mid l))),$$

2.) **and** reflected by the human at distance d_i

$$c_2 = c_h P_h(d_i \mid h_k).$$

By indexing the cases according to this enumeration, we get Eq. (3.9) which can be transformed via Eq. (3.10) into Eq. (3.11).

$$\begin{aligned}
& P(d_i \mid h_j, l) \\
= \ & P(a_1 \wedge a_2 \ \vee \ b_1 \wedge b_2 \ \vee \ c_1 \wedge c_2) && (3.9) \\
= \ & 1 - [1 - P(a_1 a_2)] \cdot [1 - P(b_1 b_2)] \cdot [1 - P(c_1 c_2)] && (3.10) \\
= \ & 1 - \left[1 - \left(1 - \sum_{k<i}(1 - (1 - P_u(d_k))(1 - c_h P_h(d_k \mid h_j)))\right) \cdot c_d \ P_m(d_i \mid l)\right] \cdot \\
& \left[1 - \left(1 - \sum_{k<i} P(d_k \mid l)\right) \cdot c_r\right] \cdot \\
& \left[1 - \left(1 - \sum_{k<i}(1 - (1 - P_u(d_k))(1 - P_m(d_k \mid l)))\right) \cdot c_h P_h(d_i \mid h_k)\right] && (3.11)
\end{aligned}$$

It is worth noting that this extended model of proximity sensors has only *one* additional parameter compared to the original model of proximity sensors. This parameter c_h

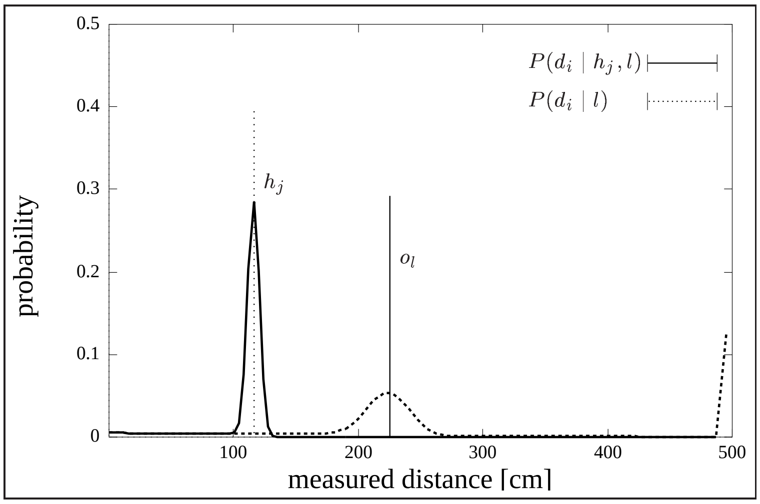


Fig. 3.5. Probability $P(d_i \mid l, h_j)$ of measuring distance d_i if the human (110cm) is in front of the map obstacle (230cm).

represents the probability that a human is detected by the sensor if it stands in the sensing direction (we set this value to 0.99 in our work).

Two examples of the extended sensor model defined in Eq. (3.11) are depicted in Figure 3.5 and Figure 3.6 as solid lines. Here, the graphs represent the probability $P(d_i \mid h_j, l)$ for a fixed position of the robot. In the first figure, the human stands in front of the next obstacle in the map and in Figure 3.6 the human is behind the map obstacle. As can be

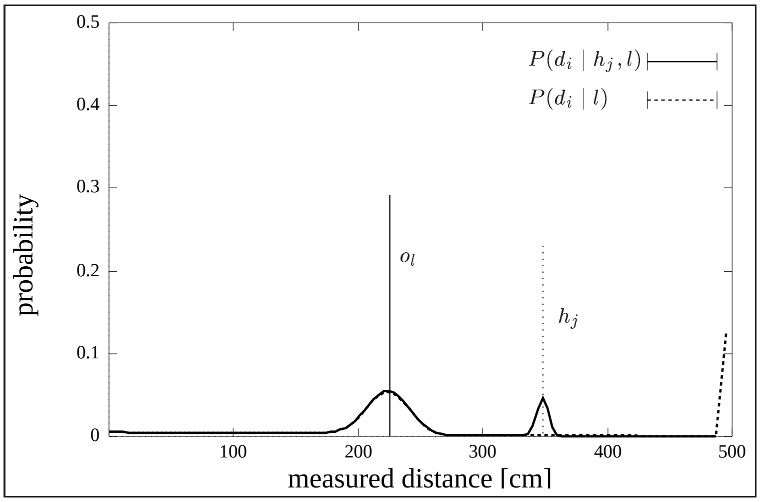


Fig. 3.6. Probability $P(d_i \mid l, h_j)$ of measuring distance d_i if the human (350cm) is behind of the map obstacle (230cm).

seen in Figure 3.5, the human in front of the map obstacle almost certainly reflects the sensor beam and longer measurements have proabilities close to zero. However, if the human is behind the map obstacle, then the beam is most likely reflected by the map obstacle but there remains a slight probability that the beam reaches the human (see Figure 3.6). This difference is based on the high probability $c_h = 0.99$ that humans are detected by the sensor. Obstacles in the map are only detected with probability c_d, for which we extracted a value of 0.75 from real data measured in the museum environment. To illustrate how the presence of a human influences the probability of measuring a certain distance we additionally plotted in the figures the probabilities *without* considering the human (dashed lines). As expected, the human behind the map obstacle does not affect the detection of the map obstacle. Thus, both graphs in Figure 3.6 are identical until the sensor beam reaches the human.

Examples for Estimating the Distance to Humans

Now we have derived a probabilistic description for the outcome of a proximity sensor given that we know the distance to a human and the position of the robot. With this extended model for proximity sensors we are ready to compute the probability for the presence of humans. $P(h_j \mid l, d_i)$, namely the probability that a human is in distance h_j if the robot is at location l and measures distance d_i is defined by the following equation (compare Eq. (3.7)).

$$P(h_j \mid d_i, l) \quad = \quad \frac{P(d_i \mid h_j, l)\ P(h_j \mid l)}{P(d_i \mid l)} \tag{3.12}$$

Three examples of this probability distribution are plotted in Figures 3.7 - 3.9. In all three cases, the distance to the next obstacle in the map is 230 cm.

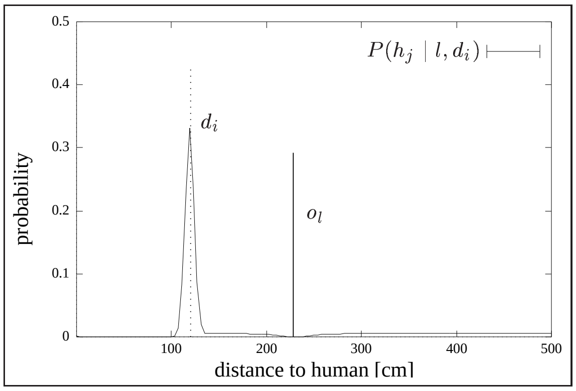


Fig. 3.7. Probability $P(h_j \mid l, d_i)$ that a human is at distance h_j if the robot measures a distance d_i of 120cm and the next obstacle in the map is at a distance of 230cm.

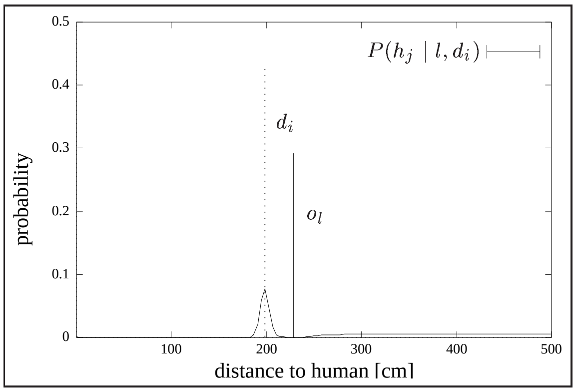


Fig. 3.8. Probability that a human is at distance h_j if robot measures a distance d_i of 200cm.

As expected, the probability for the detection of a human is high, if the measured distance differs strongly from the distance expected from the map (Figures 3.7 and 3.9). However, if the measured distance is close to the distance to the map obstacle, then it is very likely that the sensor beam has been reflected by the obstacle in the map. Accordingly, the probability for the presence of a human is considerably low (Figure 3.8).

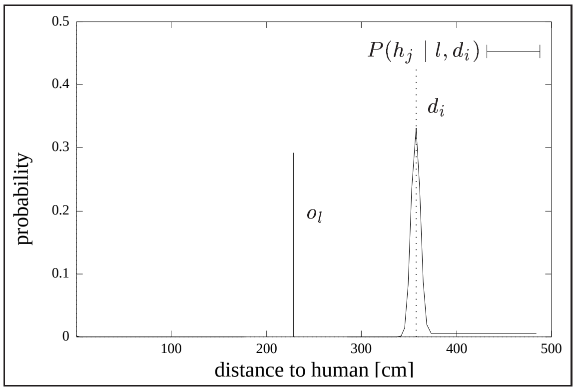


Fig. 3.9. Probability that a human is at distance h_j if the robot measures a distance d_i of 360cm.

All three examples share another interesting aspect of the distribution. While the probability that a human is behind the measured distance is not affected by the sensor reading, the probability that a human is closer than the measured distance is almost zero because otherwise the sensor beam would not have reached the measured distance d_i.

While this extended sensor model can be used to estimate the positions of humans within the environment, we want to apply this model to the *blockage filter* for detecting sensor readings corrupted by such unmodeled events.

The Resulting Blockage Filter

Given our model for the detection of humans we are prepared to compute the *blockage filter* according to Eq. (3.6). Figure 3.10 depicts an example distribution of the probability that a certain measurement is caused by a human blocking the closest obstacle in the map. This distribution is defined by:

$$P_{\text{blocked}}(d_i \mid l) \;=\; \sum_{j<o_l} P(h_j \mid d_i, l) \tag{3.13}$$

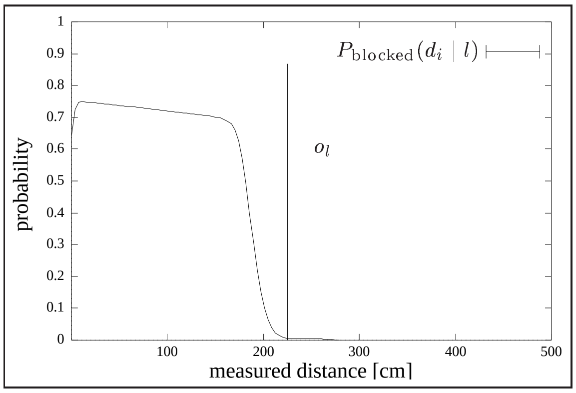


Fig. 3.10. Probability $P_{\text{blocked}}(d_i \mid l)$ of the obstacle at distance o_l being blocked by a human when the measured distance is d_i.

In this example the position of the robot is assumed to be known and the distance resulting to the next obstacle in the map is 230 cm.

To summarize, we introduced in this section two different filters especially designed for detecting proximity sensor measurements corrupted by unmodeled obstacles. The first filter, called *distance filter*, is based on the assumption that the shortness of sensor readings directly corresponds to the probability that they are caused by unmodeled obstacles. The *blockage filter* is based on a probabilistically correct model of the situation we want to detect. For this purpose, we first extended our sensor model introduced in Section 2.6.1 in order to consider knowledge about the presence of humans. Based on this extended sensor model, it is straightforward to compute the probability that a certain sensor measurement is caused by a human in front of a modeled obstacle. As expected, the resulting filters are very similar (compare Figure 3.3 and Figure 3.10).

The next section is concerned with empirical evaluations illustrating the consequences of the filters on the performance of the Markov localization procedure. They are compared with respect to (a) their ability of keeping track of the robot's position in densely populated environments and (b) their ability to recover from serious estimation failures.

3.6 Experimental Results

Localization using the entropy filter was a central component of the tour-guide robot deployed in the Deutsches Museum Bonn. Accurate and reliable localization was crucial for this project, as many of the obstacles were "invisible" to the robot's sensors (e.g. glass cages, metal bars, staircases, and the alike). Only through accurate localization could collisions with those obstacles be avoided (see Section 5.4). On the other hand, the robot often was circumvented by dozens of people, i.e., the environment was highly dynamic. Using Markov localization with entropy filters, our approach led only to a single software-related collision, which involved an "invisible" obstacle and which was caused by a localization error that was slightly larger than a 30cm safety margin. In this application, only the entropy filter was used. The two proximity filters were developed post facto, based on an analysis of the collision reported above, in an attempt to prevent similar effects in future installations. In the experiments described in this section we only report the evaluations of the distance filter, because we could not find any significant difference between the distance filter and the blockage filter[2]. Thus all conclusions we draw for the distance filter can be transfered to the blockage filter.

The evidence from the museum is purely anecdotal. Based on sensor data collected during the museum tour-guide project, we also investigated the merit of our filter techniques more systematically, and under even more extreme conditions. In particular, we were interested in the localization results ...

1. ... when the environment is densely populated (more than 50% of the sensor reading are corrupted) and

2. ... when the robot suffers extreme dead-reckoning errors (induced by a person carrying the robot somewhere else).

3.6.1 Datasets

Two datasets were used in our comparison, both of which were recorded in the museum, and which mainly differed by the amount of disturbances.

1. The first dataset was collected during 2.0 hours of robot motion, in which the robot traversed as much as 1,000 meters. The corresponding path of the robot is shown in Figure 3.11 (a). This data set was collected when the museum was closed, and the robot guided only remote internet-visitors through the museum. The robot's top speed was as low as 50cm/sec (to make the robot look more realistic for Web users).

[2]The blockage filter with the threshold θ set to 0.5 filters almost the same readings as the distance filter with the threshold θ set to 0.99.

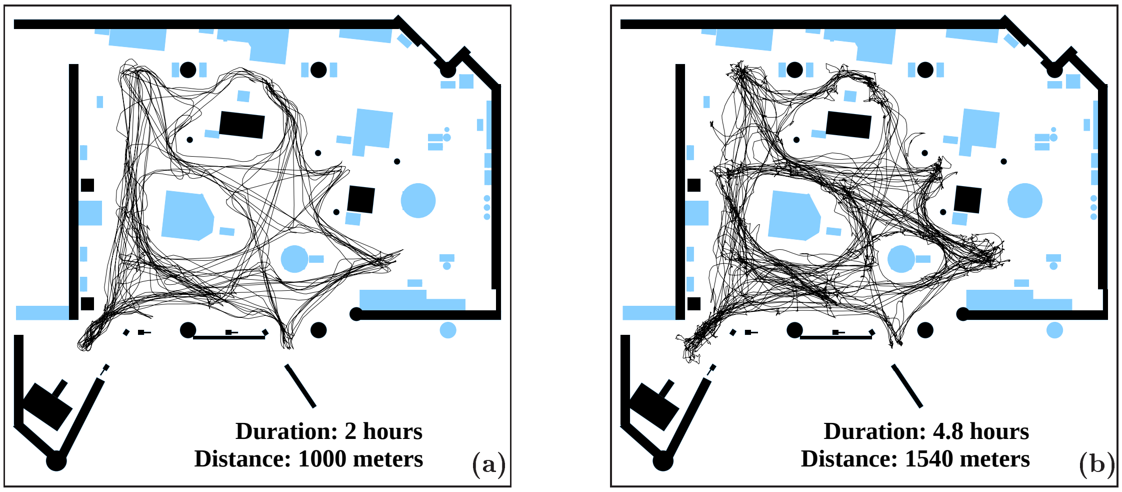


Fig. 3.11. Paths of the datasets obtained when the museum was (a) closed and (b) during peak traffic hours.

Thus, this dataset was "ideal" in that the environment was only sparsely populated, and the robot moved slowly.

2. Figure 3.11 (b) shows the path of the second dataset, which represents 4.8 hours, or 1,540 meters of robot motion through dense crowds. This dataset was collected during peak traffic hours on the most crowded day during the entire exhibition. When collecting this data, the robot was frequently faced with situations as illustrated in Figure 3.1 and Figure 3.2. The top speed in this dataset was 80cm/sec.

Both datasets consist of logs of odometry and laser-range finder scans collected while the robot moved through the museum. Using the time stamps in the logs, all tests have been conducted in real-time simulation on a SUN-Ultra-Sparc 1 (177-MHz). The first dataset contained more than 32,000, and the second dataset more than 73,000 laser scans. The reader may notice that only the obstacles shown in black in Figure 3.11 were actually used for localization; the others were either invisible, or could not be detected reliably.

Figure 3.12 shows the estimated percentage of corrupted sensor readings over time for both datasets. The dashed line corresponds to the first data set while the solid line illustrates the corruption of the second (longer) data set. In the second dataset, more than half of all measurements were corrupted for extended durations of time. These numbers are estimates only; they were obtained by analyzing each laser reading as to whether it was significantly shorter than the distance to the next obstacle.

To evaluate the different localization methods quantitatively, we faced the problem of determining the *true* locations of the robot at different points in time. Unfortunately, the environment and the sheer size of the datasets prohibited accurate measurements of these locations for all 105,000 sensor measurements, especially when the robot was in

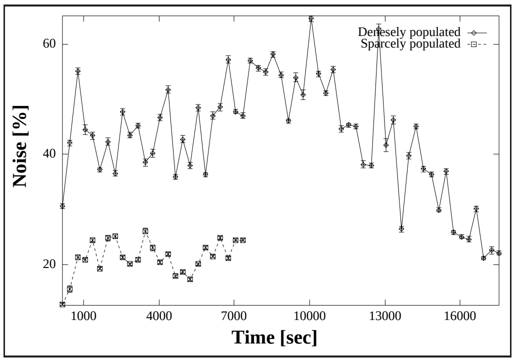


Fig. 3.12. Percentage of noisy sensor measurements averaged over time intervals of five minutes.

motion. In order to estimate the *accuracy* of the methods in the first experiment, we selected 118 representative reference positions from the second dataset when the robot was not in motion. For these positions we manually determined the robot's location as closely as possible through careful comparison of sensor scans, the robot's path, and the environment. For the other experiments we only needed to decide when the robot's position was lost. Because this decision did not need such highly accurate reference positions, we were able to generate *reference paths* for both datasets by averaging over the estimates of nine independent runs for each filter on the datasets (with small random disturbances). Visual inspection of the results made us believe that the resulting reference locations were indeed correct and accurate enough.

3.6.2 Tracking the Robot's Position

Robustness Against Corrupted Measurements

In our first series of experiments, we were interested in comparing the ability of all three approaches —plain Markov localization, localization with the entropy filter, and localization with the distance filter— to keep track of the robot's position under normal working conditions.

All three approaches worked nicely for tracking the robot's position in the empty museum (first dataset), exhibiting only negligible errors in localization. The results obtained for the second, more challenging dataset, however, were quite different. In a nutshell, both filter-based approaches tracked the robot's position accurately, whereas conventional Markov localization failed frequently and, thus, had it been used in the museum exhibit, would inevitably have led to a large number of collisions with obstacles.

Table 3.1 summarizes the results obtained for the different approaches at stake. In this and the following tables, we shaded unsatisfying results in light-grey. The first row of Table 3.1 provides the percentage of failures for the different filters on the first dataset

Filter	None		Entropy		Distance	
$\text{failures}_{\text{I}}$ [%]	1.6	± 0.4	0.9	± 0.4	0.0	± 0.0
$\text{failures}_{\text{II}}$ [%]	26.8	± 2.4	1.1	± 0.3	1.2	± 0.7

Table 3.1: Ability to track the robot's position.

(error values represent 95% confidence intervals). Position estimates were considered a "failure," if the estimated location deviated from the reference path by more than 45 cm for at least 20 seconds. The percentage is measured in time during which the position was lost against total time of the dataset. As can be seen here, all three approaches work well, and the distance filter provides the best performance. The second row provides the failures on the second dataset. While plain Markov localization failed in 26.8% of the overall time, both filter techniques show almost equal results: in both cases, the failure rate was around 1%. These results confirm the conjecture that, while Markov localization is prone to fail in highly dynamic environments, the two filter techniques are robust against such corruptions.

Accuracy in Densely Populated Environment

In order to evaluate the accuracy of the different approaches we computed the average Euclidean errors between the estimated positions at the 118 reference positions of the second dataset. These values are provided in Table 3.2. The first row, labeled $\overline{d}$, gives this average distance from the reference positions. Here the gap between conventional Markov localization and our approaches is large. The second row, labeled $\overline{d}'$, shows the same error averaged only for those positions that are *not* considered a failure. The reader may notice that the accuracy of the filter techniques is higher than the grid resolution of 15cm.

Filter	None		Entropy		Distance	
$\overline{d}$ [cm]	188.9	± 26.9	9.2	± 0.5	11.4	± 3.4
$\overline{d}'$ [cm]	22.8	± 0.9	8.9	± 0.4	8.5	± 0.4

Table 3.2: Accuracy at reference positions in dataset II.

Analysis of Sample Data

To shed light onto the question as to why Markov localization performs so poorly when compared to the filter algorithms, we analyzed the sensor readings that each method considered during localization. Obviously, plain Markov localization considers all sensor readings.[3] The other filters choose reading selectively, in an attempt to filter out readings that are corrupted by noise.

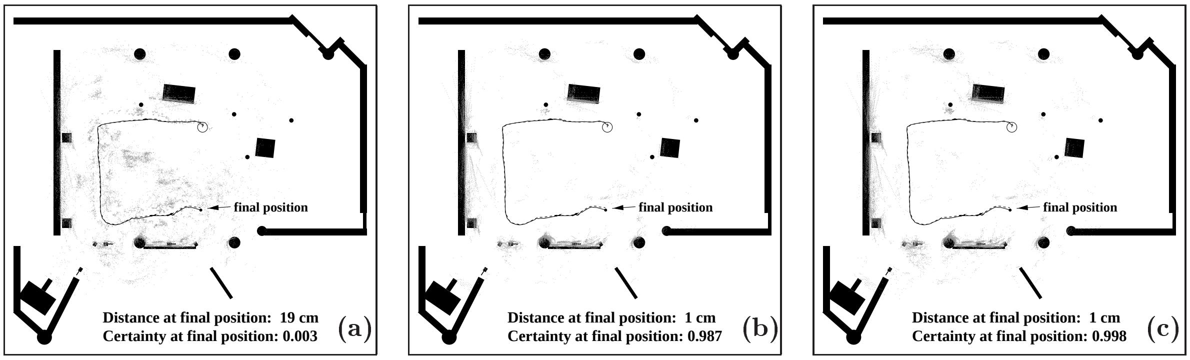


Fig. 3.13. Estimated and real paths of the robot along with endpoints of integrated sensor measurements using (a) no filter, (b) entropy filter, and (c) distance filter.

Figure 3.13 shows, for a a small fraction of the data, the measurements considered by either of the methods. Each figure plots only those measurements that were incorporated into the robot's belief. The points have been generated by storing the sensor measurements considered for localization during each run and plotting their end points *relative to the positions on the reference path*. The figures illustrate that both filter approaches manage to focus their attention on the "right" sensor measurements, whereas conventional Markov localization incorporates massive amounts of corrupted (misleading) measurements. Moreover, both filters show similar behavior. As also can be seen in Figure 3.13, both filter-based approaches produce more accurate results and higher certainty in the correct position for this example. In fact, the plain Markov localization failed to keep track of the robot's position shortly after the end of this example. These results demonstrate that our approaches scale much better to populated and dynamic environments than Markov localization.

3.6.3 Recovery from Extreme Localization Failures

A key advantage of the original Markov localization technique lies in its ability to *recover* from extreme localization failures. Relocalization after a failure is often more difficult than global localization from scratch, since the robot has to

1. detect that its current belief does not reflect its real location and

2. globally re-establish its state afterwards.

Since the filter-based approaches incorporate sensor data selectively, it is not clear that they still maintain the ability to recover from global localization failure.

[3] As noted in Section 2.6.2, our implementation of Markov localization considers only a random subset, to localize the robot in real-time.

Our experiments under normal operation conditions did not lead to such failures for the two filter techniques; thus, we manually introduced such failures into the data to test the robustness of these methods in the extreme. More specifically, in our experiments we "tele-ported" the robot at random points in time to other locations. Technically, this was done by changing the robot's orientation by 180±90 degree and shifting it by 0±100cm, without letting the robot know. These perturbations were introduced randomly, with a probability of 0.005 per meter of robot motion. Obviously, such incidents make the robot lose its position.

Each method was tested on 20 differently corrupted versions of both datasets. This resulted in a total of more than 50 position failures in each dataset. For each of these failures we measured the time until the methods relocalized the robot correctly. Relocalization was assumed to have succeeded if the distance between the estimated position and the reference path was smaller than 45cm for more than 10 seconds.

Filter	None		Entropy		Distance	
Dataset I						
$\overline{t_{\text{rec}}}$ [sec]	237	± 27	1779	± 548	188	± 30
failures [%]	10.2	± 1.8	45.6	± 7.1	6.8	± 1.6
Dataset II						
$\overline{t_{\text{rec}}}$ [sec]	269	± 60	1310	± 904	235	± 46
failures [%]	39.5	± 5.1	72.8	± 7.3	7.8	± 1.9

Table 3.3: Summary of recovery experiments.

Table 3.3 provides relocalization results for the various methods and for the two different datasets. Here $\overline{t_{\text{rec}}}$ represents the average time in seconds needed to recover from a situation when the position was lost due to a manually introduced perturbation. The results are remarkably different from the results obtained under normal operational conditions. Both conventional Markov localization and the extension using distance filters are relatively efficient in recovering from extreme positioning errors in the first dataset, whereas the entropy filter-based approach is an order of magnitude less efficient (see first row in Table 3.3). The unsatisfactory performance of the entropy filter in this experiment is due to the fact that it always tries to confirm the current belief of the robot, which is wrong in the case of our manually introduced perturbations (see the example below). The percentage of failures confirms this result: Again, the distance filter is slightly better than the approach without filter, while the entropy filter performs poorly in such situations. The results for the average time to recover from failures on the second dataset are similar to those in the first dataset. The bottom row provides the percentage of failures. Please note, that these values include localization failures due to manually introduced perturbations *and* localization failures due to the corrupted sensor measurements in this dataset. Here, the distance filter-based approach performs significantly better than both other approaches.

Comparison of Entropy and Distance Filter

Figure 3.14 illustrates the main features of the entropy and the distance filter. It shows two different situations where RHINO has been placed into the museum and receives a fixed scan. While the estimated position corresponds to the true location of the robot in the first case (a) and (c), the robot has been rotated by 30° relative to its real position in the second case (b) and (d).

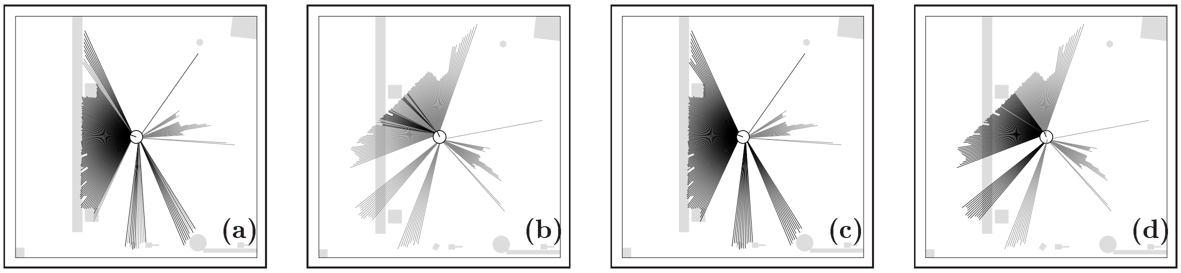


Fig. 3.14. Sensor Measurements selected using entropy filter (a, b) and using distance filter (c, d).

Obviously, some of the beams correspond to static obstacles that are part of the map, whereas others are caused by humans. As in the previous figures, maximum range measurements are not shown for the sake of clarity. The shading of the beams indicates the results of the two filters. The black lines correspond to selected beams whereas grey lines are the beams which are filtered out. In the first situation in which the position of the robot is estimated correctly both filters show similar performance: They successfully filter out all measurements reflected by objects not contained in the map (which are humans in this case). In the second situation in which the robot is rotated so that the belief state $Bel(L)$ does not reflect the true state of the robot, the entropy filter selects only those beams that are close to their expected distance, while the distance also chooses those sensor measurements which are longer than expected.

Thus, the difference between both filters can be characterized as follows. The entropy filter always tries to confirm the current belief of the robot (whether this is right or wrong) while the distance filter also allows the incorporation of very unlikely sensor measurements (see Figure 3.14(d)).

3.6.4 Summary of Experimental Results in the Museum

The results of the different experiments conducted in the museum are summarized in the two charts below. The charts represent the average percentage of failures during the different experiments (confidence intervals are omitted). While the first chart comprises the localization failures made on the authentic datasets (tracking experiments), the second chart shows the failures on the manually corrupted datasets (recovery experiments).

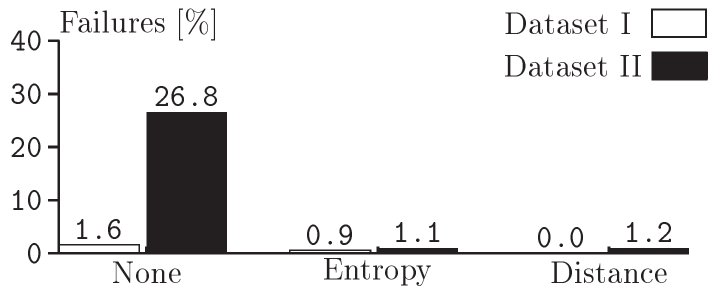


Chart 1. Failures when applying the different filters in the **tracking** experiment.

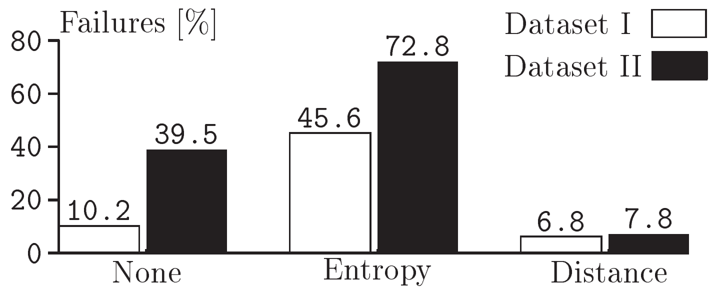


Chart 2. Failures when applying the different filters in the **recovery** experiment.

The key features of the different approaches tested here can be stated as follows.

Markov localization *without* filtering is

- robust against extreme localization failures if the environment is not highly dynamic (see Chart 2) *but*
- scales poorly if the sensor data is highly corrupted (see Dataset II in both charts).

Markov localization extended with the *entropy filter* is

- able to track the robot's position even in highly dynamic environments (see Chart 1) *but*
- lacks the ability to recover from extreme localization failures (see Chart 2).

Markov localization extended with the *distance filter* [4] is

- is able to robustly track the robot's position even in highly dynamic environments (see Chart 1) *and*
- is robust against extreme localization failures even in highly dynamic environments (see Chart 2).

These results are encouraging. They illustrate that despite the fact that sensor readings are processed selectively, the distance filter-based approach recovers more efficiently from extreme localization errors than the conventional Markov approach. These findings are specifically interesting in the light of the fact that in the Deutsches Museum the entropy-based filter was used, which, according to these results, would have led to poor recovery from extreme failures.

3.6.5 Localization in Dynamic Office Environments

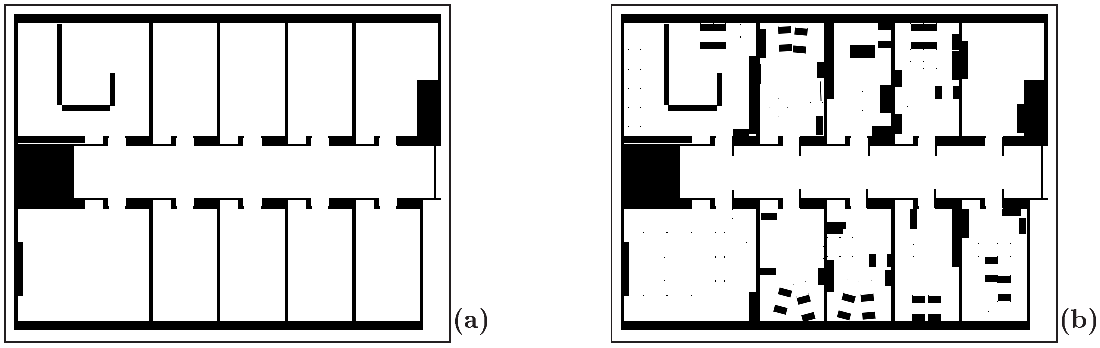


Fig. 3.15. (a) Outline used for localization and (b) environment including furniture detectable by the laser range-finders.

We also conducted experiments in our office environment in order to test the ability of the distance filter technique to deal with the specific dynamics that typically occur in such environments. Here, the dynamics are mainly caused by refurnishings. We tested our technique in a kind of worst case application: The model of the environment only contained

[4]It should be emphasized that experiments conducted with the blockage filter resulted in the same localization performance as the one observed for the distance filter.

an outline of the offices (see Figure 3.15 (a)). A map of the environment including the doors and the furniture that can be detected by the robot's laser range-finders is depicted in Figure 3.15 (b).

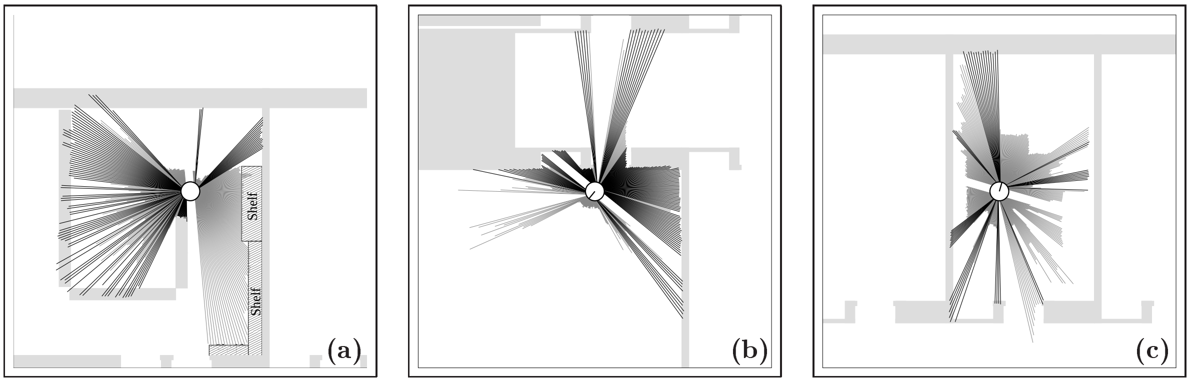


Fig. 3.16. Examples of filtered (grey) and incorporated (black) sensor readings in different locations of the environment.

Example scans of the laser range-finders taken at different locations in the environment are shown in Figure 3.16. Sensor readings filtered out are plotted in grey. As can be seen in the plots, sensor measurements reflected by the furniture are successfully filtered out as the robot moves through the environment. It should be noted that in rare cases, our approach fails to correctly filter sensor measurements. This can happen if e.g. the major part of a wall is covered by a small shelf: If the robot passes such a shelf and is not very certain about its position, it might confuse this shelf with a wall and thus fail to filter the corresponding sensor readings. A scan taken close to such a shelf is plotted in Figure 3.16 (a). In this figure, the readings are correctly filtered out, but in the worst case we observed that the robot confused the shelf with the wall and thus the estimated position was off the real position by up to 30cm. Fortunately, the localization quickly recovers from such failures as the robot moves on.

In addition to these examples we conducted more systematic experiments to determine the accuracy of the position estimation using our distance filter technique based on the incomplete map shown in Figure 3.15 (a). Therefore, we collected sensor data as the robot moved through the office environment and determined the exact positions of 22 reference points on the path of the robot (see Figure 3.17 (a)). The average distances between the estimated locations and the reference positions using the distance filter method are plotted in Figure 3.17 (b). These distances were averaged over 8 real-time simulations based on the time stamps in the data[5]. Again, the error-bars provide 95% confidence intervals. As can be seen in this plot, the localization error lies within reasonable bounds at all reference positions.

[5]By introducing small noise on the odometry data we increased the probability that different laser scans

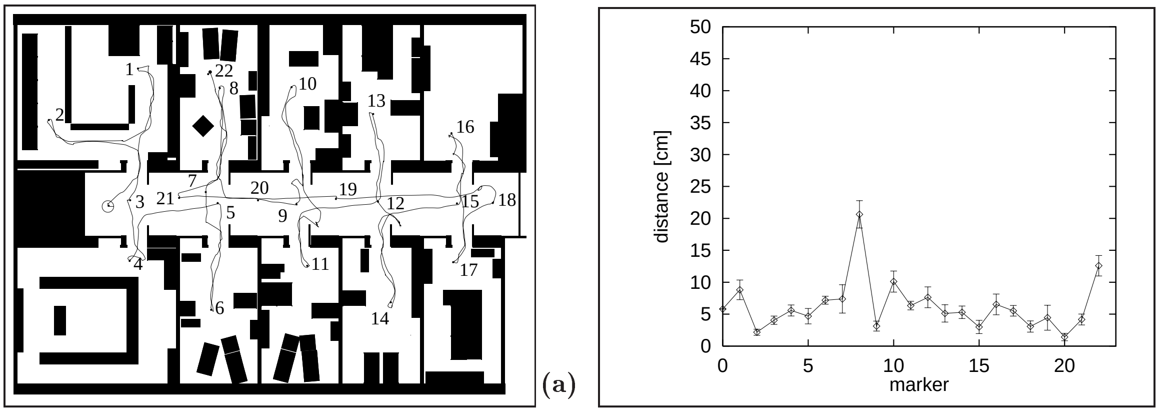


Fig. 3.17. (a) Map of the environment and path of the robot including reference positions. (b) Localization error at the reference positions when using only an outline of the environment.

For comparison, we additionally evaluated the dataset using the original Markov localization without filtering sensor measurements. In this case, position estimation was based on (a) only the outline and (b) the complete map of the environment (compare Figure 3.15 (a) and (b)). The average distances obtained with these two world models are depicted in Figure 3.18. As can be seen there, the position estimation without filtering is significantly less accurate than our filter technique when relying only on an outline of the environment (compare Figure 3.17 (b) and Figure 3.18 (a)). As expected, the results obtained with the complete map (see Figure 3.18 (b)) confirm the high accuracy of our grid based localization.

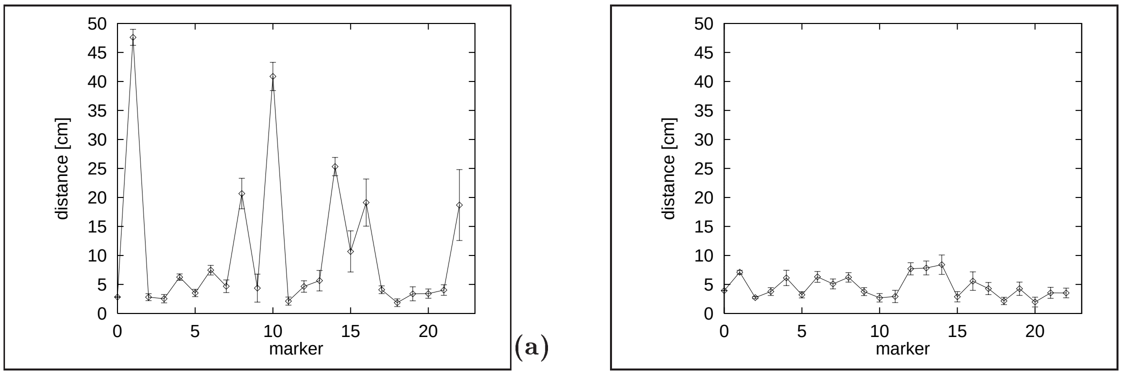


Fig. 3.18. Localization error at the reference positions without filtering when relying on (a) an outline and (b) the complete map of the environment.

A comparison of Figure 3.17 (b) and Figure 3.18 (b) shows that our filter technique based on an outline of the environment is not significantly less accurate than the original localization based on the complete map. The only significant deviations are at markers number 8 and 22, both in the same room. A post the fact analysis of the laser scans taken

were used during the different evaluations.

at these two markers indicates that the position of the markers has been measured relative to the *furniture* in this room. Unfortunately, this position did not exactly conform with the marker position relative to the *walls*. Thus, position estimation based on the complete map is inherently closer to these reference positions than position estimation based only on an outline of the environment. To summarize, this experiment supports our belief that our grid based localization in combination with the distance filter technique allows us to accurately localize mobile robots even when relying only on an outline of the environment.

3.7 Discussion

In this chapter we proposed an extension of Markov localization that has been demonstrated to reliably localize mobile robots even in extreme situations: global localization in dynamic environments. These environments are characterized by the presence of various dynamic effects, such as crowds of people that might block the robot's sensors. Our approach filters sensor data, so that the damaging effect of corrupted data is reduced. Three specific filters were proposed and evaluated, one of which considers conditional entropy for selecting sensor readings, and two of which take into account additional knowledge about the effects of possible environment dynamics.

The entropy filter was essential for successfully operating a mobile robot in a crowded museum. Experimental comparisons using data collected there demonstrated that by processing sensor readings selectively, the filters especially designed for proximity sensors are able to localize robots in densely crowded environments, while retaining the ability to efficiently recover from global failures in localization. Furthermore, the tests conducted in an office environment suggest that our extension of Markov localization can accurately estimate the position of a robot even if only an outline of such an environment is used as a world model. These experiments demonstrate the robustness of our approach to changes expected in typical office environments. Further evaluation of the specific characteristics and problems of such an application is beyond the scope of this work and is a topic of future research.

We believe that the results presented here are essential for operating mobile robots in highly dynamic and densely populated environments. How specific are these results to the problem of mobile robot localization? Especially the extended sensor model applied for the blockage filter can be used to additionally estimate the location of people in the environment as the robot performs its tasks. Of course, such an application would be most effective if the world model contained all static obstacles. In fact, we applied the distance filter to indicate the presence of people during the museum tour-guide project. In conjunction with a measure of progress this filter was applied to detect when the robot's path was blocked by visitors. In such a situation the robot blew a horn to indicate its intention to move on. This turned out to be an important feature for the interaction between the robot and the people (see also [BCF+98a]).

We also believe that the first filter proposed here, the entropy filter, is applicable to a much wider variety of state estimation problems in dynamic environments. Loosely speaking, this filter makes robot perception highly selective, in that only sensor readings are considered that confirm the robot's current belief[6]. This filter rests only on two assumptions: First, that the variable to be estimated is represented probabilistically, and second, that sensor readings can be sorted into two bins, one which only contains corrupted readings, and one that contains authentic (non-corrupted) measurements. A straightforward application of this filter is the detection and compensation of hard ware failures of sensors, a problem extensively addressed in Murphy's work [MH96]. Future work will aim to characterize quantitatively, to what extent this filter can make robots more robust to sensor failures.

[6]In order to make this filter more robust against estimation failures one could use randomized strategies such as filtering sensor readings probabilistically, where the likelihood of filtering out a sensor reading is proportional to the change in entropy.

Chapter 4

Active Localization

4.1 Introduction

In the previous chapters we introduced Markov localization as a robust technique for estimating the global position of a mobile robot within its environment. As well as the vast majority of existing approaches, Markov localization is *passive*. Passive localization exclusively addresses the estimation of location based on an incoming stream of sensor data. It rests on the assumption that neither robot motion, nor the pointing direction of the robot's sensors can be controlled. While passive techniques have the obvious advantage that they do not interfere with the robot's primary tasks, they can be extremely inefficient especially during the process of global localization.

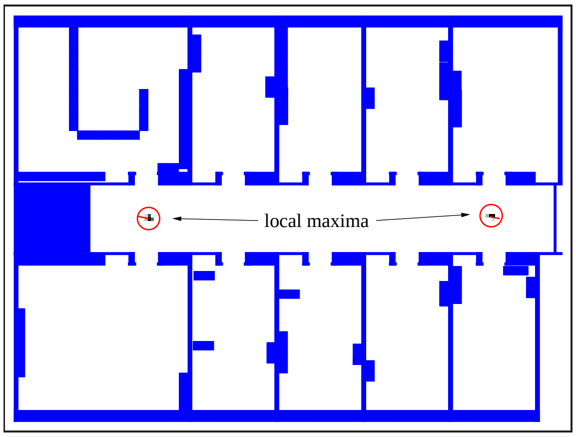


Fig. 4.1. Example situation in which passive localization is extremely inefficient.

Consider a situation that typically occurs during global localization in our department. The robot was placed in the corridor and Figure 4.1 depicts the belief state after random motion within this corridor. The two local maxima stem from the symmetry of the corridor and in such a situation the efficiency of localization can significantly be improved by guiding the robot into one of the offices. In symmetric office environments such as this one, random motion or perpetual wall following is often incapable of or at least inefficient at globally determining a robot's position.

Active localization goes beyond the passive paradigm. It assumes that during localization, the position estimation routine has partial or full control over the robot, providing it with a set of actions that can be used to disambiguate its belief. This gives the opportunity to increase the efficiency and the robustness of localization. Actively controlling the actuators of a robot in order to globally localize it pays off whenever the environment possesses relatively few features that enable it to unambiguously determine its location. For example, in many office environments, corridors and rooms often look alike for a mobile robot, hence a robot might have to move to a remote room in order to uniquely determine its location. Key open issues in active localization are *"where to move"* and *"where to look"* so as to best localize the robot.

In this chapter we will derive a decision-theoretic extension of Markov localization which provides rational criteria for determining where to move and where to point the robot's sensors. The guiding principle of our approach is to control the actuators so as to minimize future expected uncertainty. Uncertainty is measured by the entropy of future belief distributions. By choosing actions to minimize the expected future uncertainty, the approach is capable of actively localizing the robot.

In the next section we will discuss related work. After introducing the general framework for active localization in Section 4.3, we will apply this principle to active navigation and sensing in the following section. Key aspects of our efficient implementation of active localization are discussed in Section 4.5. After describing the experimental results we will discuss further research issues in Section 4.7.

4.2 Related Work

Active localization has received surprisingly little attention in the literature to date. In this section we discuss approaches that deal with this problem from three different points of view: First, place recognition during map building, second, theoretical bounds on the complexity of global localization, and, third, decision theoretic approaches to active localization.

Exploration during Map Building

The process of localization is closely related to the process of map building: in order to build consistent maps of its environment a mobile robot must know where it is during the construction of a map. Thus recent work in map building has concentrated on probabilistic methods for integrating map building and localization (see e.g. [SK97; TFB98; KS96]). However, most of these approaches do not consider the possibility of controlling the robot in order to enhance its localization performance.

Kuipers and Byun introduced a test procedure in order to check whether a certain place has already been visited during the process of map building or whether this place should be added as a new node to the topological map [KB91b]. They applied the so-called rehearsal procedure to actively disambiguate different possible locations. Engelson proposed several heuristic "opportunity scripts" to enhance the process of map building [Eng95]. These scripts include procedures which address the problem of localizing the robot within its map: e.g. the "retrace step" reduces position uncertainty by guiding the robot back to a previously visited location of which the position is known. Another script disambiguates different positions by performing actions with different results for the possible locations. However, both [KB91b] and [Eng95] do not provide an explicit probabilistic model of uncertainty in sensing and acting when performing active localization.

Results from Theoretical Computer Science

Work from algorithm theory developed efficient algorithms for localizing in graph structures. Kleinberg concentrates on the *complexity* of localizing a mobile robot within its environment [Kle94]. They address the problem of controlling a robot in order to most efficiently localize it globally in a known environment. Here, the environment is represented as a bounded-degree tree or as rectangles in the 2-dimensional plane. The performance of the algorithms is analyzed using the competitive ratio, which is the worst-case ratio between the distance traveled when using their on-line algorithm and the distance traveled based on the optimal solution. They propose a method with $O(n^{2/3})$-competitive ratio for trees with maximal n branch vertices.

Dudek et al. showed that in general, the problem of localizing a robot in a known environment by traveling a minimum distance is NP-hard [DRW98]. Here the environment is modeled as a simple polygon. They propose a strategy with competitive ratio $(k-1)$, where k is the number of locations to be disambiguated. They also show that this bound is the best possible ratio.

Both approaches concentrate on the problem of "hypothesis elimination ([DRW98])": Here the robot initially applies a simple motion strategy such as spiral search until the number of hypotheses about its location becomes small. Starting at this point, their approaches generate navigation actions yielding a unique position estimate. Both approaches are based on the assumption that the orientation of the robot is known and that there is no uncertainty in sensing and acting. Thus it remains unclear how these results scale towards real world applications.

Partially Observable Markov Decision Processes

The framework of *partially observable Markov decision processes* (short POMDPs) provides criteria for acting optimally under consideration of uncertainty in both acting and sensing. Controllers for POMDPs consist of the following two main components. The *state estimator* determines the state of the world based on incoming sensor data and on the actions performed by the agent. In addition, the *policy* maps states of the world into actions so as to maximize future rewards for the agent (see e.g. [KLC95] for an introduction to POMDPs). Applied to the mobile robot domain, the state of the world is given by the position of the robot and the policy maps such states to actions of the robot. The key problem of POMDPs can be stated as follows. What is the optimal policy in order to maximize future rewards? By taking position uncertainty into account when planning paths to target locations, POMDPs are *inherently* able to integrate actions for localization into such navigation plans. A key obstacle in applying POMDPs to active localization is their complexity. In the worst case, computing the complete evaluation function within the domain of POMDPs is exponential in space and time [PR95]. Several techniques have been introduced for efficiently generating approximate solutions to POMDPs (e.g. [Lov91; PR95; KLC95]). Unfortunately, even the most efficient techniques are only feasible for domains with less than hundred states and observations. Thus most applications in the robotic domain do not exploit the full potential of POMDPs for optimally controlling mobile robots but concentrate on the problem of localization as an application of state estimation (see also Section 2.3). Simmons and Koenig take the uncertainty in the position estimation into account when planning navigation actions, but they do not consider the ability of actively reducing the position uncertainty [SK95]. They assume that the robot will finally reach its goal when following the generated navigation plans. However, Kaelbling [KCK96] showed in an experimental study that this approach can lead to "cycles in the belief states and actions that do not make progress toward the goal".

One approach to use the framework of POMDPs in order to integrate active localization into robot control has been done in [KCK96]. Because of the complexity of solving POMDPs, [KCK96] propose and evaluate two heuristics for considering actions explicitly aiming at reducing uncertainty measured by the entropy of the position estimation. The first technique is called *action entropy* and is based on the following strategy. The robot ignores uncertainty in the position as long as the uncertainty with respect to the best navigation action is below a threshold. If this certainty about the best action becomes large, the robot is confused about what to do and chooses the next action by minimizing the expected entropy of the belief state. The second strategy is called *entropy-weighting* because the value of belief states used for generating navigation plans is given by a weighted sum. The first summand measures the utility of the belief with respect to increasing the certainty of the position estimation and the second summand measures the utility of the belief with respect to goal attainment. The weight is given by the entropy of the belief so that the robot tends to perform actions aiming at increasing the position certainty as it becomes more uncertain. While both strategies provide a good approach to combining actions for localization and actions for goal attainment, they still have an important drawback: The horizon for planning actions for position estimation is either set to a single action which is not able to carry the robot to remote locations or set to a fixed sequence of such local actions. To overcome the limited lookahead of this approach we have chosen to use more complex actions. Therefore, our approach is able to guide a robot to arbitrary remote locations while still keeping the computational complexity in reasonable bounds. Our technique for guiding a robot so as to efficiently localize it will be introduced in Section 4.4.1.

4.3 General Scheme for Action Selection

In decision theory, actions are selected by trading off their expected costs against their expected utility. The measure for costs and utility has to be defined for the problem at hand. In this section we will derive the description for computing the expected utility $U(a)$ and costs $C(a)$ of certain actions a for *position estimation*. Given these measures, the robot chooses the next action a^* according to the following rule.

$$a^* \quad = \quad \operatorname*{argmax}_a (U(a) - \beta \cdot C(a)). \tag{4.1}$$

Here $\beta \geq 0$ determines the relative importance of certainty versus costs. The choice of β depends on the application. In our experiments, β was set to 1.

Utility of Actions

During global localization, the utility of performing an action a is determined by the increase in the certainty of the position estimate upon executing a. We use the entropy $H(L)$ in order to measure the uncertainty of the belief $Bel(L)$. This measure has already been applied successfully in the context of localization problems in [KCK96] and in Section 3.4. Let $E_a[H(L_{t+1})]$ denote the expected entropy *after* having performed action a at time t and *after* having fired the sensors of the robot at time $t+1$. The utility $U_t(a)$ of performing an action a is measured by the decrease in uncertainty as given in Eq. (4.2). By averaging over all possible sensor measurements s, we obtain Eq. (4.3), where $P(s \mid a)$ is the probability of perceiving s after execution of action a.

$$U_t(a) \quad = \quad H(L_t) - E_a[H(L_{t+1})] \tag{4.2}$$

$$= \quad H(L_t) - \sum_s H(L_{t+1} \mid s, a) P(s \mid a) \tag{4.3}$$

$$= \quad H(L_t) + \sum_s \sum_l Bel(L_{t+1} = l \mid s, a) \ \log Bel(L_{t+1} = l \mid s, a) \ P(s \mid a) \tag{4.4}$$

$$= \; H(L_t) + \sum_s \sum_l P(s \mid l) \; Bel(L_{t+1} = l \mid a) \log \frac{P(s \mid l) Bel(L_{t+1} = l \mid a)}{P(s \mid a)} \tag{4.5}$$

Let $Bel(L_{t+1} = l \mid s, a)$ denote the belief of being at position l after having performed a and perceiving s; then expression (4.4) is obtained from the definition of the entropy given in Eq. (3.1). By applying the update equations (2.13) - (2.15) of the Markov localization algorithm we get Eq. (4.5) for computing the utility of an action a. Here, $P(s \mid a)$ is equivalent to the normalizer $P(s \mid L_t)$ introduced in Eq. (2.15) and can be computed from $Bel(L_{t+1})$.

In general, this principle can be applied to arbitrary kinds of actions. In order to compute the utility of actions one has to specify the following two factors.

1. $Bel(L_{t+1} = l \mid a)$, namely how the position of the robot represented by the belief changes upon executing action a, and

2. $P(s \mid l)$, namely the probability of perceiving a measurement s at a location l.

Costs of Actions

To find the best action we have to trade off the utility of each action against the cost of executing the action. Costs $C(a)$ can be given e.g. by the time needed to perform an action or by the amount of energy used up by the action. We will give an example for such costs in the next section.

4.4 Active Navigation and Active Sensing

In the previous section we have not specified what kind of actions we consider. In this section we will apply this principle to two kinds of actions, namely navigation and sensing.

4.4.1 Active Navigation

Active navigation addresses the problem of determining where to move so as to best position the robot. We use the term *active navigation* in order to emphasize its role within the context of active localization. At first glance, one might use simple motor control actions (such as "move 1m forward") as basic actions in active navigation. However, just looking at the immediate next motor command is often insufficient. For example, a robot might have to move to a remote room in order to uniquely determine its location, which might involve a long sequence of individual motor commands (c.F. 4.1).

In order to overcome the complexity of estimating the utility of a sequence of such simple actions we have chosen to consider more complex actions. In our approach, we regard arbitrary target points as atomic actions in active navigation. These target points are

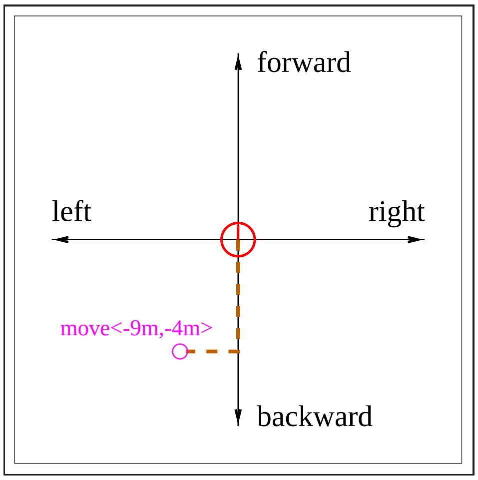


Fig. 4.2. Two-dimensional space of target actions.

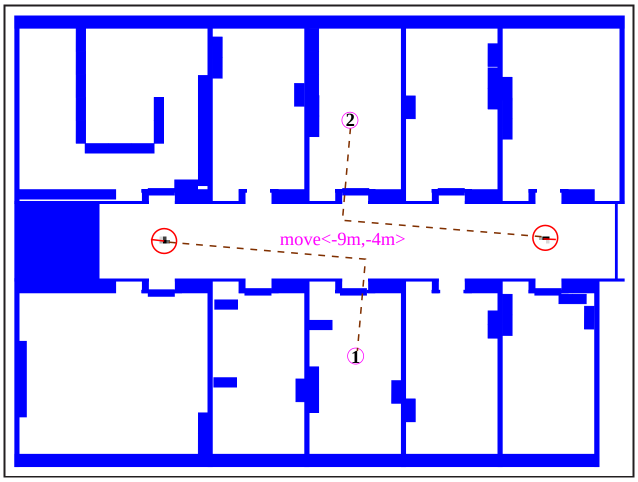


Fig. 4.3. Absolute positions of target points.

specified relative to the current robot location, not in absolute coordinates. For example, an action $a = move(-9\text{m}, -4\text{m})$ will make the robot move to a location 9 meters behind and 4 meters to the left, relative to its current location and heading direction (see Figure 4.2). Because targets are represented relative to the robot's position, the absolute position of such targets strongly depends on the belief of the position estimate. Figure 4.3 shows a situation where the belief is concentrated on the two positions marked by the big circles. The action $a = move(-9\text{m}, -4\text{m})$ might thus carry the robot to the location labeled "1" or to the location labeled "2".

More specifically, $f_a(l)$ denotes the coordinate transformation which expresses the real-world coordinates of the target location of action a, assuming that the robot is at l. Let a_f and a_s denote the forward and the side ward component of the movement action a, respectively. Then the three components of the target point when applying action a at location l are given by Eq. (4.6).

$$
\begin{aligned}
[f_a(l)]_x &= l_x + \cos l_\alpha a_f + \sin l_\alpha a_s \\
[f_a(l)]_y &= l_y + \sin l_\alpha a_f - \cos l_\alpha a_s \\
[f_a(l)]_\alpha &= l_\alpha + a_\alpha
\end{aligned}
\tag{4.6}
$$

The remainder of this section specifies the computation of the utility, the costs, and the cost-optimal path for navigation actions.

Utility of Navigation Actions

In order to determine the utility of moving to a relative target location using Eq. (4.5) we have to specify how the belief changes upon executing a navigation action a. To compute $Bel(L_{t+1} \mid a)$ we apply the inverse of the coordinate transformation in Eq. (4.6) to get the resulting belief:

$$Bel(L_{t+1} = l \mid a) \quad = \quad Bel(L_t = {f_a}^{-1}(l)). \tag{4.7}$$

Now we can write the utility of applying action a as given in Eq. (4.8). Here $P(s \mid l)$ is not limited to a certain kind of sensor and we will discuss the sensor model in Section 4.5.

$$U_t(a) = H(L_t) + \sum_s \sum_l P(s \mid l)\; Bel(L_t = {f_a}^{-1}(l)) \log \frac{P(s \mid l) Bel(L_t = {f_a}^{-1}(l))}{P(s \mid a)} \tag{4.8}$$

Cost of Navigation Actions

The costs of reaching a target point can differ substantially depending on the time-of-travel. To estimate $C(a)$ for an action a we estimate the expected costs on the cost-optimal path from the current location of the robot to the target location. To do so, our approach rests on the assumption that a map of the environment is available, which specifies which points l are occupied and which are not. In our implementation, the world model is given as an occupancy grid map.

Occupancy probabilities: Let $P_{occ}(l)$ denote the probability that location l is blocked by an obstacle. The robot has to compute the probability that a target point a is occupied. Recall that the robot does not know its exact location; thus, it must *estimate* the probability that a target point a is occupied. This probability will be denoted $P_{occ}(a)$. Geometric considerations permit the "translation" from $P_{occ}(l)$ (in real-world coordinates) to $P_{occ}(a)$ (in robot coordinates):

$$P_{occ}(a) \quad = \quad \sum_l Bel(L_t = l)\, P_{occ}(f_a(l)) \tag{4.9}$$

Again, $f_a(l)$ is the coordinate transformation introduced in Eq. (4.6). In essence, Eq. (4.9) translates, for any l, the point a into real-world coordinates $f_a(l)$, then considers the occupancy of this point ($P_{occ}(f_a(l))$). The expected occupancy is then obtained by averaging over all locations l, weighted by the robot's belief of actually being there $Bel(L_t = l)$. The result is the expected occupancy of a point a relative to the robot.

Costs and cost-optimal paths: Based on $P_{occ}(a)$, the expected path length and the cost-optimal policy can be obtained through *value iteration*, a popular version of dynamic programming (see e.g., [LDK95] for details). Value iteration assigns to each location a a *value* $v(a)$ that represents its expected distance to the robot. Initially, $v(a)$ is set to 0 for the location $a = (0, 0)$ (which is the robot's location, recall that

a is specified in relative coordinates), and ∞ for all other locations a. The value function $v(a)$ is then updated recursively according to the following rule.

$$v(a) \quad \longleftarrow \quad P_{occ}(a) + \min_{b}[v(b)] \tag{4.10}$$

Here b is minimized over all *neighbors* of a, i.e., all locations that can be reached from a with a single, atomic motor command [1]. (4.10) assumes that the costs for traversing a point a is proportional to the probability that a is occupied ($P_{occ}(a)$). Iteratively applying (4.10) leads to the cost function $C(a)$ for reaching any point a relative to the robot, and hill climbing in v (starting at a) gives the cost-optimal path from the robot's current position to any location a.

This completes the description of active navigation with the purpose of localization. To summarize, actions represent arbitrary target points relative to the robot's current position. Actions are selected by maximizing a weighted sum of (1) expected decrease in uncertainty (entropy) and (2) costs of moving there. In order to keep the computational complexity within reasonable bounds, the change of uncertainty is measured only at the target location. Costs are considered because they may vary drastically between different target points. In the next section we will apply our principle of action selection to the problem of active sensing.

4.4.2 Active Sensing

By active sensing we attack the problem of where to "point" the robot's sensors so as to best localize the robot. In this context active sensing is a special case of active navigation which is realized by pointing the sensor into the direction which maximizes the expected utility.

Utility of Sensing Actions

Let $a_\alpha = point(\alpha)$ denote the action of pointing the sensor into the direction α relative to the robot's orientation. The utility of such an action is given by

$$U_t(a_\alpha) = H(L_t) + \sum_{s_\alpha}\sum_{l} P(s_\alpha \mid l)\; Bel(L_t = l) \log \frac{P(s_\alpha \mid l) Bel(L_t = l)}{P(s_\alpha \mid a)}. \tag{4.11}$$

Here s_α are only those sensations perceivable in the corresponding direction α. Please note that we replaced $Bel(L_{t+1} = l \mid a_\alpha)$ in Eq. (4.5) by $Bel(L_t = l)$ because we assume that pointing a sensor in a specific direction does not change the position and thus the state of the robot.

[1] The reader may notice that this is a *deterministic* variant of value iteration, since actions are assumed to be deterministic.

Costs of Sensing Actions

In this work we assume that the costs for pointing a sensor does not depend on the direction. Thus we can neglect costs during active sensing and select the pointing direction solely based on the utility.

4.5 Efficient Implementation

In this section we discuss the different assumptions on which our efficient implementation of active localization is based. Furthermore we provide example situations in which our implementation will lead to sub-optimal solutions.

4.5.1 Restricted Sensing Model

The complexity of computing the utility of a single action a is in $O(|L| \cdot |S|)$, where L is the number of locations and $|S|$ is the number of possible sensor measurements (compare Eq. (4.8)). In this section we will discuss how to reduce the size of S, which in our approach consists of distances measured by proximity sensors such as sonar sensors or laser range-finders [2]. In order to take into account all the information that can be gathered at the target location when computing the utility of an action, it would be necessary to consider the situation when the robot fires *all* its proximity sensors at the target location. While this would result in the best estimate of the uncertainty after firing the robot's sensors, the number $|S|$ of possible measurements grows exponentially in the number of sensor beams to be considered (see Eq. (4.8)). The reader may notice that a scan of the proximity sensors of our robot consists of 24 ultrasound and 360 laser range-measurements. Thus it is definitely infeasible to consider all sensors. On the other hand considering only e.g. two fixed sensor beams would result in a very restricted estimation of the utility: Only features detectable in the corresponding two directions would be taken into account. In order to overcome this combinatorial explosion and still get a realistic estimate of the expected decrease in uncertainty, we assume that the robot fires *one* sensor chosen *randomly* at the target location. With this assumption we are able to average over the utilities for the different sensor beams, which results in a complexity *linear* in the number of considered sensors.

Example for Sub-optimal Estimation of Utilities

Our simplified sensing model results in loss of information when computing the utility of actions. Figure 4.4 (a) illustrates a situation in which this restricted model yields a

[2] In our implementation these measurements are discretized into 64 (ultrasound) or 128 (laser range-finder) different distances.

sub-optimal estimate of the utility of possible actions.

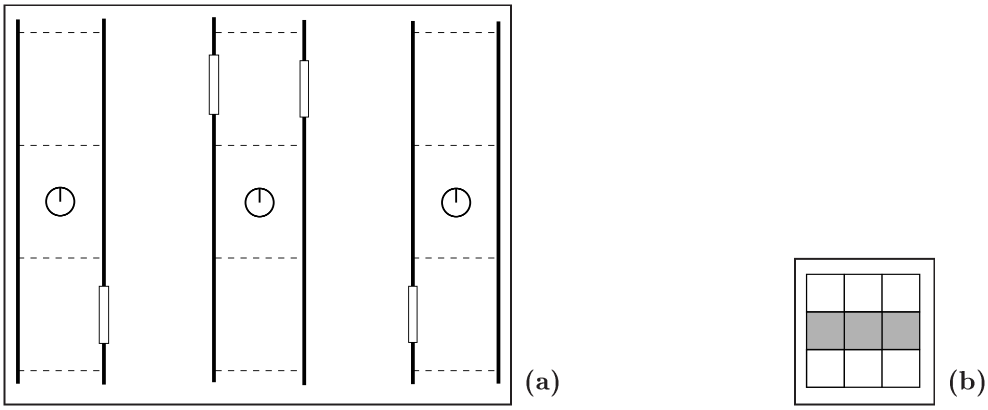


Fig. 4.4. (a) Three possible locations of the robot and (b) representation of the corresponding belief.

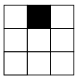
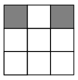
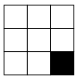
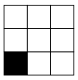
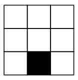
Here the robot is with equal likelihood of 1/3 in one of the three different hallways depicted Figure 4.4 (a). For simplicity, the figure only depicts these positions and their neighboring locations. The belief $Bel(L_t)$ for the resulting nine locations can be represented by the matrix in Figure 4.4 (b). In this simple example we only consider the following two actions: (1) move one step `Forward` and (2) move one step `Backward`. In order to further restrict the complexity of the example we assume that the robot has two perfect sensors: one which reliably detects doors and walls on its right side and one which reliably detects doors and walls on the robot's left side. Thus the set S of possible percepts in our example consists of the pairs <DL,DR>, <DL,WR>, <WL,DR>, and <WL,WR> of sensor measurements s, where e.g. <DL,WR> is the percept of simultaneously detecting a door on the left and a wall on the right side.

In this simple example it is most convenient to apply the following equation for computing the utility of the two actions (see also Eq. (4.4)).

$$U_t(a) \;=\; H(L_t) + \sum_s \sum_l Bel(L_{t+1} = l \mid s, a)\ \log Bel(L_{t+1} = l \mid s, a)\ P(s \mid a) \qquad (4.12)$$

To compute the sum in Eq. (4.12) we must consider all possible combinations of the four percepts s and the nine locations l for each of the two actions a. Table 4.1 includes only those combinations for which $P(s \mid a) \neq 0$. E.g. if the robot moves one step `Forward` it is not possible to observe <DL,WR>, namely a door on its left and a wall on its right (c.F. 4.4 (a)).

The fourth column in the table represents the belief $Bel(L_{t+1} \mid a, s)$ after performing action a and perceiving s. Given these *a posteriori* beliefs, it is clear which of the two actions should be preferred: While the percepts on the locations in front of the robot cannot disambiguate the three possible locations, each percept made after performing the

Action a	Percept s	$P(s \mid a)$	$Bel(L_{t+1} \mid a, s)$	$H(L_{t+1} \mid s, a)$	$U_t(a)$
`Forward`	<DL,DR>	1/3		0	0.91
`Forward`	<WL,WR>	2/3		1	
`Backward`	<DL,WR>	1/3		0	1.58
`Backward`	<WL,DR>	1/3		0	
`Backward`	<WL,WR>	1/3		0	

Tab. 4.1. Components of the utilities when considering both sensors simultaneously.

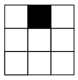
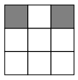
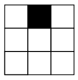
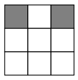
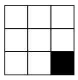
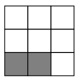
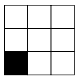
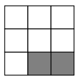
action `Backward` yields a unique determination of the robot's location (c.F. 4.4 (a)). This fact is also represented in the entropies $H(L_{t+1} \mid s, a)$. Thus when taking *all* sensors simultaneously into account, the resulting utilities $U_t(a)$ correctly reflect the preference of the action `Backward`.

When applying our restricted sensing model, the sensors are considered independently and the resulting percepts s are <DL>, <WL>, <DR>, and <WR>. It should be noted that only in this simple example the number $|S|$ of considered percepts is not reduced by using this sensing model (four percepts in both cases). The relevant values for computing the utility of the two actions are given in Table 4.2. By considering the percepts independently, it is not possible to detect that the location can be uniquely determined after performing the action `Backward` and firing the robot's sensors (see the beliefs in the fourth column). Thus even though the real utility of the two actions is significantly different, our sensing model assigns the same utility to both actions.

In this example we showed that when considering the sensors of the robot independently, the utility of actions is sometimes *underestimated*.

4.5.2 Actions

As noted above, the complexity of computing the utility of an action is in $O(|L| \cdot |S|)$. In our implementation of active navigation the discretization of the action space as shown in Figure 4.2 is as small as the resolution of the applied position probability grid. This results in an overall complexity of $O(|L|^2 \cdot |S|)$ for computing the next best action a^*. It should be noted that this complexity grows exponentially in the number of successive actions to be considered. In order to overcome this increase in complexity while still retaining the ability to consider remote locations for active localization, we have chosen our special

Action a	Percept s	$P(s \mid a)$	$Bel(L_{t+1} \mid a, s)$	$H(L_{t+1} \mid s, a)$	$U_t(a)$
`Forward`	<DL>	1/6		0	0.91
`Forward`	<WL>	1/3		1	
`Forward`	<DR>	1/6		0	
`Forward`	<WR>	1/3		1	
`Backward`	<DL>	1/6		0	0.91
`Backward`	<WL>	1/3		1	
`Backward`	<DR>	1/6		0	
`Backward`	<WR>	1/3		1	

Tab. 4.2. Components of the utilities when considering both sensors independently.

representation for actions. Here, even the very next action can guide the robot to a remote target location (c.F. 4.3).

For computing the utility of an action, we only consider the decrease in uncertainty when firing the robot's sensors *at the target location.* Instead one could think of trying to additionally estimate the information gathered *on the trajectory to the target location.* Due to the uncertainty in robot motion and the dependency of the belief states (and thus the entropies) at successive points in time, estimating the overall information gain on the trajectory would result in a task almost as complex as computing the optimal navigation strategy based on POMDPs.

Example for Sub-optimal Estimation of Utilities

Figure 4.5 illustrates a situation in which ignoring the information gathered on the way to a target location results in a non-optimal evaluation of the utility of actions.

Here the robot can be in one of the four different locations depicted in Figure 4.5. The sensors of the robot are the same as in our previous example and we only consider the following two actions: (1) move to a location two steps `Ahead` and (2) move to a location

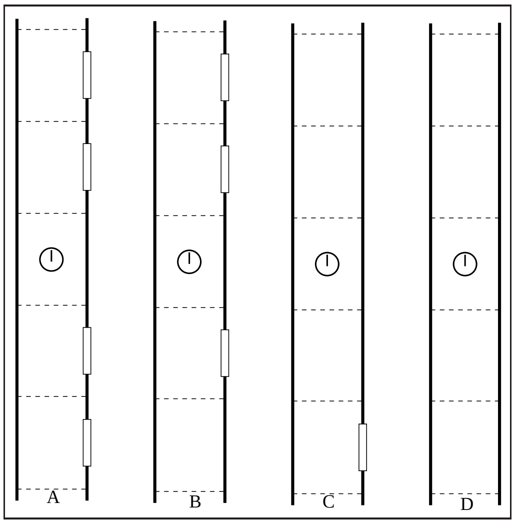


Fig. 4.5. Example situation for sub-optimal action evaluation.

two steps `Behind`. By taking into account that the robot is able to perceive its environment when moving to a target location, it is straightforward to distinguish the utility of these two actions. When moving two steps forward, the robot is not able to distinguish the location in the corridor labeled A from the location in corridor B because in both cases it will make the sequence (<WL,DR>, <WL,DR>) of percepts. The same problem holds for distinguishing location C from location D, because both start locations result in the percepts (<WL,WR>, <WL,WR>) when moving two steps forward. In contrast to this, the robot is able to uniquely determine its position when moving two steps backwards because all possible locations result in a unique sequence of percepts.

This preference of the backward action cannot be detected by our efficient implementation because we only consider the percepts made at the target location: If the robot moves two steps backwards and fires its sensors, it is still confused about being in corridor A or C (percept <WL,DR>) or about being in corridor B or D (percept <WL,WR>). Thus in our simplified implementation both actions result in an ambiguous position estimation with the same utility.

Here we only discussed the problem of ignoring the information gathered as the robot moves to a remote target location. But what happens if the robot gets *more uncertain* about its position while navigating to such a target location? This might happen if the robot has to cross a large open area in order to reach a target. While our efficient implementation provides no means to handle such situations we fortunately never observed such pathological cases in our office environment.

4.5.3 Additional Efficiency Considerations

The active navigation and sensing methods described here have been implemented and tested using *position probability grids* introduced in Section 2.6. As noted above, the complexity for determining the utility of an action a is in $O(|L| \cdot |S|)$. In the previous section we showed how to reduce the number S of possible percepts. In order to make the computation of the utilities even more efficient, we additionally reduce the number L of considered locations. Instead of taking into account all locations l in the environment we only consider the set L_m of local maxima of the belief state $Bel(L)$. These local maxima are extracted from the overall belief as described in Section 2.6.3. While this often reduces the umber of considered locations by more than four orders of magnitude we are convinced that the set L_m of local maxima represents the most important aspects of the belief state: L_m can be seen as the set of hypotheses about where the robot might be. Our approach can also be seen as a probabilistic version of *hypothesis elimination* discussed in [DRW98] (see also Section 4.2).

The resulting complexity of computing the best action is in $O(|L| \cdot |L_m| \cdot |S|)$. In combination with the fast sensor model introduced in Section 2.6.1 to compute the quantities $P(s \mid l)$, action selection can be performed in reasonable time for active navigation and in real-time for active sensing (see Section 4.6). The examples provided in this section show that our efficient implementation occasionally fails to estimate the true utility of actions. Nevertheless, in our extensive experiments, we never experienced that the robot was not able to globally localize itself. Furthermore, in our real world experiments described in Section 4.6.1, we did not observe situations as those presented here.

4.6 Experimental Results

In this section we test the influence of our active extension to Markov localization on the performance of the position estimation.

4.6.1 Active Navigation

Active navigation was tested by placing the robot in the corridor of our department as plotted in Figure 4.6. The task of the robot was to localize itself using only its ultrasound sensors. Notice that the corridor in this environment is basically symmetric and possesses various places that look alike, making it difficult for the robot to determine where it is. In this environment, the robot must move into one of the offices, since only here it finds distinguishing features due to the different furniture in the offices.

Please note that the state of the doors as depicted in Figure 4.6 has no influence on the position estimation because the doors are not modeled in the map used for localization.

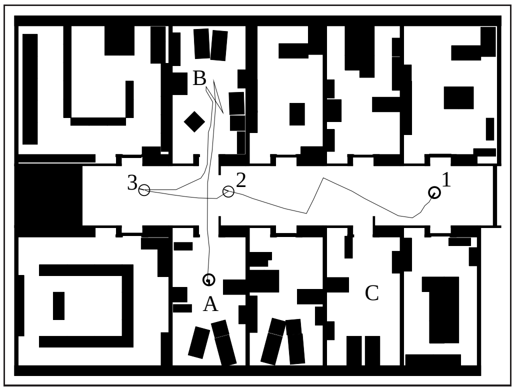


Fig. 4.6. Outline of the environment and path of the robot

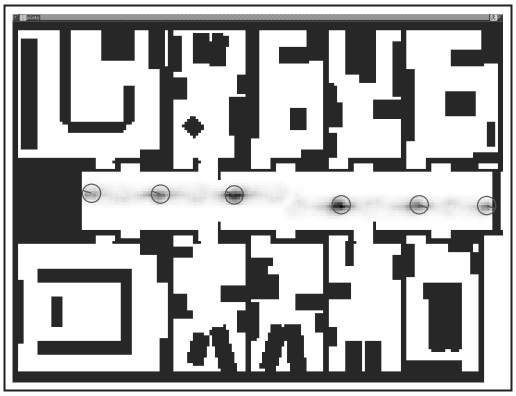


Fig. 4.7. Belief $Bel(L)$ position 2

The doors are only considered when determining the expected costs $C(a)$ of moving into the different rooms. The specific configuration shown in the figure is used for illustrative purposes. The robot is only able to uniquely localize itself by moving either into room B or room C.

The path shown in Figure 4.6 is a representative example of the path taken during active localization. In this particular run we started the robot at position 1 in the corridor facing south-west. The task of the robot was to determine its position within the environment and then to move into room A (so that we could see that localization was successful).

In order to reduce the time to compute the utility of the actions (see Section 4.5), the robot starts with a wandering routine until the belief is concentrated on several local maxima (hypotheses). These local maxima correspond to the reduced representation of the belief state introduced in Section 2.6.3. This early stage of the localization procedure is a straightforward implementation of "hypotheses generation" discussed e.g. in [Kle94; DRW98]. After about ten meters of motion, the robot reaches position 2 (c.F. 4.6). Figure 4.7 depicts the belief $Bel(L)$ at this point in time (more likely positions are darker). At this point in time the set L_m consists of six local maxima, whose positions and orientations are marked by the six circles. Only these local maxima are considered for computing the utility and costs of actions.

The expected occupancy probabilities $P_{occ}(a)$, obtained by Eq. (4.9), are depicted in Figure 4.8 (a). High probabilities are shown in dark colors. Note that this figure roughly corresponds to a weighted overlay of the environmental map relative to the six local maxima, where the weights are given by the probabilities of these maxima. Figure 4.8 (a) also contains the origin of the corresponding coordinate system. This point represents the current position of the robot or the action $a = move(0\text{m}, 0\text{m})$, respectively (with robot facing right). Figure 4.8 (b) displays the expected costs for reaching the different target points using the occupancy probabilities from Figure 4.8 (a). These costs have been computed

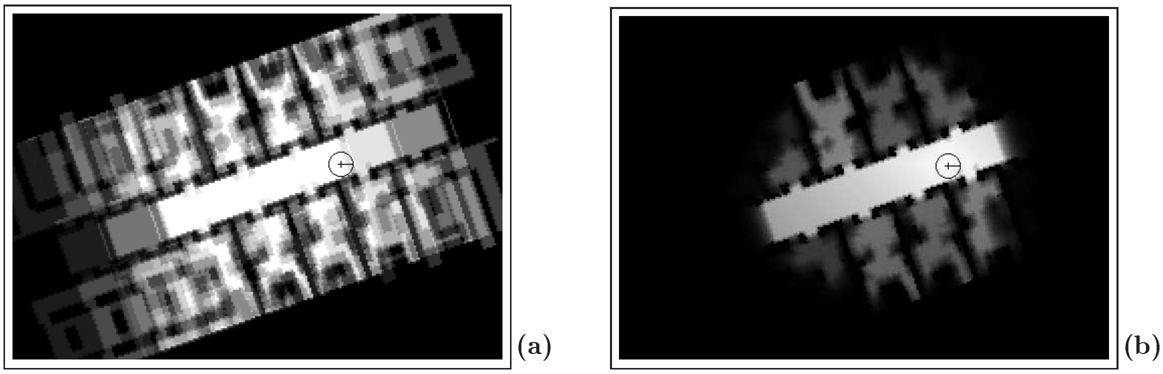


Fig. 4.8. (a) Occupancy probabilities $P_{occ}(a)$ and (b) expected costs $C(a)$ at pos. 2

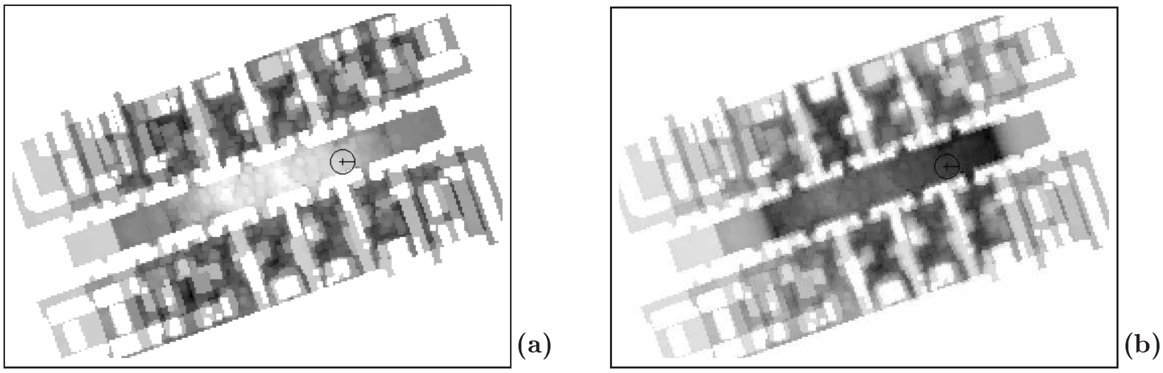


Fig. 4.9. (a) Utility $U(a)$ and (b) payoff $U(a) - \beta \cdot C(a)$ at pos. 2

using value iteration based on Eq. (4.10).

Figure 4.9 (a) shows the utility $U(a)$ of the target points according to Eq. (4.8). As can be seen there, the expected entropy of locations in rooms is low, thus yielding high utilities and making them favorable for localization. The utility is also high, however, for the two ends of the corridor, since those can further reduce uncertainty. Based on the utility-cost trade-off depicted in Figure 4.9 (b), the robot now decides to first pick a target at the end of the corridor. At this point it is important to notice that the exact trajectory from the current position to the target point cannot be computed off-line. This is due to unavoidable inaccuracies in the world model and to unforeseen obstacles in populated environments such as our office. These difficulties are increased if the position of the robot is not known, as is the case during localization. To overcome these problems the robot must be controlled by a reactive collision avoidance technique. In our implementation a global planning module uses dynamic programming as described in section 4.4.1 to generate a minimal cost path to the target location (see [TB96]). Intermediate target points on this path are presented to our reactive collision avoidance technique described in chapter 5. The collision avoidance then generates motion commands to safely guide the robot to these targets.

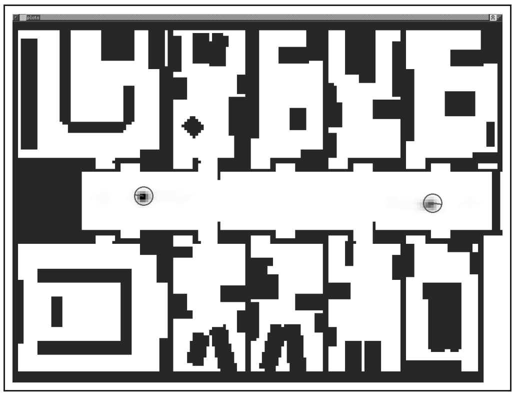


Fig. 4.10. Belief $Bel(L = l)$ at pos. 3

After having reached the end of the corridor (position 3 in Figure 4.6) the belief state contains only two local maxima (see Figure 4.10). The occupancy probabilities and the resulting costs of the different actions for this belief are depicted in Figures 4.11 (a) and (b), respectively. Please note that due to the state of the doors, the costs for reaching room B or C are remarkably lower than those for reaching the other rooms.

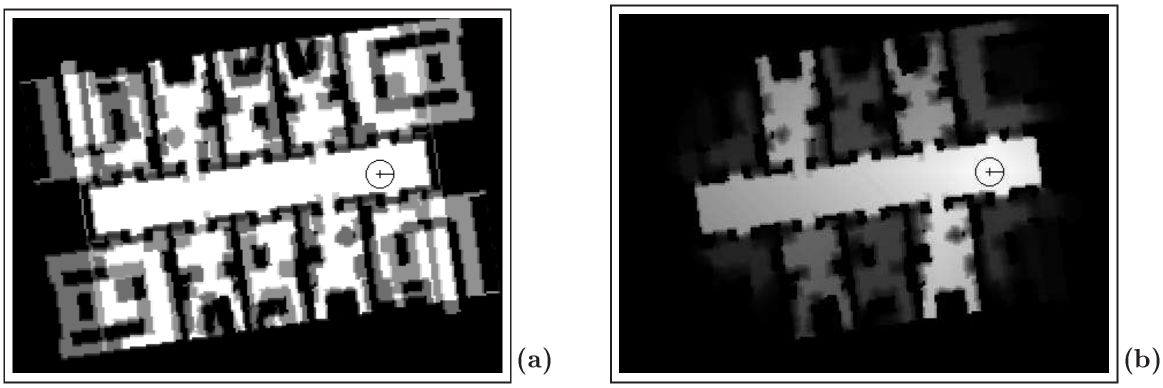


Fig. 4.11. (a) Occupancy probabilities $P_{occ}(a)$ and (b) expected costs $C(a)$ at pos. 3

The ambiguity in the belief at position 3 can no longer be resolved without leaving the corridor. Accordingly the utility shown in Figure 4.12 (a) is low for target points in the corridor compared to the utility of actions which guide the robot into the rooms. Because of the state of the doors and the resulting costs, the overall payoff as displayed in Figure 4.12 (b) is maximal for target points in rooms B and C.

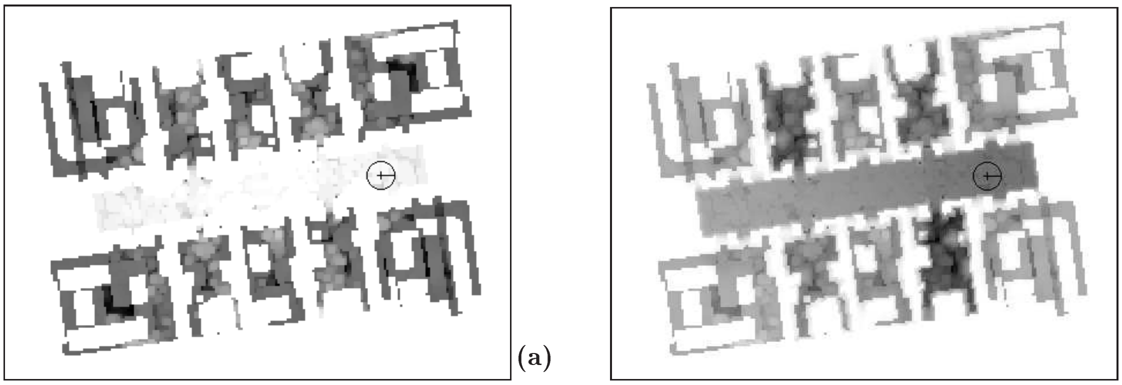

(a) (b)

Fig. 4.12. (a) Utility $U(a)$ and (b) payoff $U(a) - \beta \cdot C(a)$ at pos. 3

As shown in Figure 4.6 the robot decides to move into the room behind him on the right, which in this case turned out to be room B. Here the robot has been able to resolve the ambiguity between the rooms B and C based on the different furniture in the two rooms. After having uniquely determined its location the robot moved straight to the target location in room A. Figure 4.13 shows the belief state at this final point.

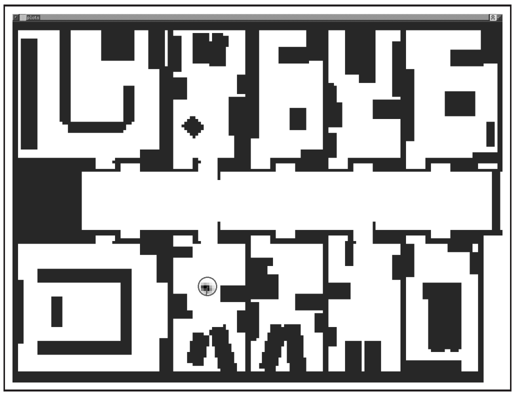


Fig. 4.13. Final belief $Bel(L)$

The path described above is just one example for the more than 20 experiments we conducted using the active navigation approach presented here. In each case the robot was placed in the corridor and had to reach room A. In these experiments, the robot always managed to localize itself in a considerably short amount of time (the whole experiment described here took less than 12 minutes). In a total of 10 additional experiments, random wandering and/or wall following consistently failed to localize the robot. This is because our wandering routines are unable to move the robot through narrow doors, and the symmetry of the corridor made it impossible to uniquely determine the location without leaving the corridor.

In addition to runs in our real office environment we did preliminary testing in artificial hallway environments taken from [KCK96]. Here, our active navigation system successfully localized the robot by automatically detecting junctions of hallways and openings as crucial points for the localization task, and was uniformly superior to passive or random localization. A more systematic evaluation of our technique in these environments and a comparison with other methods than random wandering[3] is beyond the scope of this thesis and is a topic of future work.

4.6.2 Active Sensing

In the following experiments we demonstrate how the efficiency of localization can be improved by choosing the optimal pointing direction of the robot's sensors. Our experiments are conducted with two different sensors (ultrasound and laser-range finder) in order to demonstrate the ability of our technique to select the kind of sensor which is most appropriate in the given situation (e.g. a camera mounted on a pan head with two different zoom values, one corresponding to sonars and one corresponding to lasers). The difference between ultrasound sensors and laser-range finders lies in their accuracy. While ultrasound sensors have an angular resolution of 15°, our laser-range finders have an angular resolution of 1°. In addition to this, laser-range finders are able to measure obstacles more accurately than ultrasound sensors. Please note that in our approach these differences are represented solely in the model of the sensors, specifically in the probabilities $P(d_i \mid l)$ of measuring distance d_i if the robot is at location l (see Section 2.6.1 and especially Figure 2.3 for more details).

Figure 4.14 shows the setup of the experiments: the robot was placed in the corridor and moved up and down with a constant velocity of 30 cm/sec. As can be seen here, the corridor ($23 \times 4.5\,m^2$, all doors closed) is symmetric except for a single obstacle on its side and this obstacle can only be detected by the robot's sonar sensors. Thus, to uniquely determine its location, the robot has to sense this obstacle with a sonar sensor.

To simulate active sensing, we allowed the robot to read only a single sensor at any point in time. As our passive method, we chose a sensor at random. This passive method was compared to our active approach, where sensors are chosen by maximizing decrease in entropy (see Eq. (4.11)). To evaluate the quality of localization we use the normed Bayesian estimation error in localization $E(L)$ as given in Eq. (4.13). Here $l^\star$ denotes the true location of the robot, which we determined off-line. In order to combine translational and rotational error we compute $\|l^\star - l\|$ by a weighted sum of both deviations (see Eq. (4.14)),

[3] In our office environment, for example, following a wall and moving into the next room might result in a performance comparable to our active approach.

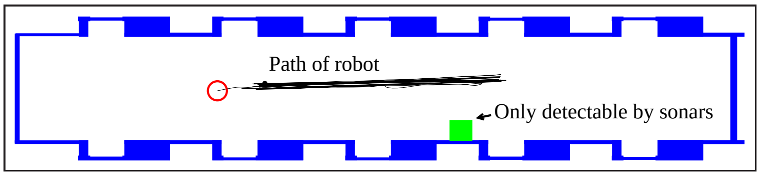


Fig. 4.14. Corridor of the department and path of the robot.

where x_{max} ,y_{max}, and α_{max} denote the maximal possible deviation in the corresponding coordinate.

$$E(L) = \sum_l Bel(L = l)\ \|l^\star - l\| \tag{4.13}$$

$$= \sum_l Bel(L = l)\ \left(0.5\ \sqrt{\frac{(l_x^\star - l_x)^2 + (l_y^\star - l_y)^2}{x_{max}^2 + y_{max}^2}} + 0.5\ \sqrt{\frac{(l_\alpha^\star - l_\alpha)^2}{\alpha_{max}^2}}\right) \tag{4.14}$$

In the first experiment the robot was only allowed to choose among its 24 sonar sensors. The results are depicted in Figure 4.15 (a). This figure plots the normed error $E(L)$ as a function of time, averaged over 12 runs, along with their 95% confidence intervals (bars). As can be seen here, the error decreases much faster when sensors are selected actively (solid line). This result clearly demonstrates the benefit of active sensing.

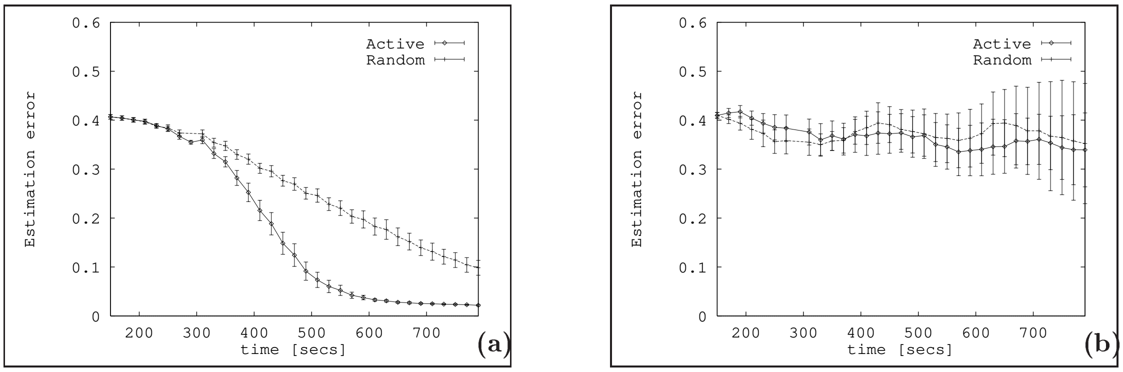


Fig. 4.15. Estimation error when using (a) only sonar and (b) only laser-range finder .

Figure 4.15 (b) depicts the corresponding results for the laser range-finder. In this case active sensing is not superior to choosing a pointing direction randomly. Obviously none of the methods is able to uniquely determine the position of the robot, which is due to the inability of the laser-range finder to detect features breaking the symmetry of the corridor. During this kind of experiments we always observed two local maxima in the position probability distribution, one representing the true location of the robot and one representing the position mirrored by 180°.

In the final experiment the robot was allowed to choose among both sonar sensors and laser range-finders. The result is depicted in Figure 4.16. Here again, active sensing significantly improves the efficiency of the localization process. In order to get an impression of the qualitative difference between active sensing and random selection we additionally evaluated this experiment by using a selection strategy with perfect knowledge: the sensing direction was selected *after* the sensors have been fired. Again, we chose the sensing which mostly reduces the entropy of the belief [4]. The corresponding reference values are represented by the dotted line in Figure 4.16. As can be seen there, our active sensing strategy is within reasonable bounds even compared to this "perfectly informed" strategy.

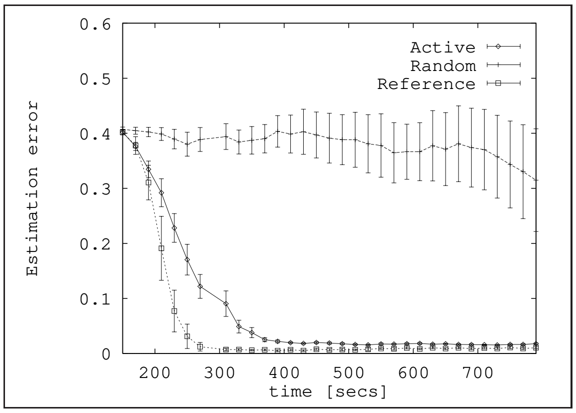


Fig. 4.16. Estimation error when using both sensors.

Figures 4.17 (a) and (b) depict two other measures for this final experiment. The evolution of the entropy over time is plotted in Figure 4.17 (a). As expected the three curves are very similar to those of the estimation error in Figure 4.16. In order to estimate the certainty of the different approaches of being at the true location l^* we summed up the probabilities assigned to positions close to the true location of the robot. These values are depicted in Figure 4.17 (b). As can be seen here active sensing is extremely certain of being at the true location after less than 400 seconds. The random strategy on the other hand is not able to uniquely determine the position of the robot within the scope of the experiment.

To shed light onto the question as to why active localization performs significantly better than the passive method we analyzed the sensor readings that the two methods considered

[4]Please note that this strategy is only used for illustrational purpose because it cannot be applied to active sensing, where the pointing direction has to be selected *before* having perceived the sensor values.

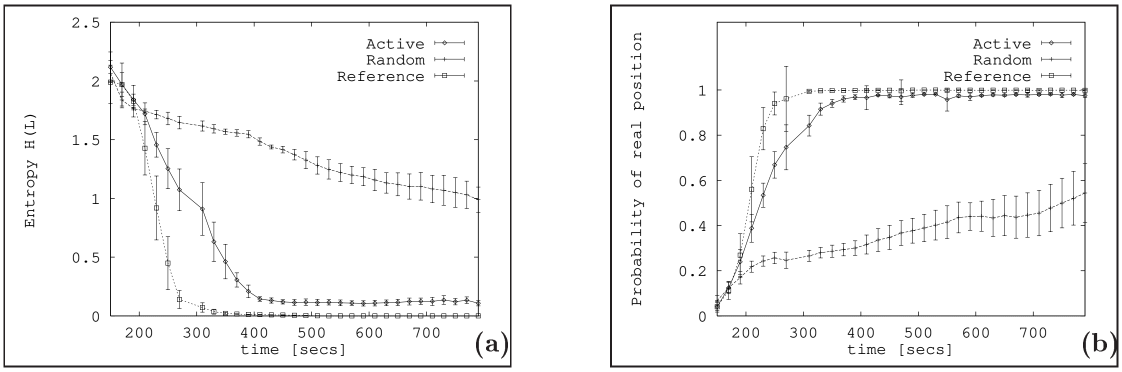


Fig. 4.17. (a) Entropy of belief states and (b) probability of real positions.

during localization. Figures 4.18 (a) and (b) plot the sonar and laser measurements that were incorporated into the robot's belief when choosing readings randomly. The drawn points have been generated by storing the sensor measurements considered for localization and plotting their end points relative to the true position of the robot at that time. As expected, the points are randomly scattered over the whole environment. Figures 4.19 (a) and (b) depict the end points of the sensor readings used for position estimation when applying our active strategy. Please note that these points stem from *one* experiment where the robot was allowed to choose among both sensors.

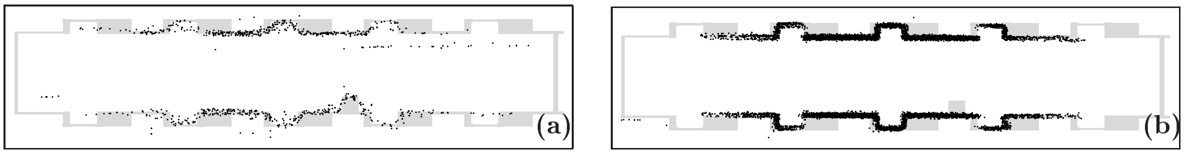


Fig. 4.18. End points of (a) sonar and (b) laser measurements of randomly selected readings.

Here the advantage of our method becomes obvious: the less accurate sonar sensors are only used to scan the box and its symmetric position (upper left cloud of points) in order to break the symmetry of the corridor. In most other cases the laser sensors are preferred due to their higher accuracy (c.F. 4.19(b)). This also explains why active sensing when using both sensors is significantly better than active sensing when using only one kind of sensor (c.F. 4.15 and 4.16): By selecting the right kind of sensor it combines the accuracy of the laser range-finder with the ability of the sonar sensors to disambiguate the position estimation.

4.7 Discussion

This chapter advocates a new, active approach to mobile robot localization. In active localization, the robot controls its various effectors so as to most efficiently localize itself.

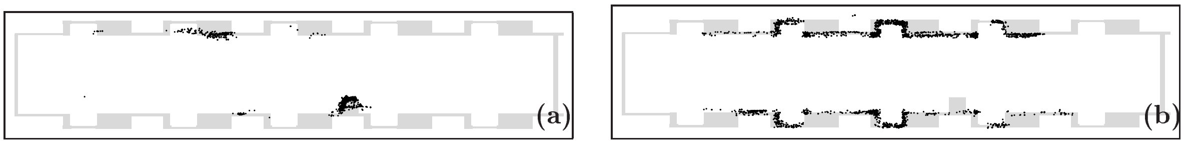


Fig. 4.19. End points of (a) sonar and (b) laser measurements selected in order to minimize uncertainty.

In essence, actions are generated by minimizing the future expected uncertainty, measured by entropy. This basic principle has been applied to two active localization problems: *active navigation*, and *active sensing*. In the case of active navigation, expected costs are incorporated into the action selection. Both approaches have been verified empirically using our RWI B21 mobile robot.

The key results of the experimental comparison are:

1. The efficiency of localization is increased when actions are selected by minimizing entropy. This is the case for both active navigation and active sensing. In some cases, the active component enabled a robot to localize itself where the passive counterpart failed.

2. The relative advantage of active localization is particularly large if the environment possesses relatively few features that enable a robot to unambiguously determine its location.

Despite these encouraging results, there are some limitations that deserve future research. Our implementation of active navigation is based on the assumption that the environment is static. While minor changes such as people passing by have no significant impact on our technique, opened or closed doors can dramatically change the costs for reaching target locations behind these doors. Therefore a main topic of future work is to estimate the state of doors during global localization. We are confident that this problem can be solved because even though the position of the robot is not known during localization, we only need to know the state of doors *relative* to the robot in order to compute the costs of actions.

Another key limitation arises from the algorithmic complexity of the entropy prediction. While some algorithmic tricks made the computation of entropy feasible within the complexity bounds of our environment, more research is needed to scale the approach to environments that are significantly larger (e.g., 1000m$\times$1000m). A hierarchical representation of the environment in combination with a focussed search for actions might already result in a much more efficient implementation of the approach proposed here. A second limitation arises from the greediness of action selection. In principle, the problem of optimal exploration is NP-hard, and there exist situations where greedy solutions will fail. However, in none of our experiments we ever observed that the robot was unable to localize itself using our greedy approach, something that quite frequently happened with the passive counterpart.

Chapter 5

A Hybrid Approach to Collision Avoidance

5.1 Introduction

To operate successfully in populated environments, mobile robots must be able perceive their environments and react to unforeseen circumstances, and (re)plan dynamically in order to achieve their missions. To ensure safe navigation, most existing robot systems rely on reactive collision avoidance modules to control the robot (see e.g. [Nil84; Fir94; KMRS97; SGH+; TBB+98]). The predominant paradigm of these approaches to collision avoidance is strictly sensor-based: Sensor readings are continuously analyzed to determine collision-free motion. Because only a small fraction of the environment is considered, such local techniques have the obvious disadvantage that they cannot produce optimal solutions: They are easily trapped in local minima (such as U-shaped obstacle configurations). In order to overcome this inherent disadvantage, these approaches are often combined with a global path planner that generates intermediate target locations or directions to guide the collision avoidance module.

Unfortunately, the sensor-based paradigm of these reactive components has important limitations. If the environment is complex, it might be difficult to equip a robot with a sensor suite capable of detecting arbitrary obstacles. For example, if the environment possesses large obstacles made of glass, light-based sensors will not be able to detect them and even sound-based sensors such as sonars usually have severe problems due to specular reflections, which often occur at smooth surfaces such as glass. The severity of the problem increases with the speed of the robot, as obstacles have to be detected early enough to allow the robot to decelerate safely.

The problem of undetectable obstacles was a a major obstacle in the way of successful operation in the "Deutsches Museum Bonn" (see also Section 2.7.2). What made this task specifically challenging for safe navigation was the nature of the environment. While

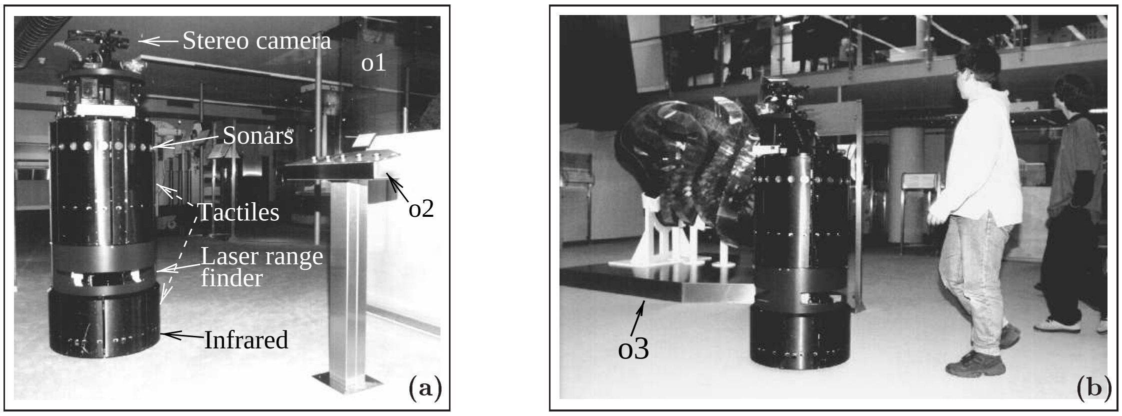


Fig. 5.1. (a) Sensors of the robot RHINO. The label $o1$ highlights an almost invisible glass surface, and the label $o2$ shows a metal console which is just below the robot's sonar sensors. (b) The label $o3$ points to another hard-to-detect obstacle.

RHINO is equipped with five different sensor systems (see Figure 5.1 (a)), various obstacles were virtually undetectable for the robot, such as: glass cages, put up to protect exhibits (see e.g. label $o1$ in Figure 5.1 (a)), metal bars at various heights (label $o2$), small metal plates on which exhibits were placed (label $o3$ in Figure 5.1 (b)), just to name some. For the museum tour-guide application to be successful, the robot had to move at walking speed. Avoiding collisions was of utmost importance due to the nature of the "obstacles" in a museum. The reader may notice that similar conditions are expected to be found in private homes and various other anticipated task environments for future service robots.

In this chapter we propose a solution to the problem of undetectable obstacles. Our hybrid approach to reactive collision avoidance integrates sensor information with model-based information about obstacles. It ensures that with high likelihood, the robot avoids collisions even with obstacles not detectable with its sensors. This technique is based on the dynamic window approach [BBC+95; FBT96; FBT97], a strictly sensor-based collision avoidance technique which is especially designed to consider the dynamics of mobile robots. The model-based extension, called μDWA (short for *model-based dynamic window approach*), differs from most previous techniques in that it

1. considers the *dynamics* of the robot and
2. integrates *model-based* information about obstacles into robot control.

In order to integrate model-based information into the dynamic window approach, μDWA generates virtual sensor readings. These readings simulate a proximity sensor mounted on the robot which is able to reliably detect *all* obstacles represented in the world model. The basic precondition to generate such readings is that the location of the robot relative to the obstacles in the map has to be known. μDWA applies our metric version of Markov localization as introduced in Section 2 to estimate the position of the robot within the map. By considering the belief state of the position estimation, our approach is able to avoid collisions with high (e.g. 99%) probability even when the robot is not certain about its position, which is the case especially during global localization. Thus, μDWA allows a mobile robot to safely localize itself within its environment (see Section 4).

The remainder of this chapter introduces our hybrid approach to collision avoidance. After discussing related work, Section 5.3 describes the purely sensor-based dynamic window approach. In Section 5.4 we introduce our extension to hybrid collision avoidance by generating *virtual* sensor readings using a geometric map of the environment. Experimental results are summarized in Section 5.5, followed by a discussion of further research issues.

5.2 Related Work

Some of the most popular methods to reactive collision avoidance are based on the physical analogy of potential fields. These approaches, as proposed in [Kha86], determine the steering direction of the robot by (hypothetically) assuming that obstacles assert negative forces on the robot and that the target location asserts a positive force. These methods are extremely fast, and they typically consider only the small subset of obstacles close to the robot. While the physical analogy of considering the robot as a free-flying object is attractive, Koren and Borenstein identified that in practice, such methods often fail to find trajectories between closely spaced obstacles [KB91a]. An extended version of the potential field approach is introduced in [KC95]. By modifying the potential function the motion of the robot becomes more efficient and different behaviors such as wall following and tracking can be achieved.

Borenstein and Koren proposed the *virtual force field histogram* [BK90] which closely resembles potential field methods. This approach uses an occupancy grid to represent information about obstacles close to the robot. This grid is generated and updated continuously using ultrasonic proximity sensors. In 1991 Borenstein and Koren extended this approach to the *vector field histogram* [BK91]. Occupancy information is transformed into a histogram description of the free space around the robot, which is used to compute the motion direction and velocity for the robot.

As is the fact for most existing approaches to reactive collision avoidance, the techniques discussed so far generate motion commands for mobile robots under the assumption that infinite forces can be asserted on the robot. Thus, the dynamics of the robot are not considered when determining a desired motion direction. However, for safe navigation of robots with limited accelerations it is necessary to take into account the impulse of the robot (see Section 5.5 for a further discussion of this topic). Simmons developed independently from our dynamic window approach technique the *curvature velocity method* [Sim96]. Both approaches are especially designed to deal with the dynamics of mobile robots. The search for motion commands is performed directly in the space of translational and rotational velocities and it is assumed that the robot moves on circular arcs. In both approaches the dynamics of the robot are considered by constraining the search space of admissible velocities.

In order to deal with obstacles that cannot be detected by the robot's sensors it is necessary to integrate model-based information into reactive collision avoidance. Only considerably small attention has been payed to this problem in the literature. Schmidt and Azarm proposed an approach to motion planning which in principle could solve this problem [SA92]. The emphasis of their work lies in the combination of global path planning and local collision avoidance, not on the problem of avoiding invisible obstacles. Here, motion commands are generated based on a global model of the environment which is updated based on sensory input. They propose a method which efficiently extracts a path to a goal point based on such a map. However, the authors do not consider the problems of robot dynamics and uncertainty in the position estimation. Furthermore it is not clear how the static model of the environment is combined with the quickly changing information obtained on-the-fly.

5.3 The Dynamic Window Approach

In the dynamic window approach control is chosen in the space of velocities. For synchro-drive robots the space of possible control is parameterized by the *translational* and *rotational* velocity. In appendix B we show that it is possible to approximate the trajectory of synchro-drive robots by a sequence of circular arcs (curvatures). Each such curvature is uniquely determined by a combination of a translational and a rotational velocities: As long as the velocities are not changed, the robot will move on the same curvature. The diameter of such a circular arc is determined by the ratio of translational and rotational velocity[1].

In regular time intervals (e.g., every .25 seconds), our approach chooses velocities so as to best obey various constraints and preferences. *Constraints* are vital for the robot's safety and are imposed by torque limits. For example, the maximum torque induces a maximum change of velocity, which severely limits the space of possible control (e.g., a fast moving robot cannot take a 90 degree turn). Constraints are also imposed if a velocity would inevitably lead to a collision with an obstacle. These constraints rule out certain controls from further consideration. Notice that they do not specify preferences among the different control options; neither do they take into account the robot's task. *Preferences* are expressed for both the motion direction and the velocity of the robot. They are modeled by an evaluation function which is maximized in order to find the best combination of

[1]The case of straight motion (rotational velocity of zero) is a special case which is treated analogously.

translational and rotational velocity to control the robot.

After describing the representation of sensor data, we will discuss the different constraints and preferences applied in the dynamic window approach.

Representation of Sensor Data

As local world model we use a so-called obstacle line field [BBC+95], which is a two-dimensional description of sensory data relative to the robot's position. It contains a line for each reading of a sonar sensor, which is perpendicular to the main axis of the sensor beam at the measured distance. The length of the line is determined by the breadth of the beam in the given distance. Measurements obtained from laser range-finders are represented by points at the measured distance. Sensor information of infrared and tactile sensors is included in the same way. To ensure that the robot is able to quickly adapt to changes in the environment, obstacle information is only stored for a certain amount of time, which is usually less than 2 seconds. Figure 5.2 (a) shows a scan of RHINO's two laser range-finders taken in the "Deutsches Museum Bonn". The corresponding obstacle line field is depicted in Figure 5.2 (b) (sonar readings are plotted as lines).

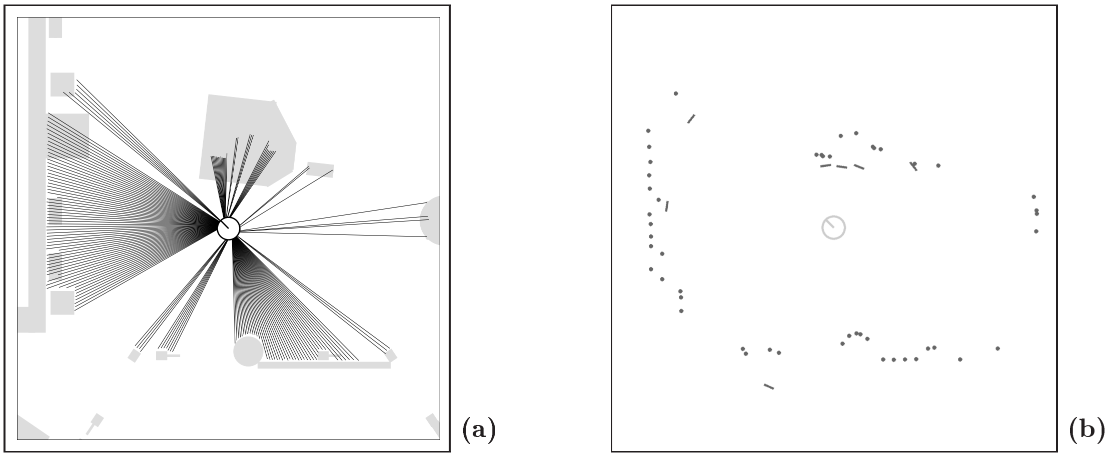


Fig. 5.2. (a) Scan of the laser sensors and (b) corresponding obstacle line field.

Using this obstacle representation, the distance to obstacles on a certain curvature can be computed by simple geometric reasoning: let $r = v/\omega$ be the radius of a circular trajectory, and let γ be the angle between the intersection with an obstacle and the position of the robot (see Figure 5.3); then the distance to this obstacle on the corresponding curvature is given by $\gamma \cdot r$.

In the remainder of this section we will use the situation plotted in Figure 5.2 (b) to illustrate the different aspects of the constraints imposed on curvatures.

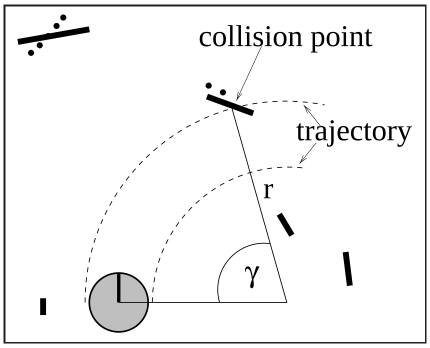


Fig. 5.3. Determination of the distance to obstacles using the obstacle line field.

5.3.1 Constraints

Safe Velocities

In order to guarantee safe navigation, the robot is not allowed to choose velocities which could cause a collision with an obstacle. Thus the space V_s of safe velocities is restricted to those velocities, for which the robot is able to stop before it reaches the closest obstacle on the corresponding curvature. Let $\mathrm{dist}(v, \omega)$ denote the distance to the closest obstacle on the curvature defined by the velocity (v, ω). This distance can be computed based on the obstacle line field described in the previous section. Furthermore let $\dot{v}_{max}$ and $\dot{\omega}_{max}$ denote the maximal translational and rotational acceleration, respectively. To ensure that the robot stays on the chosen curvature during breakage, the accelerations $\dot{v}_b$ and $\dot{\omega}_b$ used for breakage have to suffice the following constraint. $\frac{\dot{v}_b}{\dot{\omega}_b} = \frac{v}{\omega}$. By assuming that the robot always chooses maximal accelerations for breakage, we can set $\dot{v}_b = \min(\dot{v}_{max}, \frac{v \cdot \dot{\omega}_{max}}{\omega})$.

With these considerations the set V_s of safe velocities is defined as follows.

$$V_s = \left\{ (v, \omega) \mid \dot{v}_b = \min(\dot{v}_{max}, \frac{v \cdot \dot{\omega}_{max}}{\omega}) \ \wedge \ v \leq \sqrt{2 \cdot \mathrm{dist}(v, \omega) \cdot \dot{v}_b} \right\}. \tag{5.1}$$

Example: *Figure 5.4 shows the space of velocities considered for collision avoidance: the x-axis represents rotational velocities and the y-axis gives translational velocities (positive rotation results in a right-turn). Each point in this two-dimensional space corresponds to a certain curvature. Now consider the situation depicted in Figure 5.2 (b). The safe velocities in this situation given maximal accelerations of* $\dot{v}_{max} = 50$ *cm/sec*^{2} *and* $\dot{\omega}_{max} = 60$ *deg/sec*^{2} *are represented by the shaded area in Figure 5.4. The curvatures on which the robot is not able to stop before reaching an obstacle are given by the white area. When comparing this area with the obstacle line field it becomes clear that these velocities are not safe because they would result in a collision with the obstacles on the right side of the robot.*

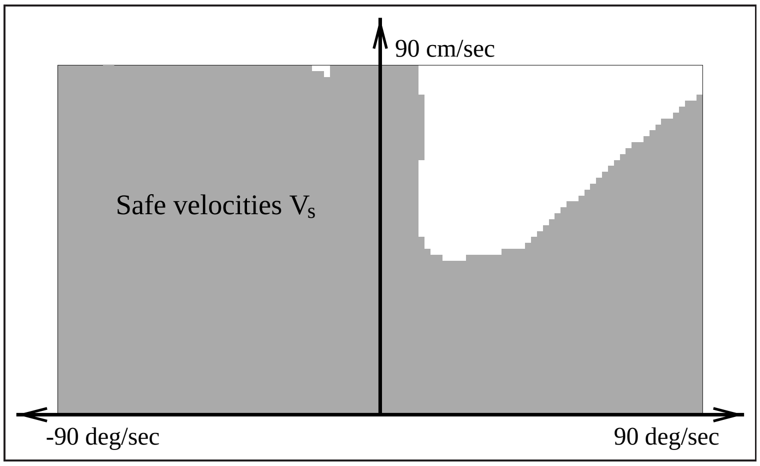


Fig. 5.4. Safe velocities given the situation depicted in Figure 5.2 (b)

Dynamic Window

In order to take into account the limited accelerations exertable by the motors, the space of considered velocities is reduced to the *dynamic window* which contains only the velocities that can be reached within the next time interval. Let t be the time interval during which accelerations $\dot{v}$ and $\dot{\omega}$ will be applied and let $(v_c,\, \omega_c)$ be the current velocity. Then the dynamic window V_d is defined as

$$V_d = \{(v, \omega) \mid v \epsilon [v_c - \dot{v} \cdot t, v_c + \dot{v} \cdot t] \wedge \omega \epsilon [\omega_c - \dot{\omega} \cdot t, \omega_c + \dot{\omega} \cdot t]\} \,. \tag{5.2}$$

The dynamic window is centered around the current velocity and the extensions of it depend on the accelerations that can be exerted. All curvatures outside the dynamic window cannot be reached within the next time interval and thus are not considered for controlling the robot.

Example: *A dynamic window obtained for our example situation given accelerations of 50 cm/sec^2 and 60 deg/sec^2 and a time interval of 0.25 sec is shown in Figure 5.5. The position of the dynamic window stems from a current velocity of 40 cm/sec straight motion. The two dotted arrows pointing to the corners of the rectangle denote the most extreme curvatures that can be reached.*

These restrictions imposed on the space of velocities result in the area V_a of admissible velocities within the dynamic window. This area is is defined as the intersection of the restricted areas: $V_a = V_s \cap V_d$.

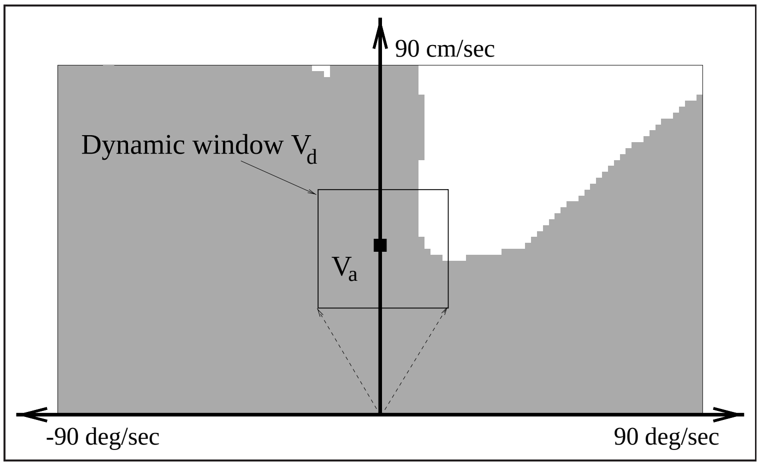


Fig. 5.5. Dynamic window

5.3.2 Preferences

Our approach utilizes the following three preferences to choose among the motion direction and velocity of the robot.

1. **Target heading:** is a measure of progress towards the goal location. It is maximal if the robot moves directly towards the target.

2. **Clearance:** is the distance to the closest obstacle on the trajectory. The smaller the distance to an obstacle the higher is the robot's desire to move around it.

3. **Velocity:** is the forward velocity of the robot and supports fast movements.

We will now describe these three preferences in more detail and apply them to our example situation.

Target Heading

The target heading heading(v, ω) measures the alignment of the robot with the target direction. It is given by $180 - \theta$, where θ is the angle between the robot's heading and the target point (see Figure 5.7).

Since this direction changes with the different velocities, θ is computed for a predicted position of the robot. To determine the predicted position we assume that the robot moves with the selected velocity during the next time interval. For a realistic measurement of the target heading we have to consider the dynamics of the rotation. Therefore, θ is computed at the position which the robot will reach when exerting maximal deceleration after the next interval. This consideration of the dynamics yields a smooth turning to the target in the behavior of the robot when it has circumvented an obstacle.

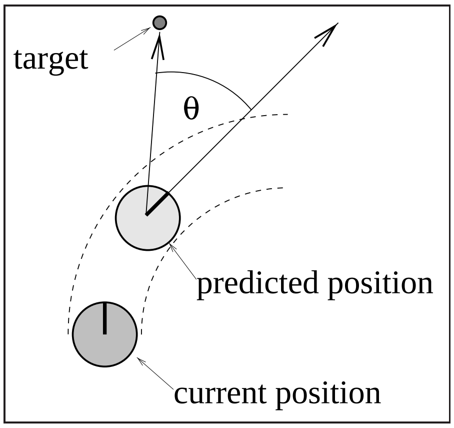


Fig. 5.6. Determination of the angle θ to the target.

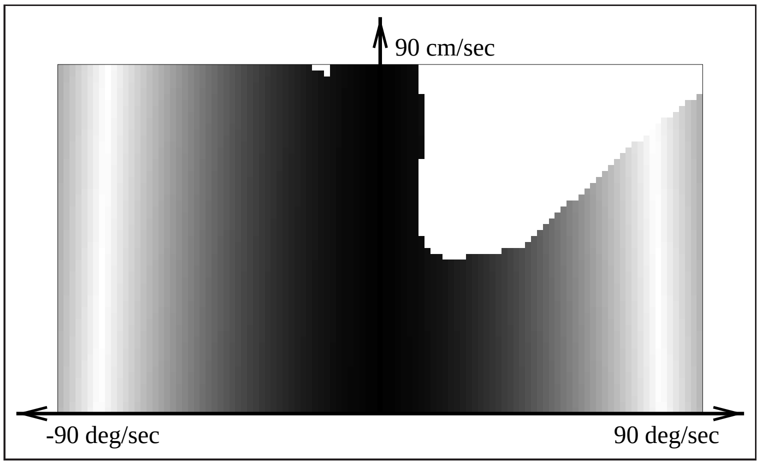


Fig. 5.7. Evaluation of the target heading for the velocities in the example situation.

Example: *Figure 5.7 shows the evaluation of the target heading for the different velocities in the example situation depicted in Figure 5.2 (b) (higher values are darker)* [2]. *Here it is assumed that the target location is in front of the robot. Therefore velocities yielding straight motion have maximal evaluation, while the function declines for higher rotational velocities, because in these cases the robot turns away from the target.*

[2] In this figure and the following figures the values for unsafe velocities are set to zero (compare with Figure 5.4). For illustrational purpose we show the evaluation of the whole velocity space and do not restrict it to the dynamic window.

Clearance

The function $\mathrm{dist}(v, \omega)$ represents the distance to the closest obstacle on the corresponding curvature. The determination of this distance is described in the beginning of Section 5.3. The distance is set to a large constant if no obstacle is on the curvature.

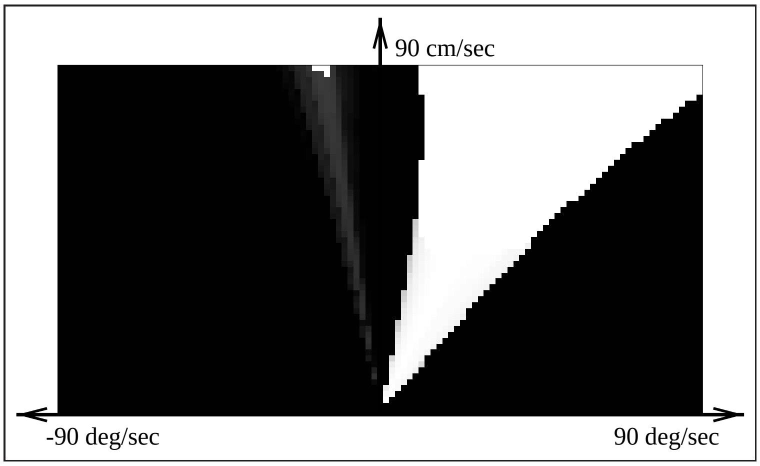


Fig. 5.8. Evaluation of the clearance for the velocities in the example situation.

Example: *The evaluation of the distances computed for the obstacle line field in Figure 5.2 (b) is depicted in Figure 5.8. In the given plot one can find low evaluations for those curvatures which lead the robot to the obstacles on its right side. In fact these are the curvatures which would cause a collision for high translational velocities (white area).*

Velocity

The function $\mathrm{velocity}(v, \omega)$ is used to evaluate the progress of the robot on the corresponding trajectory. It is simply a projection on the translational velocity v, as can be seen in Figure 5.9.

Combining the Preferences

In order to find the velocity which achieves the best behavior of the robot we trade off the three preferences:

$$G(v, \omega) \;=\; \sigma(\alpha \cdot \mathrm{heading}(v, \omega) \;+\; \beta \cdot \mathrm{dist}(v, \omega) \;+\; \gamma \cdot \mathrm{velocity}(v, \omega)) \tag{5.3}$$

Here α, β, and γ are used to weight the different components. σ smoothes the evaluation and thereby keeps the robot from choosing velocities close to non-admissible velocities.

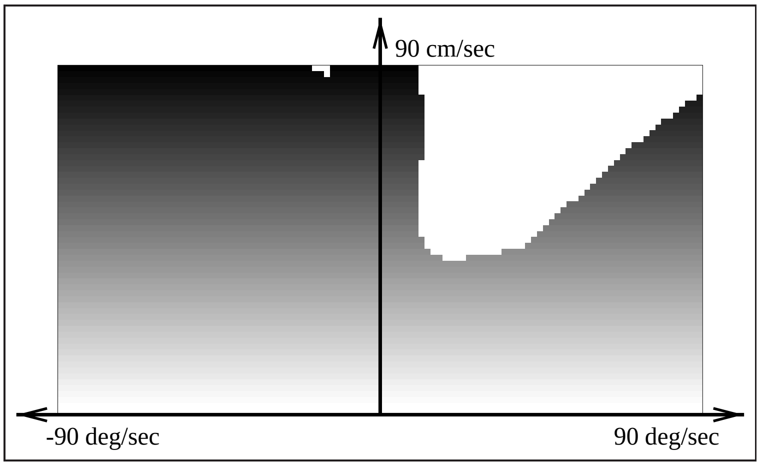


Fig. 5.9. Evaluation of the velocity for the curvatures in the example situation.

This smoothing results in an increased side-clearance in the actual behavior of the robot. The combined evaluation of the curvatures is depicted in Figure 5.10.

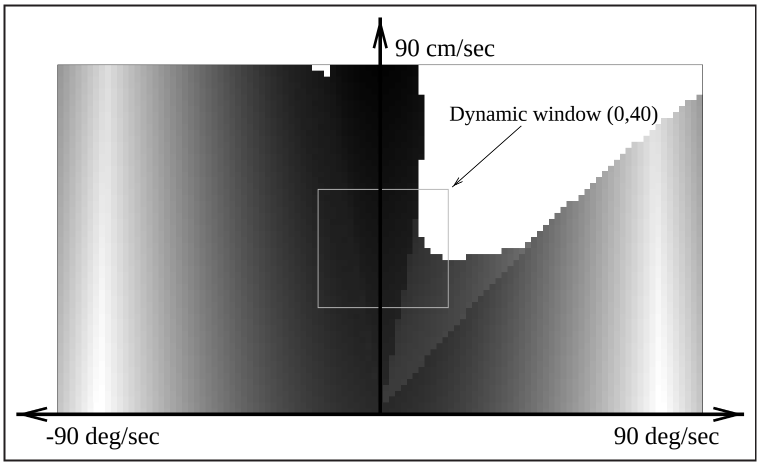


Fig. 5.10. Combined evaluation of the velocities and dynamic window if the robot is in straight motion (40 cm/sec).

Here the weights were set to $\alpha = 4.5$, $\beta = 0.5$, and $\gamma = 0.8$. The influence of different settings on the behavior of the robot is discussed in [FBT97].

In our implementation the objective function is maximized over the safe velocities in the dynamic window. This is done by discretization of V_a. Figure 5.10 highlights a dynamic window if the robot is in a straight motion of 40 cm/sec. A "zoom" into this area is plotted in Figure 5.11 (a) (note that the colors are rescaled). Due to the fact that the target point is in front of the robot, straight motion is preferred. Thus the fastest straight velocity maximizes the objective function (marked by the cross).

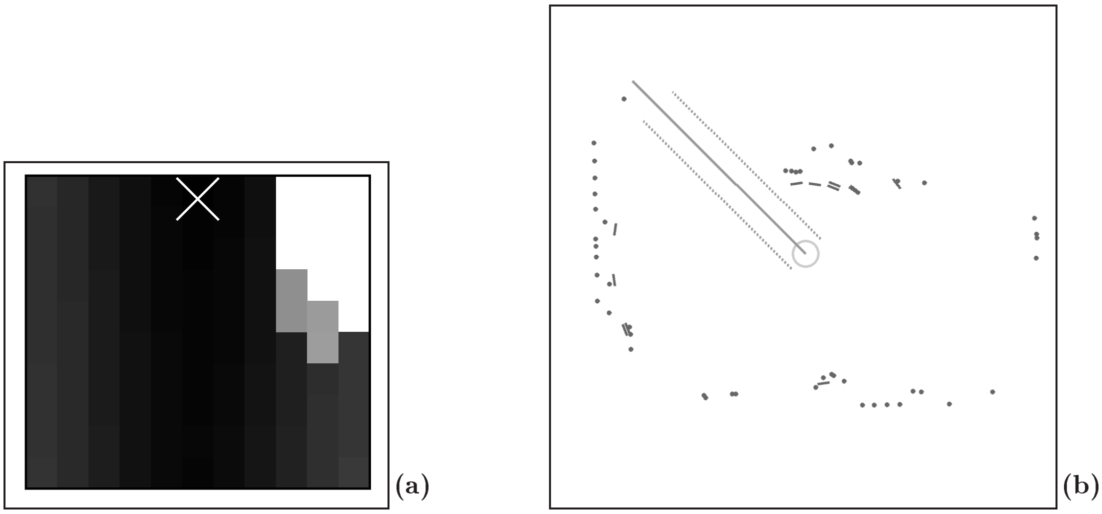


Fig. 5.11. (a) Evaluation of velocities within the dynamic window and (b) trajectory maximizing the objective function.

Figure 5.11 (b) shows the robot within the obstacle line field and the trajectory (dotted line) chosen according to the objective function. The direction to the target location is indicated by the solid line.

It should be noticed that all three components of G, the target heading, the clearance and the velocity are necessary. By maximizing solely the clearance and the velocity, the robot would always travel into free space but there would be no incentive to move towards a goal location. By solely maximizing the target heading the robot quickly would get stopped by the first obstacle that blocks its way, unable to move around it. By combining all three components, the robot circumvents collisions as fast as it can under the constraints and preferences listed above, while still making progress towards reaching its goal[3]. Further discussion of implementational details and examples for the importance of the dynamic window can be found in [FBT97].

5.4 Model-based Proximity Sensor

After having introduced our purely sensor-based method to reactive collision avoidance we now describe our approach to integrating model-based information into such a reactive technique. As mentioned above, such an integration is essential for safe navigation if not all obstacles can be detected by the robot's sensors. The key idea of our model-based

[3]It should be noted that due to the local nature of the approach, the robot still might get stuck in specific obstacle configurations

dynamic window approach (μDWA) is the generation of "virtual" sensor data, derived from a map of the environment and the position of the robot. To estimate this position we apply our grid-based implementation of Markov localization. If the robot knew its position with absolute certainty, integrating maps into sensor-based collision avoidance would be straightforward. Markov localization, however, often produces a multi-modal distribution over the set of possible locations. This raises the important question, how to synthesize the "virtual" sensor data under uncertainty in position. On the one hand, one wants to minimize the risk of colliding with an (invisible) obstacle and, on the other hand, one wants to maximize the robot's freedom, even when it is uncertain as to where in the world it is.

5.4.1 Probabilistic Description of the Virtual Sensor

In this section, we introduce a probabilistic representation of a *perfect* virtual sensor "mounted" at angle α relative to the robot's coordinate system. Let $d_\alpha(l)$ denote the distance to the nearest obstacle when the robot's position (in x-y-θ space) is l. $d_\alpha(l)$ can be computed using a map (see Section 2.6.1). Furthermore, let X_α denote a random variable that models a measurement of such a virtual sensor. If l is given, then the probability $P(X_\alpha = d \mid l)$ that this sensor returns a value $d \geq 0$ is normally distributed with mean at distance $d_\alpha(l)$. The standard deviation of this distribution depends on the accuracy of the world model and the discretization of the location l of the robot.

$P(X_\alpha = d \mid l)$ assumes knowledge of the robot's position. Suppose one is given a belief $Bel(L)$ about its current position. Then, the probability that the sensor returns a value d is given by Eq. (5.4).

$$P(X_\alpha = d) \quad = \quad \sum_l P(X_\alpha = d \mid l) \; Bel(L = l) \tag{5.4}$$

Figure 5.12 (a) and Figure 5.12 (b) depict two different position probability distributions in the museum environment: One, in which the robot does not know its position well, and one where it is highly certain about its position.

Figure 5.13 depicts the probability density of the measurement X_α in these two situations (imagine the robot being on the left side of the plot). As can be seen easily, when the robot is uncertain about its position, X_α is spread over many different measurements (solid line). If the robot knows its position well, the distribution of X_α is centered around a single distance (dashed line). In both cases, however, the robot assigns non-zero likelihood to extremely short measurements, since Markov localization never excludes a position with absolute certainty.

5.4.2 Selecting a Measurement

Once we have generated the distribution $P(X_\alpha)$ of the virtual sensor according to the current belief of the robot, we have to select a certain measurement to be provided to

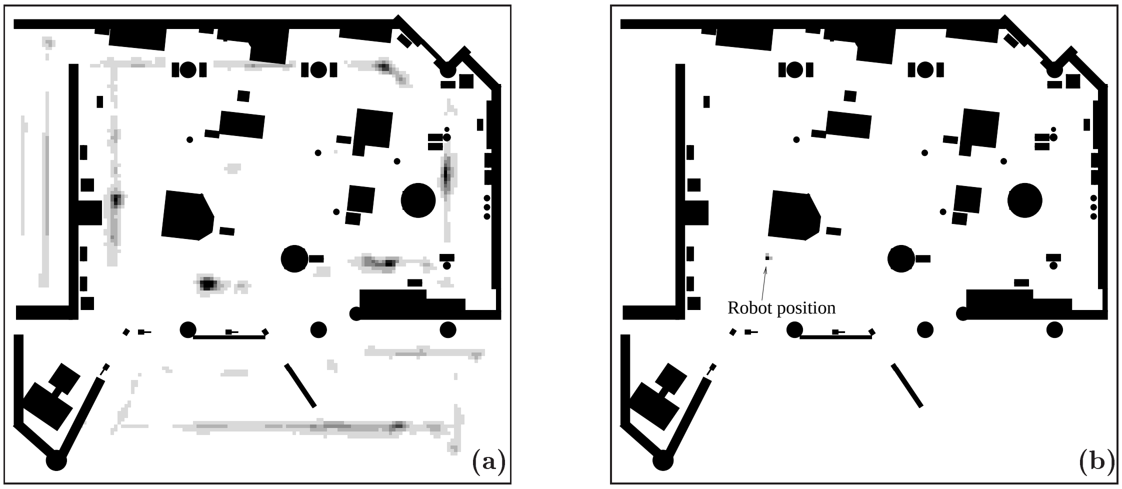


Fig. 5.12. Map of the museum (black) along with position probabilities (grey) (a) during global localization and (b) when the robot is highly certain.

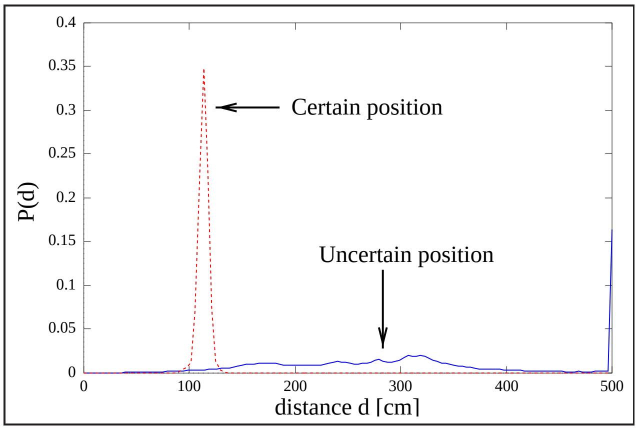


Fig. 5.13. Typical densities $P(X_\alpha = d)$ of a virtual sensor.

the collision avoidance. A straightforward selection scheme would be to pick a distance according to the distribution of X_α. Unfortunately, this would result in frequently picking too long readings especially when the position estimation is uncertain such as in Figure 5.13. In contrast to this, μDWA selects the "virtual" measurements using a conservative rule:

*The virtual measurement of a sensor is the largest distance d^*, such that with probability θ a distance **larger** than d^* is measured*[4]:

$$d^* \quad = \quad \max\{d \mid P(X_\alpha > d) \geq \theta\} \tag{5.5}$$

[4]The threshold θ is set to 0.99 in all our experiments.

Here, the probability that the sensor returns – according to its distribution – a value larger than d is given by

$$P(X_\alpha > d) \;=\; \sum_{d_i > d} P(X_\alpha = d_i). \tag{5.6}$$

Figure 5.14 (a) and Figure 5.14 (b) depict the example distributions of X_α as dashed lines. The corresponding values $P(X_\alpha > d)$ are plotted as solid lines.

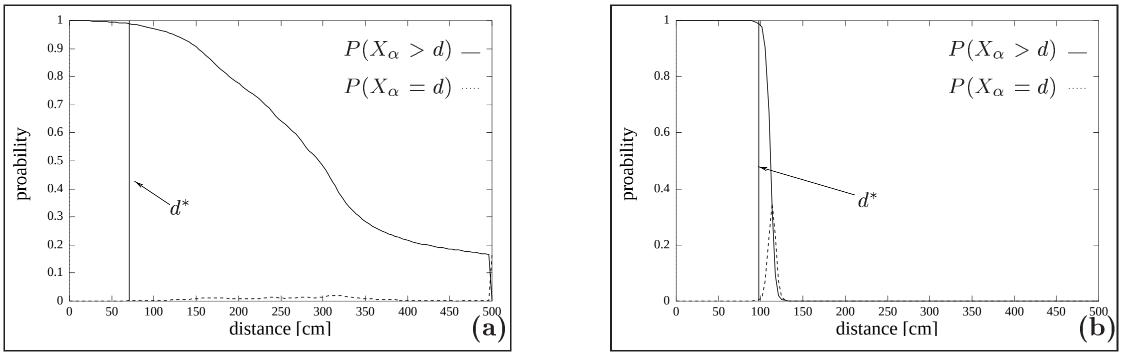


Fig. 5.14. Virtual measurement d^* if the position is known with (a) low and (b) high certainty.

The vertical lines illustrate the resulting distances d^* according to our selection scheme. Now the nature of this conservative scheme becomes clear: Even though in this example the mean of the distribution $P(X_\alpha)$ is *bigger* if the robot is uncertain about its position, this uncertainty yields the selection of a *shorter* distance d^* than if the robot is certain about its position.

By conservatively picking a sensor value that, with high probability, is shorter than the proximity of the obstacle, the robot is likely to avoid a collision even in the face of uncertainty. Notice that our approach provides maximum freedom under the constraints of 1% error probability. Another alternative scheme, such as picking the minimum distance among those locations l whose likelihood is above a certain threshold is *not* guaranteed to yield the same probabilistic bound in the likelihood of failure.

5.4.3 Application to the Example Situation

Let us reconsider the example situation used in Section 5.3. Figure 5.15 (a) depicts the position of the robot and Figure 5.15 (b) represents the corresponding obstacle line field including both real (grey) and virtual (black) sensor measurements (c.f. 5.2).

The evaluation of the curvatures in this situation is given in Figure 5.16. When comparing this figure with Figure 5.10 the influence of the virtual measurements becomes clear: By detecting that the obstacle to the right of the robot is closer than indicated by the real

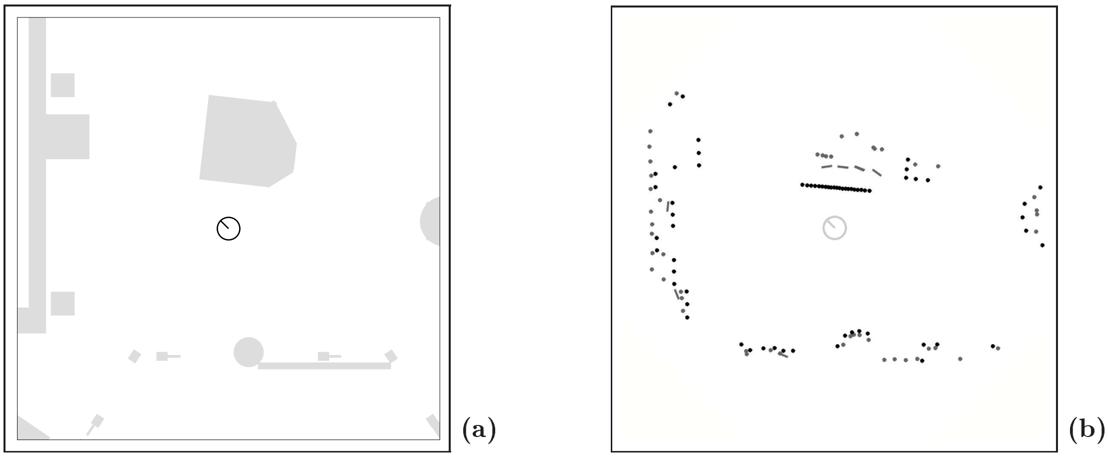


Fig. 5.15. (a) Position of the robot and (b) corresponding obstacle line field including virtual measurements.

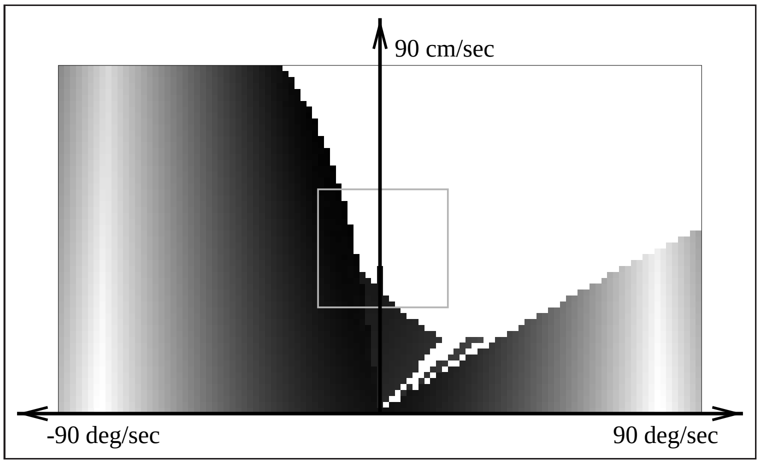
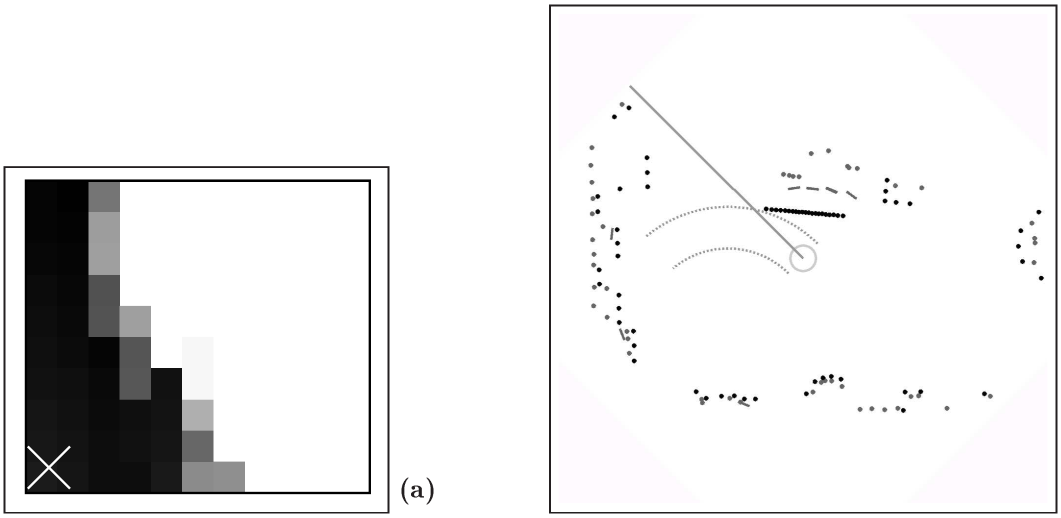


Fig. 5.16. Evaluation of the velocities when considering virtual measurements.

sensor measurements, the area of unsafe curvatures is much larger than by considering the real measurements only (c.f. 5.10).

The evaluation of the curvatures in the dynamic window is plotted in Figure 5.17 (a). Please note that the colors are rescaled to illustrate the different evaluations in the dynamic window. As can be seen here, a major fraction of the velocities are not admissible and according to the close obstacle to the right of the robot, the curvature which results in maximal deceleration and maximal rotation to the left is preferred (see cross in Figure 5.17 (a)). The resulting trajectory is depicted in Figure 5.17 (b).

Please notice that without considering the virtual sensor the robot would have chosen to move on a straight trajectory (see Figure 5.11 (b)) which would certainly have resulted in a collision with the obstacle.



(a) (b)

Fig. 5.17. (a) Evaluation of velocities within the dynamic window and (b) trajectory maximizing the objective function.

5.4.4 Efficient Implementation

For reactive collision avoidance a scan of the virtual sensor has to be generated frequently (e.g. 90 measurements every 50cm of robot motion). In our implementation of μDWA we have modified the basic code to fulfill this task efficiently enough. The most important modification concerns the computation of the density of a measurement X_α: instead of integrating over *all* locations l in Eq. (5.4), only a subset of all possible locations is considered, namely those with probability above a threshold. The threshold is set such that these cells in most cases represent more than 99% of the position probabilities. Our simplification is somewhat justified by the observation that in practice, Bel is usually quickly centered on a small number of hypotheses and approximately zero everywhere else.

5.5 Experimental Results

Based on the dynamic window approach to collision avoidance, different RWI B21 robots have been operated safely in various environments over the last four years. The maximum velocity of such robots is constrained to approximately 95 cm/sec. This velocity is usually reached in large openings and hallways, if no obstacles block the robot's way. If obstacles block its way, slower velocities are selected, and collisions are avoided by selecting appropriate trajectories. For example, the robot typically decelerates to approximately 20cm per second when moving through doors. In this section, we will give experimental results of our hybrid approach. See [FBT97] for more experiments illustrating the performance of the purely sensor-based component of μDWA.

5.5.1 Importance of the Dynamics

The first experiment demonstrates the influence of the dynamic window on the behavior of the robot. Consider the situation given in Figure 5.18 and suppose that the robot is in a fast straight motion while the target point is in the small opening to its right. Obviously, the optimal target direction implies a turn to the right. But ignoring that its forces are not high enough to perform the necessary sharp turn, the robot could collide with the right wall. The dynamic window approach is especially designed to handle such situations.

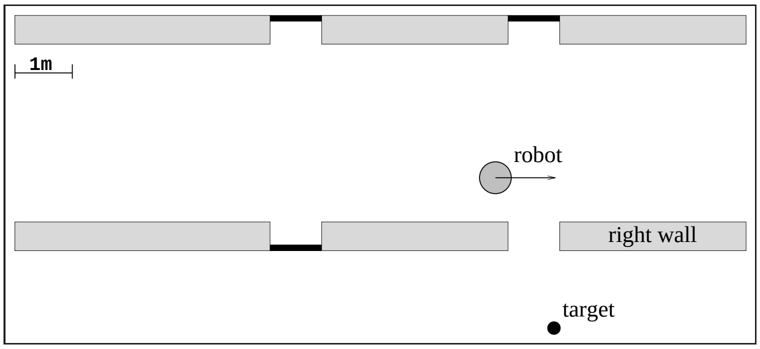


Fig. 5.18. Example situation with robot in straight motion.

The three paths in Figure 5.19 are examples for the typical behavior of the robot under different dynamics constraints. The crucial point of the experiment is the path taken in the dark shaded *decision area*. In this area the robot detects the opening to its right and has to decide whether to perform the sharp turn to the right or not. This decision strongly depends on the dynamics of the robot. Only if the actual velocity of 40 cm/sec and the possible accelerations (50 cm/sec^2, 60 deg/sec^2) allow a sharp turn to the right, the robot directly moves to the target point. This trajectory is represented by the dashed line. In the other two cases the robot decides to pass the opening and moves parallel to the wall until the evaluation of the target heading angle(v, ω) becomes very small.

Notice that without considering the dynamics constraints, an attempt to turn right would have almost certainly resulted in a collision with a wall. In fact, in initial experiments with a simulator, in which we ignored some of the dynamics effects, we experienced these type collisions frequently.

5.5.2 Museum Tour-Guide Project

A preliminary version of μDWA was tested extensively in the Deutsches Museum Bonn. Safe navigation was of utmost importance, since RHINO is strong and heavy enough to damage exhibits, some of which have an extremely high value. In order to guarantee this

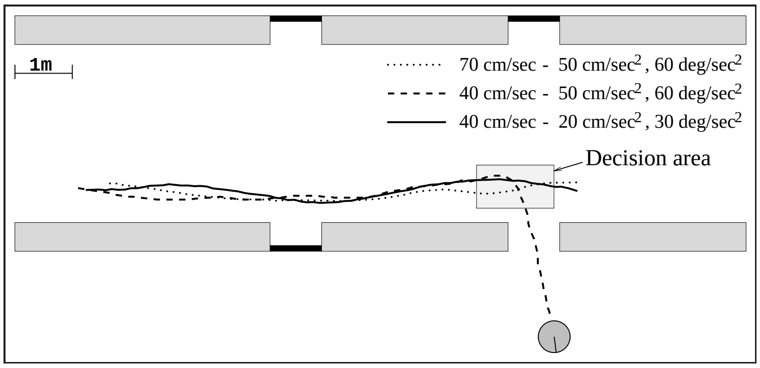


Fig. 5.19. Example paths of the robot chosen with different dynamics constraints.

safety, the robot only operated if its position was known uniquely (this was the case during the whole experiment). Thus, we parameterized μDWA so that it was equivalent to a technique which only considers the most likely position.

Several factors contributed to the difficulty of safe navigation in the museum:

1. **Invisible obstacles:** The virtual map sensor was extremely important because various obstacles were basically "invisible" to the robot, despite the fact that our robot applied four different sensor systems for obstacle detection (c.f. 5.1).

2. **Speed requirements:** To be "interesting" to people, the robot had to navigate at walking speed.

3. **Dynamic obstacles:** Large crowds often blocked much of the free space, and they often challenged the robot in various ways. Operating on a pre-planned, static path was not feasible. Instead, the robot had to continuously assess the situation and plan its motion accordingly.

4. **Accurate Localization:** In some narrow passages the robot had to closely approach invisible obstacles. Especially in these areas, accurate estimation of the robot's position was important for μDWA to allow the robot to move through such passages[5].

Avoiding collisions was clearly more difficult than in any of the various office environments in which our software was previously developed and tested.

During a total of 47 hours within six days of robot navigation, μDWA proved to be highly reliable, and was clearly essential for the success of the entire system. Important aspects of the project regarding collision avoidance are summarized in Table 5.1: In the six days of the museum tour-guide project RHINO traveled more than 18.6 km. When

[5] Results pertaining accurate localization are summarized in Section 3.6

Hours of operation	47
Number of visitors	> 2000
Kilometers traveled	18.6
Maximal speed of travel	> 80 cm/sec
Average speed during motion	36.6
Number of collisions	6
Number of requests	2400
Percentage of completed requests	99.75

Tab. 5.1: Some key figures from the museum tour-guide project.

the robot was not explaining an exhibit, it moved at an average speed of 36.6 cm/sec. In crowded situations, the average velocity was often lower; however, at times the robot moved at speeds of 70 cm/sec or higher for extended durations of time. More than 2,000 visitors were guided by RHINO. RHINO fulfilled 2,400 tour requests by real and virtual visitors of the museum.

The success-rate of 99.75% of this project clearly demonstrates the reliability of μDWA. During the entire project, the robot never collided with any of the visitors. We counted a total of six collisions with exhibits in the museum, all of which were minor and none of which caused any damage. Out of those six collisions, only one was directly related to μDWA: Here an "invisible" obstacle was approximately 20cm closer than our localization technique and thus μDWA had determined, causing the robot to touch the metal platform of one exhibit. Three other collisions were caused by hardware problems (such as low battery power). The remaining two collisions were caused by flaws in the hand-crafted map, which initially lacked some essential obstacle information.

5.5.3 The Role of Probabilistic Integration

A key aspect of μDWA is its ability to generate virtual sensor readings even if the robot does not know where it is (c.f. 5.12 (a) and Eq. (5.4)). To illustrate the importance of considering the entire belief Bel instead of just a single estimate, we empirically compared μDWA to an approach which only considers the *most likely* robot position ($\text{argmax}_l\, Bel(L = l)$) to generate virtual sensor readings. This approach can be thought of as the logical counterpart of μDWA if the localization component is not probabilistic and just maintains a single estimate.

Our experiments indicate that μDWA's integration is safer when the robot is not certain about its location. Figure 5.20 shows a map of a hallway in our university building along with position probabilities. In this example the robot started with complete ignorance about its starting position and due to the symmetry of the corridor the robot has not been able to uniquely determine its location.

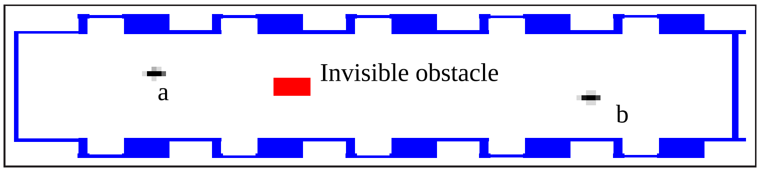


Fig. 5.20. Ambiguous situation in a corridor.

Consider the situation when the robot is truly at the location labeled a facing right but assigns slightly higher probability to the location labeled b facing left. Here the advantage of μDWA is obvious: While the maximum likelihood approach considers exclusively location b when generating the virtual sensor readings, μDWA takes both potential locations into account, thus picking the most conservative virtual sensor reading.

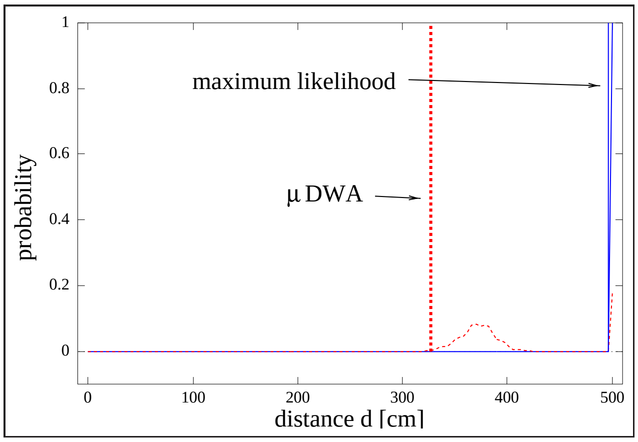


Fig. 5.21. Distribution $P(X_\alpha = d)$ of a virtual sensor using probabilistic integration (μDWA) or the most likely position only.

To see, consider the Figure 5.21 where $P(X_\alpha = d)$ is plotted for the virtual sensor pointing to the invisible obstacle in the corridor. Here the dashed line shows $P(X_\alpha = d)$ when averaged over all locations (as in μDWA), whereas the solid line shows the distribution based on the most likely estimate only. As a result, the maximum likelihood approach will falsely generate a long reading (max range in this case) as indicated by the vertical line, whereas μDWA will generate a reading that prevents the robot from colliding. For μDWA to err, the robot has to assign less than 2% probability to the correct location.

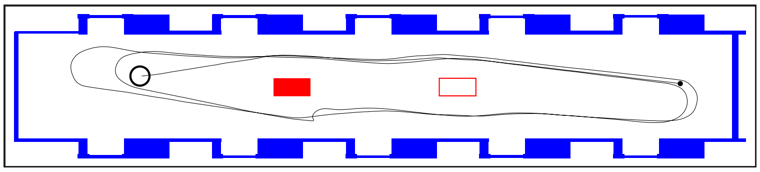


Fig. 5.22. Path of the robot.

Figure 5.22 shows a path taken by the robot using our hybrid approach to collision avoidance. Again, the robot was placed in the corridor without knowing its starting position. The target locations for μDWA were either on the right or on the left side of the corridor. The reader may notice that the robot not only successfully circumvented the real invisible obstacle (filled) but due to uncertainty in its position it also avoided the symmetric position of this obstacle marked by the non-filled rectangle.

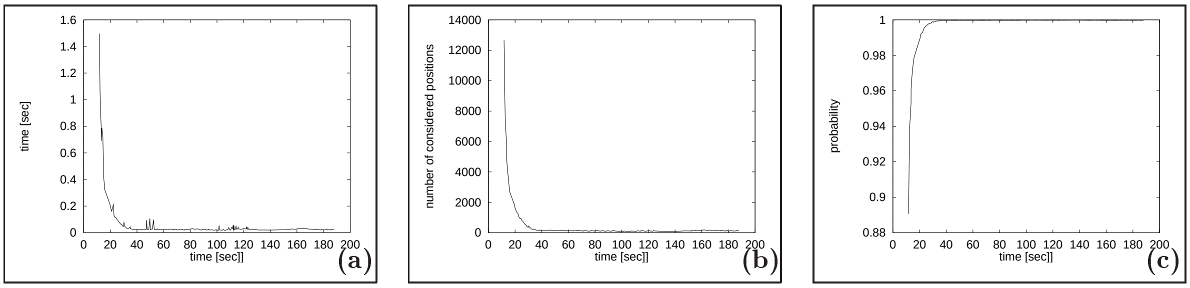


Fig. 5.23. (a) Time required to generate a scan of the virtual sensor, (b) number of positions considered, and (c) summed probability of considered positions.

Figure 5.23 (a) - (c) illustrate important aspects of our efficient implementation of μDWA during this experiment. The experiment took about 200 seconds and the x-axis in the figures represents the time from the beginning of the experiment. Figure 5.23 (a) depicts the time necessary to compute a scan including 90 readings of the virtual sensor. Even in the beginning as the robot is extremely uncertain about its position, a scan can be computed in less than 1.5 seconds. As the certainty in the position grows the time for determining a scan becomes almost neglectable (less than 0.1 secs). This is due to the fact that we only consider locations l with probability above a certain threshold for computing the virtual sensor according to Eq. (5.4). In this experiment the threshold was set to 1e-5. The number of considered locations is plotted in Figure 5.23 (b). As can be seen here, this number decreases as the robot updates its position estimation and quickly converges to a number of less than 100 locations. Figure 5.23 (c) depicts the summed probability of the considered locations. It should be noted that even though the number of these locations decreases, the summed probability of those positions converges almost to 1.0 after less than 40 seconds.

5.6 Discussion

In this chapter we described a hybrid approach to collision avoidance, called μDWA, which integrates both sensor data and data from a previously supplied map into collision avoidance. The key idea of our technique is to generate virtual sensor readings based on a map of the environment. The position of the robot relative to the map is estimated using our implementation of position probability grids. The virtual measurements are generated using a conservative rule which ensures that the robot avoids collisions with high likelihood even if it does not know its position. Based on the dynamic window approach, μDWA differs from most previous techniques in that it

1. considers the *dynamics* of the robot and
2. avoids collisions with invisible obstacles even if the robot is uncertain about its position.

This approach combines the best of both worlds: it reacts adequately to unexpected obstacles (such as humans), but it also avoids collisions with undetectable obstacles whose locations are known. The reader may notice that by generating virtual sensor measurements, our approach to integrating model-based information into reactive collision avoidance can be applied to most existing purely sensor-based approaches to collision avoidance such as e.g. [Kha86; KC95; BK90; BK91; Sim96]. μDWA has been tested extensively in various densely populated environments (including a museum), in which a large number of obstacles (exhibits) could not be detected with the robot's sensors.

This technique may have significant impact on future low-cost robot applications. The ability to integrate map-based information into collision avoidance, even if the robot is not certain about its actual location, reduces some of the burden to equip robots with potentially high-cost sensors. For example, in (ongoing) experiments using data recorded in the museum, we have found evidence that the grid-based *position tracking* technique described in [BFH97; Hen97] would have been able to track the robot's location based solely on the sonar sensors, and thus would be sufficient for model-based collision avoidance. Such a finding suggests the feasibility of installing robots that only use sonar sensors (instead of the much more expensive laser sensors) in environments as complex and unstructured as this specific museum. Hybrid approaches to collision avoidance, which react to sensor readings but also consider models of the environment, have not received much attention in the literature. However, we believe that many environments require such hybrid approaches, of which the Deutsches Museum is certainly one.

The current approach also suffers limitations, which mainly arise from the need of an accurate metric map. In the particular experiments reported here, the map was constructed manually, using measuring tape. In most robot applications such an approach is justifiable by the fact that the installation costs (i.e., acquiring a map) are small compared to the day-to-day operational costs. However, in some environments, such as private homes, the need for a metric map might make it difficult to apply the method described here. Recent research on map acquisition [Thr99b; TGF$^+$98; LM97] provides a way to acquire the map autonomously.

Of course, methods for map acquisition model only those obstacles that can be detected by the robot's sensors. While the real-time requirements for reactive collision avoidance prohibit the use of time-consuming sensor-interpretation techniques such as complex visual scene interpretation, such techniques can be applied to improve the world model in order to include obstacles not detectable by the proximity sensors of the robot. Additionally it appears to be feasible, however, to "label" either undetectable obstacles or even forbidden areas by driving the robot (manually) along the boundary of its legal operational space. With such an approach, μDWA should be applicable even in many domains where up-front map information is unavailable.

Chapter 6

Integration of Localization into Robot Control

6.1 Introduction

To successfully perform a variety of different tasks, mobile robots must reliably fulfill and coordinate different kinds of subtasks like navigation, manipulation, and perception. Many service robots employ high-level control systems, which control, coordinate, and monitor these subtasks during execution to ensure the robot's operational effectiveness [Nil84; Gat92; TBB+98; SGH+; KMRS97]. In this chapter we focus on the aspect of the integration of localization into such robot control systems. In this respect, passive approaches to position estimation have the key advantage that they do not interfere with the robot's main tasks such as e.g. office delivery. Therefore, integration of passive localization into robot control is straightforward. Active localization, on the other hand, can be much more efficient at the cost of possibly interfering with the robot's primary tasks. Thus, the integration of active approaches into robot control systems becomes an important aspect of mobile robot localization.

In this chapter we will introduce our approach to integrating localization into such a control system. In order to minimize the interference between the robot's primary tasks and active localization, we only hand over the control to the active component if the robot's position is lost[1]. Otherwise, the robot performs its tasks and the position is tracked passively. The remainder of this chapter is organized as follows. After discussing related work, we will give a brief overview of RHINO's software architecture in Section 6.3 (see e.g. [TBB+98; BCF+98b] for detailed descriptions of the architecture). The integration of localization into robot control is addressed in Section 6.4 followed by a discussion.

[1] In our current implementation such a situation is detected if the probability of the global maximum (see Section 2.6.3) falls below the threshold of 0.9.

6.2 Related Work

Most existing approaches to position estimation are passive. Hence, the integration of active localization into robot control systems has only received considerably small attention in the literature. [GCJK97] detect situations in which the position of the robot has been lost. Whereas their approach relies on the user to specify the robot's current position to recover from such situations, our method is able to autonomously relocalize the robot. In the field of Markov localization and path planning, [NPB95; SGH+; KCK96] deal with the problem of selecting appropriate actions under position uncertainty. [NPB95] choose the shortest navigation path with respect to the most likely position of the robot. [SGH+] introduced several decision-theoretic heuristics, which take into account the uncertainty of the position estimation when planning a path to a target location. Both approaches do not consider active localization, i.e. they do not introduce actions dedicated for reducing uncertainty in position estimation. The approach introduced in [KCK96] takes uncertainty and actions for relocalization into account. They do not distinguish between the robot's primary tasks and localization actions and always choose the actions with the highest utility with respect to the belief state of the robot. While this technique is optimal with respect to the modeled quantities and the planning horizon, its applicability is limited due to the computational complexity of the approach. In contrast to these methods, our approach only considers the most likely position for planning. Whenever the certainty of the position estimation falls below a specified threshold, active relocalization is performed. Although this might not lead to the optimal behavior, extended experiments showed the reliability of this approach. We furthermore experienced, that possible additional costs are negligible during long-term operation, because our localization technique failed very rarely.

6.3 Software Architecture of the RHINO System

In this section we will give an informal overview of the RHINO software system. For more detailed information the reader is referred to e.g. [BCF+98b; TBB+98]. The overall software architecture consists of approximately 25 modules (processes), which are executed concurrently on 3 on-board PCs and 2 off-board SUN workstations, connected via Ethernet. The software modules communicate using TCX [Fed93], a decentralized communication protocol for point-to-point socket communication. Figure 6.1 shows the architecture along with the major software modules and the flow of information between them. Similar to other robot control architectures, the RHINO system is also organized in a hierarchical manner, with the device drivers at the lowest level and the user interfaces at the highest. The hierarchy, however, is not strict in the sense that modules would pass information only within the same or across adjacent layers. In RHINO's software, modules often communicate across multiple layers.

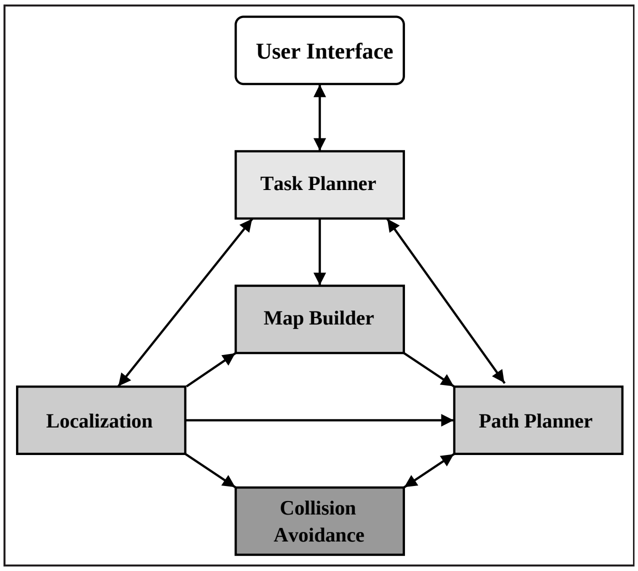


Fig. 6.1. Architecture of the RHINO control system and major flow of information.

Before we describe the different modules of the architecture, it should be noted that all sensor information collected on the robot is broadcasted to the other modules, if required. This sensor data consists mainly of the position of the robot measured by the odometry and measurements of the robot's proximity sensors (especially sonar and laser sensors). In the following, we will describe the functionality of the different modules in the normal operational mode. Specific tasks such as e.g. building a map from scratch, are described in the related literature.

Collision Avoidance

At the lowest level of the architecture, the reactive collision avoidance module is responsible for safely navigating the robot around obstacles on its path. As described in Chapter 5, our model-based dynamic window approach integrates real and virtual sensor data in order to avoid collisions even with obstacles not detectable by RHINO's sensors. The target points for the collision avoidance are frequently updated by the path planner described in turn. Please note that due to updates in the world model, the position of these target points can change drastically from one second to the other. The dynamic window approach is especially designed to handle such situations. It considers the dynamics of the robot when generating motion commands, and therefore it causes no problems to send e.g. target points behind the robot even if it is in a fast straight motion.

Localization

The RHINO system applies position probability grids described in this thesis to localize the robot. After each update based on new sensor information, the belief state is scanned for local maxima as described in Section 2.6.3. Since in most cases the position is uniquely determined, the global maximum of the belief state is assumed to be the true location of the robot (see Section 6.4 for a discussion of the situation, when the robot is uncertain about its location). The estimated position of the robot is frequently sent to the other modules.

In our current implementation the localization module additionally serves as the generator of virtual sensor readings used in the hybrid collision avoidance module (see Section 5.4). This integration is necessary since the computation of virtual measurements according to Eq. (5.4) strongly depends on the belief $Bel(L)$ of the position estimation.

Path Planner

The collision avoidance module is prone to get stuck in e.g. U-shaped obstacle configurations, because it only considers a small fraction of the environment when generating motion commands. Therefore, it is necessary for efficient navigation to guide the collision avoidance based on a model of the entire environment. RHINO's path planner uses occupancy grid maps as world model to generate minimum-cost paths to target locations. Given a goal point, the estimated cost for reaching this target are computed for each position in the environment. These cost are computed based on a deterministic version of value iteration. Here, actions of the robot are modeled deterministically and the cost for traversing a grid cell are assumed to be proportional to the occupancy value of the cell. Whenever the map of the environment changes due to new sensor information, the cost for reaching the target location are updated as well. Path planning is based on the assumption that the position of the robot is determined uniquely. Thus, the minimum-cost path to the target location is generated by steepest descent in the costs starting at the estimated robot position. Intermediate target points are extracted from this path based on a visibility criterion and are sent to the collision avoidance module.

An example for path planning in the museum environment is given in Figure 6.2. Here, the target location is on the right part of the map and the corresponding cost increase with the distance to this target. The figure also depicts the minimum cost path for a specific robot position. From this path, an intermediate target location is extracted and sent to the collision avoidance module. In this example, the location marked with the `X` is chosen. See [Thr99b; BCF$^+$98b] for more details on the path planner.

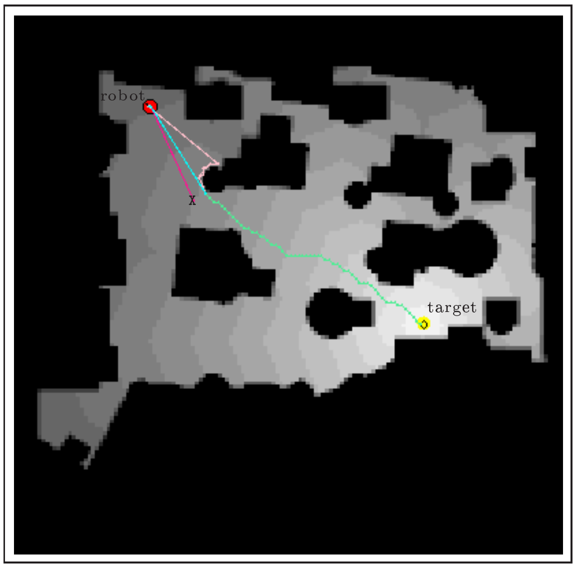


Fig. 6.2. Cost for reaching the target location determined by value iteration (dark shading indicates high cost) and the resulting minimum-cost path from the estimated robot location.

Map Builder

In the normal operational mode, the map builder is used to update a static model of the environment. This module generates occupancy grid maps (see [Elf87; ME85; Thr99b]) to estimate the occupancy of each location in the environment. Each scan of the robot's proximity sensors is converted into a local occupancy grid map. The map builder is able to integrate maps based on different kinds of sensors into a previously supplied map. Whenever the map changes due to new sensor information, the updated model is sent to the path planner, so that the robot can adequately react to changes in the environment.

Figure 6.3 illustrates a situation during the museum tour-guide project, where the robot decided to take a detour around a crowd of people in order to reach its goal. Here, a CAD map of the museum is integrated with maps based on sonar and laser measurements.

Consistent integration of local maps into a global world model requires knowledge of the position of the robot. If a map of the environment is already provided, then this position can be estimated by our localization module. Approaches for concurrently building the map from scratch and estimating the position of the robot are described in [Thr99b; TFB98; TGF+98].

Task Planner

The fundamental task of this module is to ensure that the robot can perform its missions reliably and efficiently by generating plans and monitoring their execution. These missions

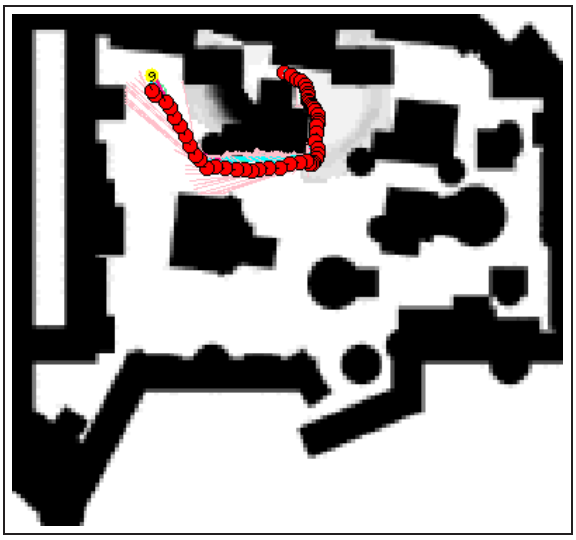


Fig. 6.3. Integrated map, acquired in a situation where the robot had to take a detour around a crowd of people.

generally include different kinds of subtasks such as navigation, interaction, perception, etc. In the RHINO system two different kinds of task planners are applied.

The first system is based on GOLOG, a logical language that represents knowledge in the situation calculus (see e.g. [LRL+97]). The gap between the logical representation and the low level modules of the software architecture is bridged by GOLEX [HBL98], which decomposes the high-level GOLOG actions into a macro-like sequence of appropriate directives and monitors their execution. This system has been successfully applied during the museum tour-guide project. The second task planner is implemented in RPL (Reactive Plan Language). This module is applied to integrate active localization into robot control and will be discussed in Section 6.4.

User Interface

RHINO's user interface is especially designed for the museum tour-guide application. In the museum, RHINO applied two user interfaces, one on-board interface to interact with people directly, and one on the Web. The on-board interface is a mixed-media interface that integrates graphics, sound, and motion. In the museum, the users were required to press buttons close to the on-board display to choose between different options (see Figure 6.4 (a)). During the tour the robot used its camera to point to the exhibits. It showed text and graphics on its display and played pre-recorded sounds from CD to explain the exhibits.

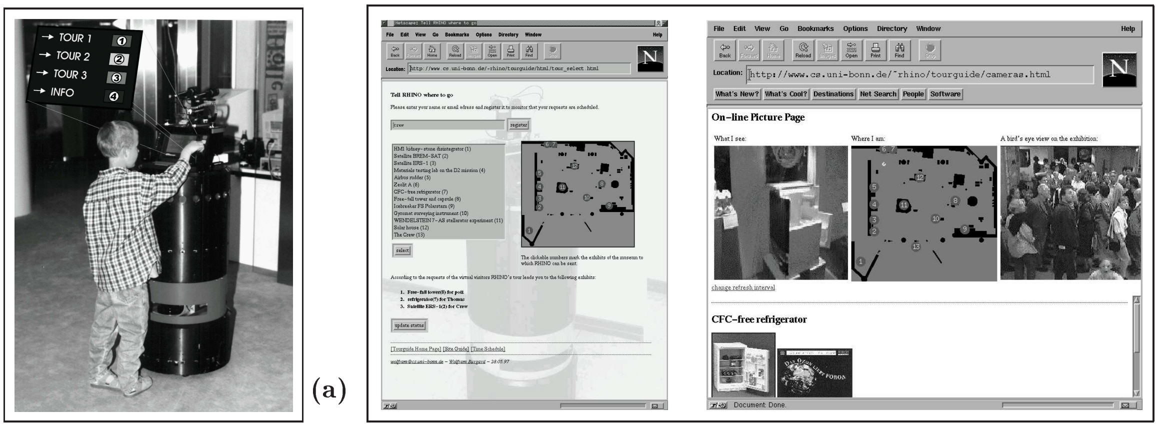


Fig. 6.4. (a) On-board interface of the tour-guide robot and (b) popular pages of the Web interface.

RHINO's Web interface consists of a collection of Web pages[2], which serves four main purposes: Monitoring, control, providing background information, and providing a discussion forum. The interface enabled remote users to control the robot (left image in Figure 6.4 (b)) and to observe it, along with the museum and the people therein (right image in Figure 6.4 (b)).

In the museum, an application of the distance filter introduced in Chapter 3 turned out to be one of the most attractive interaction features of the robot. The distance filter was used to detect unknown obstacles (mostly visitors) blocking the robot's path. In combination with a measure of progress, the robot decided to blow its horn whenever necessary. Many visitors (especially children) were amazed by the fact that the robot acknowledged their presence by blowing its horn, and repeatedly stepped in its way.

6.4 Integration of Active Localization

In the previous section we discussed the integration of localization in situations when the position of the robot is uniquely determined. In such cases, the passive version of position probability grids is able to efficiently track the position of the robot as it performs its tasks. As noted above, passive localization does not change the configuration of the robot and thus its integration into robot control is straightforward. However, we showed in Chapter 4, that the passive paradigm can be extremely inefficient if the position of the robot is not known. In such situations, actively controlling the robot so as to best localize it can drastically improve the efficiency of position estimation. Unfortunately, active localization may interfere with the robot's primary tasks. Therefore, the task planner has to synchronize localization with the other actions. In our approach, the high-level control system achieving

[2] See http://www.cs.uni-bonn.de/~rhino/tourguide/

this integration is implemented in RPL (Reactive Plan Language), a notation for specifying *structured reactive plans (SRPs)*. SRPs provide a means to synchronize concurrent activities, monitor aspects of the robot and its environment, locally recover from execution failures, and avoid sending inconsistent commands to sensors and effectors (see [BBCD97; BBDC98; McD91; McD92]). In this control system we distinguish actions performed to achieve the robot's primary tasks and actions for the purpose of localization. In order to minimize the impact of position estimation on the main tasks of a mobile robot, we propose a control paradigm that causes the robot to exhibit the behavior sketched in the following table.

while (tasks not finished)
    if (the robot knows its position sufficiently well)
        the position is tracked *passively* while the robot performs its tasks
    else
        interrupt task execution
        (re-) localize the robot *actively*
        continue task execution
    endIf
endWhile

Tab. 6.1. Integration of active localization into the control system.

Whenever the robot loses track of its position[3], the primary tasks are interrupted and control is handed over to the active localization module described in Chapter 4. To achieve maximal safety during the process of active localization, uncertainty in the robot's position is considered in two ways. First, the path planner uses an occupancy map which represents the position of obstacles under consideration of the current uncertainty in the robot's position (see e.g. Figure 4.8 (a) and Figure 4.11 (a) for such maps). Second, we apply our hybrid approach to collision avoidance to steer the robot during active navigation. As described in Chapter 5, this module is able to avoid obstacles not detectable by the robot's sensors even when the position of the robot is not uniquely determined. As soon as the robot is relocalized, it continues to execute its primary tasks.

An important precondition for our integration scheme is that the robot is able to interrupt its primary actions as soon as its position is lost. To achieve mission interruptability and continuation, primary actions include specifications of (sub-)actions that have

[3]In our implementation we detect such a situation, if the probability of the global maximum falls below the threshold of 0.9.

to be performed whenever the primary actions are interrupted and whenever they are continued after such an interruption. For example, the navigation process, responsible for computing the next target points and monitoring the progress towards the given target, generates a new plan whenever it is reactivated after a relocalization of the robot. In the next section we will present an experiment which illustrates this integration of active localization into robot control.

6.5 Experiments

To demonstrate the capabilities of our approach we performed several experiments within the context of an office delivery robot in our office environment. In all these experiments the task of the robot was to reach a location in front of a shelf in one room of our department. In order to force the localization module to lose track of the robot's position, we manually rotated the robot by about 70 degrees shortly after starting its mission. Figure 6.5 shows the initial high-level navigation plan generated by the structured reactive controller at the beginning of such an experiment. Here, the robot starts at location `1` and the target

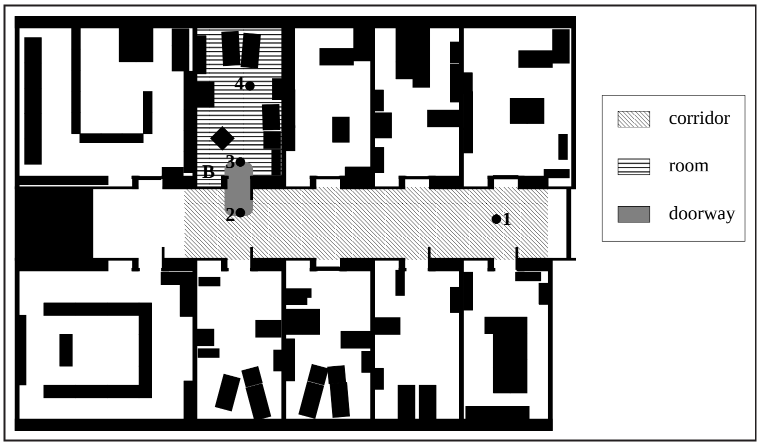


Fig. 6.5. Initial plan of the delivery robot

location is labeled `4`. The plan specifies in detail when and how the robot is to adapt its travel modes as it follows the navigation path. In many indoor environments it is advantageous to adapt the driving strategy to the surroundings: to drive carefully (and therefore slowly) within offices, to switch off the sonars when driving through doorways (to avoid crosstalk between the sonars), and to drive quickly in the hallways. The different travel modes "office," "hallway," and "doorway" are depicted through regions with different textures. Whenever the robot crosses the boundaries between regions the appropriate travel mode is set in our collision avoidance module.

Shortly after starting the execution of the navigation plan we manually turned the robot by 70° to the right. After this turn the robot was facing west while the position estimation as well as the path planner assumed the robot to face south-south-west. Consequently the path planner generated target points which caused the robot to turn right and move straight. Shortly after that, RHINO's reactive collision avoidance turned the robot to the left because of the northern wall of the corridor. While the solid line in Figure 6.6 (a) shows the real trajectory of the robot, the dashed line depicts the trajectory estimated by the position estimation module (note that this module didn't realize the manual change in orientation). At the end of this path the accumulated discrepancies between expected and measured sensor readings reduced the confidence in the position of the robot below the given threshold.

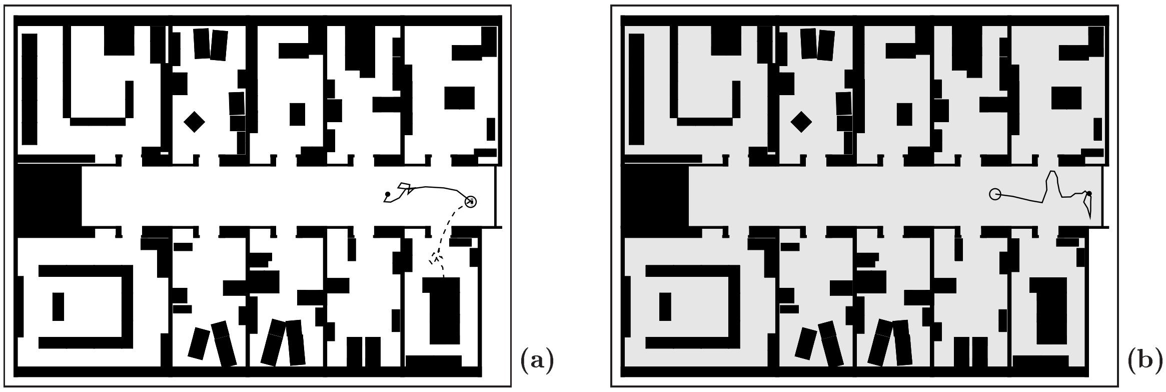


Fig. 6.6. Path of the robot (a) before active localization and (b) during initialization of the belief state at the beginning of active localization.

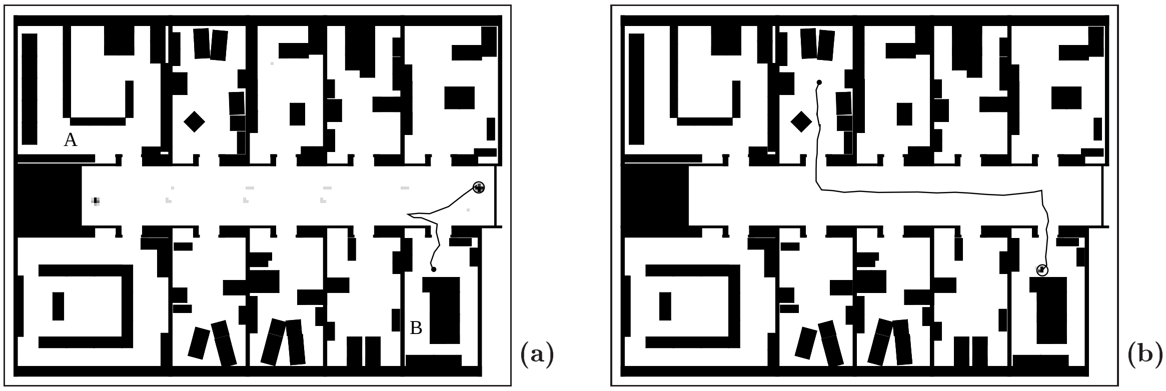


Fig. 6.7. (a) RHINO moves into room C for disambiguation and (b) continues to navigate to the target location.

At that point the robot control system interrupted the navigation process and started the active localization module. In this experiment the active localization module was parameterized to reinitialize the belief state once the position is lost[4]. The resulting uniform distribution over all possible positions in the environment is shown in Figure 6.6 (b). As discussed in Chapter 4, active localization starts with random motion in order to reduce the state space to several local maxima. Figure 6.6 (b) shows the path of the robot until the belief is reduced to 10 local maxima. At this point in time, the positions at the west end of the corridor facing east and the east end facing west were the most likely ones. The corresponding belief is depicted inFigure 6.7 (a). To disambiguate the two most likely positions, active localization decided to guide RHINO into the next room to its left (either room C or D). This target location was chosen because it is close to the robot and expected to maximally reduce the uncertainty of the position estimation. In this particular run the corresponding navigation action lead RHINO into room C, where in fact all ambiguities could be resolved. The corresponding belief is depicted in Figure 6.7 (b).

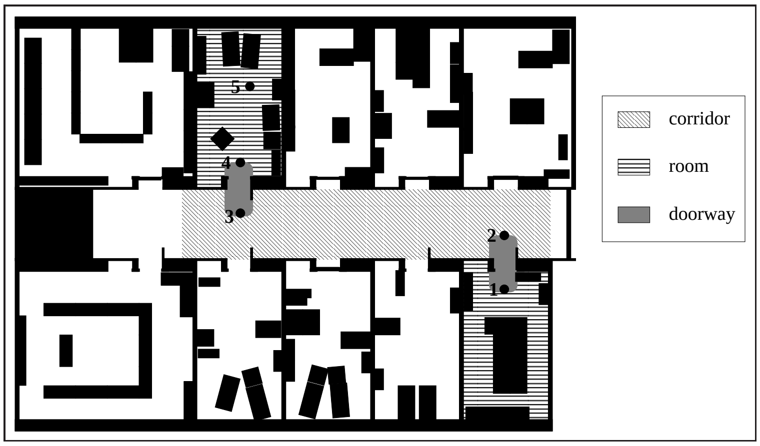


Fig. 6.8. Plan of the delivery robot after successful relocalization.

After successful relocalization, the control was given back to the navigation task which generated a new navigation plan to room B (see Figure 6.8). Figure 6.7 (b) also shows the complete trajectory taken by the robot during this experiment.

6.6 Discussion

In this chapter we introduced our integration of active localization into high-level robot control systems. The principle of this approach is to passively track the robot's position as it performs its tasks. In the normal operational mode, the position of the robot is assumed to be known uniquely, which makes especially planning more efficient. The certainty of the

[4]In most cases observed in this specific kind of experiments this turned out to speed up the process of re-localization.

position estimation is continuously monitored and active localization is invoked whenever the uncertainty grows too large. To realize this approach, active localization is treated as an *action* which possibly interferes with other actions of the robot. Thus, to ensure that the robot can reliably carry out its tasks, the high-level routines are implemented such that they provide a means for handling interrupts and continuing successfully after reactivation.

With this integration scheme, we exploit the following abilities of our extended implementation of position probability grids. They are able (1) to efficiently and accurately track the robot's position, (2) to detect situations in which the tracking has failed, and (3) to actively relocalize the robot in the case of such a failure. All these features are basic preconditions to establish mobile robots with a high degree of autonomy.

In our experience there have hardly been any cases in which the robot has lost its position when using our passive localization method. Thus, the expected average time costs for localization are extremely low. Furthermore, the time spent on localization is generally outweighed by the performance gain resulting from accurately knowing the robot's position. Despite these encouraging results there are several warrants for future research. Obviously, the efficiency of the system can be improved by adapting the position certainty measure according to the current tasks. As an example, consider the situation where the robot has to transport an object and dispose it anywhere in the hallway. In this case it suffices to know that the robot is in the hallway, but the exact location is not needed. To competently handle such problems, the control system has to predict the effect of position uncertainty on its course of action and react appropriately.

Chapter 7

Conclusions

In this dissertation we addressed several aspects of mobile robot localization based on position probability grids as a novel approach to estimating the position of a robot based on sensor data. Position probability grids are an implementation of Markov localization, which uses a general scheme for state estimation. Here, the state of interest is the position of the robot within its environment and Markov localization aims at representing arbitrary probability distributions over this state. A key advantage of this technique is that it can represent situations in which the position of a robot is not determined uniquely. Therefore, Markov localization can localize mobile robots *globally*, i.e. without knowledge of their initial position.

In contrast to the other implementations of Markov localization, position probability grids use a fine-grained grid-based discretization of the state space. Due to this representation, our implementation is able to integrate raw data from proximity sensors and is therefore able to exploit arbitrary geometric features of the environment. In order to deal with the huge state space that has to be maintained during position estimation, we introduced a probablilistic model of proximity sensors, which can be efficiently retrieved at runtime. In combination with a technique for selectively updating the belief state, position probability grids are able to globally localize a robot from scratch and efficiently track its location once the position is determined.

All techniques introduced in this thesis were validated extensively on the robot RHINO. The experiments were conducted in two different environments. One environment, a part of our comuter science department, includes all features of typical office environments such as people walking by, refurnishing of offices, and several places that look alike. The other environment is a crowded museum, in which RHINO served as an autonomous museum tour-guide robot for extended periods of time. This unstructured environment stressed additional requirements for robust localization such as accuracy and the ability to handle situations in which a major part of the robot's sensors is blocked by people.

Based on the general probabilistic representation used in our position probability grid approach, we were able to address the following additional questions in the context of mobile robot localization.

1. How can we attain robust position estimation even in dynamic environments?
2. How can we design a localization procedure, which allows a mobile robot to efficiently localize itself from scratch without the need of a human supervisor?
3. How can we achieve safe navigation during the process of global localization even in the presence of obstacles that cannot be detected by the robot's sensors?

In order to achieve the first requirement, we extended our approach to Markov localization by a technique which filters sensor data, so that the damaging effect of *dynamics* in the environment is reduced. In this context we proposed and evaluated three specific filters, one which considers conditional entropy for selecting sensor readings, and two which take into account additional knowledge about the effects of possible environment dynamics. The entropy filter was essential for successfully operating RHINO in a crowded museum for longer periods of time. Experimental comparisons using data collected there demonstrated that by processing sensor readings selectively, the filters especially designed for proximity sensors combine the robustness against corrupted sensor readings with the ability to recover from global failures in localization. Additional tests conducted in an office environment, gave evidence that our technique can estimate the position of a robot even if only an outline of the environment is used as a world model.

A solution to the second problem, namely the problem of efficient global localization, is a major precondition to operate mobile robots with a high degree of autonomy. Position probability grids are well suited to provide means on how to *actively* localize a mobile robot within its environment. In this thesis we applied the framework of decision-theoretic planning in order to guide a mobile robot to places which most likely help it to uniquely determine its location. In essence, actions are generated by minimizing the future expected uncertainty, measured by the entropy of the belief state. In combination with our model of proximity sensors, the active localization technique successfully discovers features helpful for localization such as e.g. furniture or the size of rooms. We additionally applied this principle to the problem of active sensing. Here our technique was able to select appropriate pointing directions for the different kinds of sensors.

Active localization raised the question of how to safely navigate a mobile robot even during the process of global localization. The installation of RHINO as a museum tour-guide robot showed, that a main problem with respect to safe navigation is the presence of obstacles that cannot be detected by the robot's sensors. In order to guarantee safe navigation even in the presence of such invisible obstacles, we introduced our hybrid approach to reactive collision avoidance, which integrates both sensor data and data from a previously supplied map. The key idea of our technique is to generate virtual sensor readings based on a map of the environment. The position of the robot relative to the map is estimated using our implementation of Markov localization. Here the complex representation of position probability grids enables a robot to avoid collisions with invisible obstacles even if the robot is uncertain about its position (which is the case especially during global localization).

The problem of position estimation cannot be treated independently from the other tasks of a mobile robot. Especially the integration of active localization into the control system of such a robot has to be addressed, since active localization can interfere with the robot's main tasks such as e.g. office delivery. In this thesis, we proposed the following integration scheme which exploits all advantages of position probability grids. The position of the robot is tracked accurately and efficiently as the robot performs its main tasks. At the same time, the certainty of the position estimation is continuously monitored and the robot is actively relocalized whenever the uncertainty grows too large.

In summary, we proposed a localization scheme which allows mobile robots to operate reliably and efficiently in populated and unstructured environments and thus realizes an important step towards truly autonomous mobile robots.

Appendix A

Denotations of Quantities for Models of Proximity Sensors

l A possible **l**ocation of the robot. In our experiments l is a square of size $15 \times 15\text{cm}^2$ with an angular resolution of 1-2°.

o_l Distance to the closest **o**bstacle in the map if the robot is at location l and fires a certain proximity sensor.

Δ_d Granularity of the discretization of distances (4cm in our implementation).

d_i **D**istance measured by a proximity sensor ($|d_i| = i \cdot \Delta_r$).

σ_m Standard deviation of the Gaussian distribution modeling the uncertainty in the distance o_l to the closest obstacle in the **m**ap if the robot is at location l.

σ_s Standard deviation of the Gaussian distribution modeling the uncertainty of a **s**ensor when detecting an obstacle.

σ Standard deviation of the Gaussian distribution modeling the uncertainty resulting from the combination of σ_m and σ_s: $\sigma = \sqrt{\sigma_m^2 + \sigma_s^2}$.

c_d Probability that a proximity sensor **d**etects the closest obstacle in the map.

c_r Probability that a proximity sensor is **r**eflected by an unknown obstacle at a range Δ_d of the sensor discretization.

$P_m(d_i \mid l)$ Probability of measuring distance d_i if the robot is at location l and the sensor can only be reflected by an obstacle in the **m**ap. This distribution is governed by a Gaussian distribution with standard deviation σ centered at the distance o_l to the closest obstacle in the map.

$P_u(d_i)$ Probability of measuring distance d_i due to an **u**nknown obstacle. This distribution is governed by c_r and the resulting geometric distribution.

$P(d_i \mid l)$ Probability of measuring distance d_i if the robot is at location l. This distribution is a combination of $P_m(d_i \mid l)$ and $P_u(d_i)$.

h_i Distance to a **h**uman ($|h_i| = i \cdot \Delta_r$).

$P_{occ}(d_i \mid l)$ Probability that the closest obstacle is in distance d_i if the robot is at location l.

$P_h(d_i \mid h_j)$ Probability of measuring distance d_i if the sensor can only be reflected by a **h**uman in distance h_j in the direction of the sensor beam. This distribution is given by a Gaussian distribution with standard deviation σ_s centered at h_j.

$P(d_i \mid h_j, l)$ Probability of measuring distance d_i if the robot is at location l and a human is in distance h_j. This distribution is a combination of $P_m(d_i \mid l)$, $P_u(d_i)$, and $P_h(d_i \mid h_j)$.

$P(h_i \mid l)$ Probability that a human is in distance h_i if the robot is at location l.

$P(h_i \mid d_j, l)$ Probability that a human is in distance h_i if the robot is at location l and measures distance d_j.

$P_{\textbf{blocked}}(d_i)$ Probability that the measurement d_i is caused by a human **blocking** the closest obstacle in the map.

$P_{\textbf{short}}(d_i)$ Probability that the measurement d_i is **shorter** than the expected measurement.

Appendix B

Motion Equations of Synchro-Drive Robots

This section describes the fundamental motion equations for a synchro-drive mobile robot [BEF96]. The derivation begins with the correct dynamic laws, assuming that the robot's translational and rotational velocity can be controlled independently (with limited torques). To make the equations more practical, we derive an approximation that models velocity as a piecewise constant function in time. Under this assumption, robot trajectories consist of sequences of finitely many segments of circles. Such representations are very convenient for collision checking, since intersections of obstacles with circles are easy to check. We also derive an upper bound for the approximation error. The piecewise circular representation forms the basis of the dynamic window approach to collision avoidance, described in Section 5.

B.1 General Motion Equations

Let $x(t)$ and $y(t)$ denote the robot's coordinate at time t in some global coordinate system, and let the robot's orientation (heading direction) be described by $\theta(t)$. The triplet $\langle x, y, \theta \rangle$ describes the kinematic configuration of the robot. The motion of a synchro-drive robot is constrained in a way such that the translational velocity v always leads in the steering direction θ of the robot, which is a non-holonomic constraint [Lat91]. Let $x(t_0)$ and $x(t_n)$ denote the x-coordinates of the robot at time t_0 and t_n, respectively. Let $v(t)$ denote the translational velocity of the robot at time t, and $\omega(t)$ its rotational velocity. Then $x(t_n)$ and $y(t_n)$ can be expressed as a function of $x(t_0)$, $v(t)$ and $\theta(t)$:

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\, dt \tag{B.1}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\, dt \tag{B.2}$$

Equations (B.1) and (B.2) depend on the velocities of the robot, which usually cannot be set directly. Instead, the velocity $v(t)$ depends on the initial translational velocity $v(t_0)$ at t_0, and the translational acceleration $\dot{v}(\hat{t})$ in the time interval $\hat{t} \in [t_0, t]$. Likewise, the orientation $\theta(t)$ is a function of the initial orientation $\theta(t_0)$, the initial rotational velocity $\omega(t_0)$ at t_0, and the rotational acceleration $\dot{\omega}(\hat{t})$ with $\hat{t} \in [t_0, t]$. Substituting $v(t)$ and $\theta(t)$ by the corresponding initial kinematic and dynamic configuration $v(t_0), \theta(t_0), \omega(t_0)$ and the accelerations $\dot{v}(\hat{t})$ and $\dot{\omega}(\hat{t})$ yields the expression[1] :

$$\begin{aligned} x(t_n) &= x(t_0) + \int_{t_0}^{t_n} \left(v(t_0) + \int_{t_0}^{t} \dot{v}(\hat{t})\, d\hat{t} \right) \\ &\quad \cdot \cos \left(\theta(t_0) + \int_{t_0}^{t} \left(\omega(t_0) + \int_{t_0}^{\hat{t}} \dot{\omega}(\tilde{t})\, d\tilde{t} \right) d\hat{t} \right) dt \end{aligned} \tag{B.3}$$

The equations are now in the form that the trajectory of the robot depends exclusively on its initial dynamic configuration at time t_0 and the accelerations, which we assume to be controllable (for most mobile robots the accelerations determining its motion are monotonic functions of the currents flowing through the motors [JF93]. Hence, limits on the currents directly correspond to limits on the accelerations).

Digital hardware imposes constraints as to when one can set the motor currents (and thus set new accelerations). Hence, Eq. (B.3) can be simplified by assuming that between two arbitrary points in time, t_0 and t_n, the robot can only be controlled by finitely many acceleration commands. Let n denote this number of time ticks. Then, the accelerations $\dot{v}_i$ and $\dot{\omega}_i$ for $i = 1 \ldots n$ are kept constant in a time interval $[t_i, t_{i+1}]$ $(i = 1 \ldots n)$. Let Δ_t^i and $\Delta_{\hat{t}}^i$ be defined as $\Delta_t^i = t - t_i$ and $\Delta_{\hat{t}}^i = \hat{t} - t_i$ for some interval index $i = 1 \ldots n$. Using this discrete form, the dynamic behavior of a synchro-drive robot is expressed by the following equation, which follows directly from Eq. (B.3) under the assumption of piecewise constant accelerations:

$$\begin{aligned} x(t_n) &= x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \left(v(t_i) + \dot{v}_i \cdot \Delta_t^i \right) \\ &\quad \cdot \cos \left(\theta(t_i) + \omega(t_i) \cdot \Delta_t^i + \frac{1}{2} \dot{\omega}_i \cdot (\Delta_t^i)^2 \right) dt \end{aligned} \tag{B.4}$$

B.2 Approximate Motion Equations

While Eq. (B.4) describes the general case of mobile root control, it is not particularly helpful when determining the actual steering direction. This is because the trajectories generated by these equations are complex, and geometric operations such as checks for intersections are expensive to perform.

[1]Notice that the derivation of $y(t_n)$ is analogous, thus we only describe the derivation for the x-coordinate.

To derive a more practical model, we will now simplify Eq. (B.4) by approximating the robot's velocities within a time interval $[t_i, t_{i+1}]$ by a constant value. The resulting motion equation, Eq. (B.6), converges to Eq. (B.4) as the length of the time intervals goes to zero. As we will see under this assumption the trajectory of a robot can be approximated by piecewise circular arcs. This representation is well-suited for generating motion control in real time, as described in Section 5.3.

If the time intervals $[t_i, t_{i+1}]$ are sufficiently small, the term $v(t_i) + \dot{v}_i \cdot \Delta_t^i$ can be approximated by an arbitrary translational velocity $v_i \in [v(t_i), v(t_{i+1})]$, due to the smoothness of robot motion in time. Likewise, the term $\theta(t_i) + \omega(t_i) + \frac{1}{2}\dot{\omega}_i(\Delta_t^i)^2$ can be approximated by an arbitrary $\theta(t_i) + \omega_i \cdot \Delta_t^i$, where $\omega_i \in [\omega(t_i), \omega(t_{i+1})]$. This leads to the following motion equation

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cdot \cos\left(\theta(t_i) + \omega_i \cdot (\hat{t} - t_i)\right) d\hat{t} \qquad \text{(B.5)}$$

which, by solving the integral, can be simplified to

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \left(F_x^i(t_{i+1})\right) \qquad \text{(B.6)}$$

where

$$F_x^i(t) = \begin{cases} \frac{v_i}{\omega_i}(\sin\theta(t_i) - \sin(\theta(t_i) + \omega_i \cdot (t - t_i))),\ \omega_i \neq 0 \\ v_i \cos(\theta(t_i)) \cdot t,\ \omega_i = 0 \end{cases} \qquad \text{(B.7)}$$

The corresponding equations for the y-coordinate are:

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} \left(F_y^i(t_{i+1})\right) \qquad \text{(B.8)}$$

$$F_y^i(t) = \begin{cases} -\frac{v_i}{\omega_i}(\cos\theta(t_i) - \cos(\theta(t_i) + \omega_i \cdot (t - t_i))),\ \omega_i \neq 0 \\ v_i \sin(\theta(t_i)) \cdot t,\ \omega_i = 0 \end{cases} \qquad \text{(B.9)}$$

Notice that if $\omega_i = 0$, the robot will follow a straight line. Conversely, if $\omega_i \neq 0$, the robot's trajectory describes a circle, as can be seen by considering

$$M_x^i = -\frac{v_i}{\omega_i} \cdot \sin\theta(t_i) \qquad \text{(B.10)}$$

$$M_y^i = \frac{v_i}{\omega_i} \cdot \cos\theta(t_i) \qquad \text{(B.11)}$$

for which the following relation holds:

$$\left(F_x^i - M_x^i\right)^2 + \left(F_y^i - M_y^i\right)^2 = \left(\frac{v_i}{\omega_i}\right)^2 \qquad \text{(B.12)}$$

This shows that the i-th trajectory is a circle M_i about (M_x^i, M_y^i) with radius $M_r^i = \frac{v_i}{\omega_i}$. Hence, by assuming piecewise constant velocities, we can approximate the trajectory of a robot by a sequence of circular and straight line arcs.

Notice that, apart from the initial conditions, Equations (B.5) to (B.9) depend only on velocities. When controlling the robot, however, one is not free to set arbitrary velocities, since the dynamic constraints of the robot impose bounds on the maximum deviation of velocity values in subsequent intervals.

B.3 An Upper Bound on the Approximation Error

Obviously, the derivation makes the approximate assumption that velocities are piecewise constant within a time interval. This error is bounded linearly in time between control points $t_{i+1} - t_i$.

Consider the errors E_x^i and E_y^i for the x- and y-coordinate, respectively, within the time interval $[t_i, t_{i+1}]$. Let $\Delta t_i := t_{i+1} - t_i$. The deviation in the direction of any of the two axes is maximal if the robot moves on a straight trajectory parallel to that axis. Since in each time interval we approximate $v(t)$ by an arbitrary velocity $v_i \in [v(t_i), v(t_{i+1})]$, an upper bound of the errors E_x^i and E_y^i for $(i+1)$-th time interval is governed by $E_x^i, E_y^i \leq |v(t_{i+1}) - v(t_i)| \cdot \Delta t_i$, which is linear in Δt_i.

The reader should notice that this bound applies only to the internal prediction of the robot's position. When executing control, the location of the robot is measured periodically with its wheel-encoders (four times a second in our implementation).

This completes the derivation of the robot motion. To summarize, we have derived an approximate form that describes trajectories by sequences of circular arcs, and we have derived a linear bound on the error due to an approximate assumption made in the derivation (piecewise constant velocities).

Bibliography

[AV98] Kai O. Arras and Sjur J. Vestli. Hybrid, high-precision localization for the mail distributing mobile robot system mops. In *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[BBC+95] Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox, Thomas Hofmann, Frank Schneider, Jiannis Strikos, and Sebastian Thrun. The mobile robot Rhino. *AI Magazine*, 16(2):31–38, Summer 1995.

[BBCD97] Michael Beetz, Wolfram Burgard, Armin B. Cremers, and Fox. Dieter. Active localization for service robot applications. In *Proc. of the 5th Symposium for Intelligent Robotics Systems (SIRS-97)*, Stockholm, Sweden, 1997.

[BBDC98] Michael Beetz, Wolfram Burgard, Fox. Dieter, and Armin B. Cremers. Integrating active localization into high-level robot control systems. *Robotics and Autonomous Systems*, 1998.

[BCF+98a] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Gerhard Lakemeyer, Dirk Hähnel, Dirk Schulz, Walter Steiner, and Sebastian Thrun. The interactive museum tour-guide robot. In *Proc.of the Fifteenth National Conference on Artificial Intelligence*, Madison,WI, 1998.

[BCF+98b] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Gerhard Lakemeyer, Dirk Hähnel, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with a museum tour-guide robot. Technical report, University of Bonn, Inst. of Computer Science, 1998.

[BDFC98] Wolfram Burgard, Andreas Derr, Dieter Fox, and Armin B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, 1998.

[BEF96] Johann Borenstein, H.R. Everett, and Liqiang Feng. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.

[BFH97] Wolfram Burgard, Dieter Fox, and Daniel Hennig. Fast grid-based position tracking for mobile robots. In *Proc. of the 21th German Conference on Artificial Intelligence (KI'97)*, Freiburg, Germany, 1997. Springer Verlag.

[BFHS96a] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. of the Thirteenth National Conference on Artificial Intelligence*, pages 896–901, 1996.

[BFHS96b] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Position tracking with position probability grids. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT'96)*, pages 2–9. IEEE Computer Society Press, 1996.

[BK90] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 572–577, 1990.

[BK91] Johann Borenstein and Yoram Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[Cox91] Ingemar J. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, April 1991.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. Wiley, New York, 1991.

[CW90] I.J. Cox and G.T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[DRW98] Gregory Dudek, Kathleen Romanik, and Sue Whitesides. Localizing a robot with minimum travel. *SIAM Journal on Computing*, 27(2):583–604, March 1998.

[Elf87] A.E. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[Eng94] Sean Engelson. *Passive Map Learning and Visual Place Recognition*. PhD thesis, Department of Computer Science, Yale University, 1994.

[Eng95] Sean Engelson. Continuous map learning for mobile robots. In *Proc. of the 3rd French-Israeli Symposium on Robotics.*, Herzeliah, Israel, 1995.

[FBT] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems.* to appear.

[FBT96] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[FBT97] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, March 1997.

[FBTC98a] Dieter Fox, Wolfram Burgard, Sebastian Thrun, and Armin B. Cremers. A hybrid collision avoidance method for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[FBTC98b] Dieter Fox, Wolfram Burgard, Sebastian Thrun, and Armin B. Cremers. Position estimation for mobile robots in dynamic environments. In *Proc.of the Fifteenth National Conference on Artificial Intelligence*, Madison,WI, 1998.

[Fed93] C. Fedor. *TCX. An interprocess communication system for building robotic architectures. Programmer's guide to version 10.xx.* Carnegie Mellon University, Pittsurgh, PA 15213, 12 1993.

[Fir94] James R. Firby. Architecture, representation and integration: An example from robot navigation. In *Proc. of the 1994 AAAI Fall Symposium Series Workshop on the Control of the Physical World by Intelligent Agents*, New Orleans LA, 1994.

[Gat92] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. of AAAI-91*, San Jose, CA, 1992.

[GBFK98] Jens-Steffen Gutmann, Wolfram Burgard, Dieter Fox, and Kurt Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, 1998.

[GCJK97] Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. Many robots make short work. *AI magazine*, 18(1), Spring 1997.

[GS96] Jens-Steffen. Gutmann and Christian Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proc. of the 1st Euromicro Workshop on Advanced Mobile Robots.* IEEE Computer Society Press, 1996.

[HBL98] Dirk Hähnel, Wolfram Burgard, and Gerhard Lakemeyer. GOLEX — bridging the gap between logic (GOLOG) and a real robot. In *Proc. of the 22nd German Conference on Artificial Intelligence (KI'98), Bremen, Germany*, LNCS. Springer Verlag, 1998.

[HEK93] Joachim Hartung, Bärbel Elpelt, and Karl-Heinz Klösener. *Statistik: Lehr- und Handbuch der angewandten Statistik*. R. Oldenbourg Verlag München Wien, 9 edition, 1993.

[Hen97] Daniel Hennig. Globale und lokale positionierung mobiler roboter mittels wahrscheinlichkeitsgitter. Master's thesis, University of Bonn, Germany, 1997. in German.

[HK96] Joachim Hertzberg and Frank Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT'96)*, pages 68–73. IEEE Computer Society Press, 1996.

[JF93] Joseph L. Jones and Anita M. Flynn. *Mobile Robots: Inspiration to Implementation*. A K Peters, Ltd., 1993. ISBN 1-56881-011-3.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Tansaction of the ASME – Journal of basic engineering*, pages 35–45, March 1960.

[KB91a] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, April 1991.

[KB91b] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, pages 47–63, August 1991.

[KBM98] David Kortenkamp, R.P. Bonasso, and R. Murphy, editors. MIT/AAAI Press, Cambridge, MA, 1998.

[KC95] Maher Khatib and Raja Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. International Conference on Intelligent Autonomous Systems (IAS'4)*, 1995.

[KCK96] Leslie Pack Kaelbling, Anthony R. Cassandra, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[Kha86] Oussama Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[KLC95] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. Technical report, Brown University, 1995.

[Kle94] Jon Kleinberg. The localization problem for mobile robots. In *Proc. of the 35th IEEE Symposium on Foundations of Computer Science*, 1994.

[KMRS97] Kurt Konolige, Karen Myers, Enrique Ruspini, and Alessandro Saffiotti. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[KS96] Sven Koenig and Reid Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, 1996.

[KS98] Sven Koenig and Reid Simmons. A robot navigation architecture based on partially observable markov decision process models. In Kortenkamp et al. [KBM98].

[KW90] S. King and C. Weiman. Helpmate autonomous mobile robot navigation system. In *Proc. of the SPIE Conference on Mobile Robots*, pages 190–198, Boston, MA, November 1990. Volume 2352.

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991. ISBN 0-7923-9206-X.

[LDK95] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, 1995.

[LDW91a] John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[LDW91b] John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous map building and localization for an autononmous mobile robot. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Osaka, Japan, November 1991.

[LM94] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)*, 1994.

[LM97] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[Lov91] William S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations Research*, 39(1):162–175, 1991.

[LRL+97] Hector J. Levesque, Raymond Reiter, Yves Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[May90] Peter S. Maybeck. The Kalman filter: An introduction to concepts. In Cox and Wilfong [CW90].

[McD91] Drew McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

[McD92] Drew McDermott. Robot planning. *AI Magazine*, 13(2):55–79, 1992.

[ME85] Hans P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 116–121, 1985.

[MH96] Robin R. Murphy and D. Hershberger. Classifying and recovering from sensing failures in autonomous mobile robots. In *Proc. of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[Mor88] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[Nil84] Nils J. Nilsson. SHAKEY the robot. Technical report, SRI International, 1984.

[NPB95] Illa Nourbakhsh, Rob Powers, and Stan Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, Summer 1995.

[OHD97] Sageev Oore, Geoffrey E. Hinton, and Gregory Dudek. A mobile robot that learns its place. *Neural Computation*, 9(3):683–699, April 1997.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[PR95] Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1088–1094. Morgan Kaufmann Publishers, Inc., 1995.

[RJ86] Lawrence R. Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *IEEE ASSP magazine*, pages 4–16, January 1986.

[RN95] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 17. Number 0-13-103805-2 in Series in Artificial Intelligence. Prentice Hall, 1995.

[SA92] K. Schmidt and K. Azarm. Mobile robot navigation in a dynamic world using an unsteady diffusion equation strategy. In *Proc. of the Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992.

[SC94] Bernt Schiele and James L. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1628–1634, 1994.

[SGH+] Reid Simmons, Richard Goodwin, Karen Zita Haigh, Sven Koenig, Joseph O'Sullivan, and Manuela M. Veloso. Xavier: Experience with a layered robot architecture. *ACM magazine* Intelligence. to appear.

[SGS92] Gary Shaffer, Javier Gonzalez, and Anthony Stentz. Comparison of two range-based estimators for a mobile robot. In *SPIE Conf. on Mobile Robots VII*, volume 1831, pages 661–667, 1992.

[Sim96] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. of the IEEE International Conference on Robotics and Automation*, 1996.

[SK95] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. International Joint Conference on Artificial Intelligence*, 1995.

[SK97] Hagit Shatkey and Leslie Pack Kaelbling. Learning topological maps with weak local odometric information. In *Proc. of the International Joint Conference on Artificial Intelligence*, 1997.

[TB96] Sebastian Thrun and Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[TBB+98] Sebastian Thrun, Arno Bücken, Wolfram Burgard, Dieter Fox, Thorsten Fröhlinghaus, Daniel Hennig, Thomas Hofmann, Michael Krell, and Timo Schimdt. Map learning and high-speed navigation in RHINO. In Kortenkamp et al. [KBM98].

[TFB98] Sebastian Thrun, Dieter Fox, and Wolfram Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 1998.

[TGF+98] Sebastian Thrun, Jens-Steffen Gutmann, Dieter Fox, Wolfram Burgard, and Benjamin Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proc.of the Fifteenth National Conference on Artificial Intelligence*, Madison,WI, 1998.

[Thr98] Sebastian Thrun. Finding landmarks for mobile robot navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[Thr99a] Sebastian Thrun. A bayesian approach to landmark discovery in mobile robot navigation. *Machine Learning*, 1999.

[Thr99b] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 1:27–71, 1999.

[VTG96] Sjur J. Vestli and N. Tschichold-Gürman. Mops: A system for mail distribution in office-type buildings. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT'96)*. IEEE Computer Society Press, 1996.

[WWvP94] Gerhard Weiß, Christopher Wetzler, and Ewald von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 595–601, 1994.

[Yam96] Brian Yamauchi. Mobile robot localization in dynamic environments using dead reckoning and evidence grids. In *Proc. of the 1996 IEEE International Conference on Robotics and Automation*, pages 1401–1406, Minneapolis, MN, April 1996.

[YLar] Brian Yamauchi and Pat Langley. Place recognition in dynamic environments. *Journal of Robotic Systems*, to appear. Special Issue on Mobile Robots.