ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

# Supervised Learning of Places from Range Data using AdaBoost

Master Thesis

Óscar Martínez Mozos

July 2004 – December 2004

# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt und alle Stellen, die wörtlich oder sinngem aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Außerdem erkläre ich, dass die Masterarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

(Óscar Martínez Mozos)
Freiburg, den December 21, 2004

## Acknowledgment

I would like to thank all the people who have helped me to write this thesis. Especially Prof. Dr. Wolfram Burgard who gave me the opportunity to work in his research group. I would like to thank him for his great support and the opportunity to work in an active research area. Many thanks also go out to the group members: Cyrill Stachniss, Rudolph Triebel and Dirk Hähnel. They are all great people.

Furthermore, as this work was written during a rather difficult time in my life, I would like to thank certain friends who helped stay focus. In no particular order: Adam, Michael, Christine, Christian, Julio, Paula, Ana, David, Victor, Vicente, Jose Ángel, Daniel, Karla, Karina, Patricia, Marta, and many others.

And finally, and most important, I would like to thank my parents and family.

## Agradecimientos

Me gustaria dar las gracias a toda la gente que me ha ayudado a desarollar el trabajo de esta tesis. En especial al Prof. Dr. Wolfram Burgard por darme la oportunidad de trabajar en su grupo de investigación, y al resto de componentes de este grupo: Cyrill Stachniss, Rudolph Triebel y Dirk Hähnel. Todos ellos son magníficas personas.

Por otro lado, este trabajo ha sido escrito en unos momentos personales bastantes difíciles. Por ello quiero agradecer el apoyo de todos aquellos amigos que me han ayudado a seguir adelante. Sin previo orden: Adam, Michael, Christine, Christian, Julio, Paula, Ana, David, Victor, Vicente, Jose Ángel, Daniel, Karla, Karina, Patricia, Marta y alguno más que habré olvidado.

Y finalmente, y más importante, quiero dar las gracias a mis padres y resto de familia.

# Contents

# 1. Introduction

The word *robot* comes from the Czech word *robota* which means “rudgery” or “servitude”. This term was first used by Karel Čapek in his work “R.U.R. (Rossum’s Universal Robots)” [40] written in 1920. The following is a more formal definition [28]:

> “Programable machine or electronic device which is able to manipulate objects and to carry out operations that were before carried out by humans.”

According to this definition, many different machines can be considered robots. The Japan Robot Association [15] gives a classification of robots according to their *intelligence level*: manual manipulator, fixed sequence robot, variable sequence robot, playback robot, numerical control robot, and intelligent robot. Here we focus on the class of intelligent robot, which characterizes robots that “can understand and interact with changes in the environment”.

One type of intelligent robot that has been of increasing interest in past years is the autonomous mobile robot. This kind of robot is designed and progammed to successfully and autonomously navigate in different environments, such as indoor (Burgard *et al.* [5], Stachniss *et al.* [34], Thrun *et al.* [36]), outdoor (Hähnel *et al.* [12], Talluri and Aggarwal [35]) or underwater (Whitcomb *et al.* [21], Williams *et al.* [43]) environments .

In particular, the navigation of mobile robots indoors implies that a robot should be able to perform in environments that are designed and structured for humans. Consider a mobile service-robot that servers in a domestic residence. This robot must be able to maneuver through places such as corridors or rooms. Moreover, the robot should be able to recognize these places in the same way that a person does. That is, if the robot is in a corridor, then it must be able to relate the term “corridor” to this place. When we relate a term, e.g “corridor”, with a place, we carry out what is called semantic labeling. This semantic information is more user-friendly for humans. Suppose that a member of the family asks the robot for its position. If the robot answers with the word “corridor”, it would then be easy for this person to imagine where the robot is. In contrast, if the robot answers with the exact coordinates of its pose, then the information could not be very useful to that person.

The semantic labeling of different places can also to a great extent improve the interaction between humans and mobile robots in indoor environments. For example, the sentence “go to the corridor” would be enough for the robot to go to the corridor, because the robot would know which place corresponds to the term “corridor”. Furthermore, semantic labeling allows the robot to carry out tasks more efficiently. For instance, if a service-robot has to make a coffee, it would be a waste of time to look for a coffee machine throughout the whole house. Instead, with

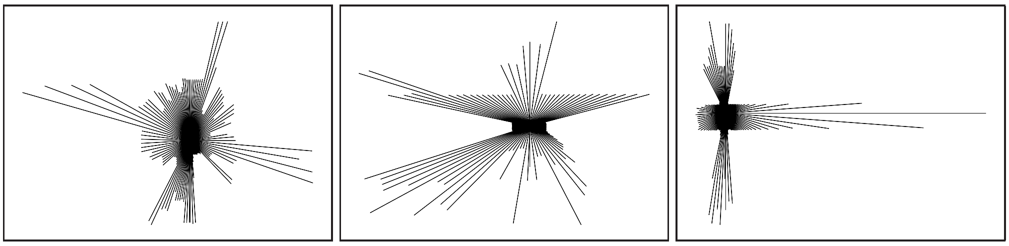


Figure 1.1.: Example raw beams recorded in a room, a doorway, and a corridor.

semantic information, the robot needs only to search for the coffee machine when it enters the appropriate places, i.e. the kitchen.

The work presented in this thesis describes a technique that enables a mobile robot to interact in an indoor environment designed for humans. In particular, this work demonstrates how a robot recognizes the places of an indoor environment, relating different terms like “corridor”, “room”, “hallway” or “doorway” to these places. To achieve this, we propose a method based on supervised learning to recognize the different places into which an environment can be divided. Our approach creates a classifier which will be used by the mobile robot to classify subsequent places of location. In order to learn the term related to a place, the robot must first make certain observations in the environment. In our case, these observations are proximity measurements obtained by a laser range finder. No other sensor information is used. Figure 1.1 shows an example of observations recorded in three different places in an office building. Each example belongs to a different place in an indoor environment. The beams are represented by their proximity measurement and by their angle relative to the robot. However, the raw proximity information is not enough to characterize a set of beams taken from a certain pose. Therefore, we designed a set of simple geometric features that represent each set of beams, and that will be used in the process of learning and classifying. Moreover, the different methods used in this work for learning are all based in the **AdaBoost** algorithm, which allows the boosting of our simple geometric features to a strong classifier for place categorization.

The contribution of this work is an approach that learns the relation between the proximity measurements taken at a pose and the semantic term related to that pose. Additionally, our approach creates a classifier that allows the classification of so far unknown poses, like the ones depicted in Figure 1.1, into a set of different semantic terms (“corridor”, “room”, “hallway” or “doorway”). Our approach is supervised, which has the advantage that the resulting semantic labels correspond to user-defined classes.

## 1.1. Related Work

In the past, several authors focussed on the problem of adding semantic information to places. In the paper by Buschka and Saffiotti [6], a virtual sensor is described which automatically segments the space into room and corridor regions, and computes a set of characteristic parameters for each region. This algorithm is incremental, in the sense that it only maintains a local topological map of the space recently explored by the robot and generates information about each detected room whilst rooms are visited. Althaus and Christensen [1] use the Hough transform from sonar readings to detect two parallel lines which are considered part of a corridor. Doorway detection is carried out using the gaps between these lines. With the detection of corridors and doorways, they construct a topological map for navigation and obstacle avoidance. Also Koenig and Simmons [19] use a pre-programmed routine to detect doorways from range data. Compared to these approaches, our method allows us to classify more classes, i.e. room, corridor, doorway and hallway. Furthermore, whereas the former methods need of several scans for extracting geometrical characteristics from the different places, our geometrical features are extracted from only one range scan. Moreover, our algorithm does not require any pre-defined routine for extracting high-level features from different places. Instead, it uses the $\mathbf{AdaBoost}$ algorithm to boost simple geometric features into a strong classifier.

Other researches also apply learning techniques to localize the robot or to identify distinctive states in the environment. For example, Oore *et al.* [26] train a neural network to estimate the location of a mobile robot in its environment using the odometry information and ultrasound data. Kuipers and Beeson [20] apply different learning algorithms to learn topological maps of the environment. Although these methods are able to create topological maps, they do not add semantic information to these maps. In contrast to this, our algorithm is able to label each pose of the robot with a semantic term and, therefore, a semantic map can be constructed.

Additionally, learning algorithms have been used to identify objects. For example, Anguelov and colleagues [2, 3] apply the $\mathbf{EM}$ algorithm to cluster different types of objects, i.e. walls and doors, from sequences of range data. The $\mathbf{EM}$ algorithm fits the model parameters that best represent the different objects used as examples. To achieve this, each object must be first modeled with a probabilistic distribution. In a recent work, Torralba and colleagues [38] use Hidden Markov Models for learning places from image data. They model the appearance of each location as a set of $K$ views (a mixture of $K$ spherical Gaussians). In contrast to these approaches, the $\mathbf{AdaBoost}$ algorithm used in this work does not need previous probabilistic models for the different places. Instead, a set of simple geometrical features is used to identify different places. These features are supply to the $\mathbf{AdaBoost}$ algorithm as a set, without any prior model.

In particular, the $\mathbf{AdaBoost}$ algorithm has been successfully applied in different classification problems. Viola and Jones [42] extract a set of features and use the $\mathbf{AdaBoost}$ algorithm to detect faces in images. Tieu and Viola [37] use $\mathbf{AdaBoost}$ for image retrieval. Also Treptow *et al.* [39] use the $\mathbf{AdaBoost}$ algorithm to track a ball without color information in the context of RoboCup. The key idea in all these approaches is to use simple features that are very easy to calculate. In these papers, the features were composed of grey value substractions between different fragments of the images. In contrast our approach is based on laser range finder data, and no image information is used. However, the key idea of our method is similar: we also calculate very simple geometrical features from our laser range scans, which will then be used by the **AdaBoost** algorithm.

The contribution of this work is a system for a mobile robot that learns the relation between the proximity information obtained at a location in the environment, and the semantic term related to this location. Aditionally, our approach is able to classify new environments for which no training data information is available. As long as the environment has a similar structure, one can re-used previously learned models.

The rest of this work is organized as follows. Chapter 2 gives a short introduction to supervised learning and explains several concepts related to it. It also describes the probably approximately correct (PAC) framework and the Boosting method. Chapter 3 introduces the **AdaBoost** algorithm. It also explains how to extend the original binary version of this algorithm to the multi-class case using different approaches. Chapter 4 explains in detail the aim of this work, which is the supervised learning of places from laser range data. It also describes the different simple geometrical features extracted from the raw laser data which will be used in the process of learning and classification. In Chapter 5 we present several experiments which demonstrate that our simple features can be boosted to a robust classifier of places. Additionally we analyze whether the resulting classifier can be used to classify places in environments for which no training data were available. Finally, Chapter 6 contains the conclusions and discusses potential future work.

# 2. Supervised Learning

## 2.1. Introduction

This section introduces, with the help of an example, some basic concepts of supervised learning that will be useful for the understanding of this thesis.

Suppose we want to develop a service-robot for domestic residences like the robot introduced in Chapter 1. The purpose of this robot is to serve a family. We want our first prototype of service-robot to do only two tasks: "make a coffee" and "make the bed". In order to make a coffee, the robot must first be able to recognize a kitchen in the house, and for making the bed the robot must be able to recognize the bedrooms.

Thus, the first step is to design a system so that the robot is able to recognize the kitchen and bedrooms of a house. Because each family has a different house it is necessary to develop a system in such a way that the robot is able to learn the difference between the kitchen and the bedrooms for each particular house. For this example, we mounted a sensor to our robot that is able to take three measurements from a room: the width, the length and the height. We assume these three measurements are enough to distinguish a kitchen from a bedroom in a house. In the next step we place the robot in a kitchen. The robot then takes the following three measurements: $\mathrm{Length} = 5\ \mathrm{m}$, $\mathrm{Width} = 4\ \mathrm{m}$, and $\mathrm{Height} = 3\ \mathrm{m}$. The robot then keeps these measurements in a database together with the term $\mathbf{Kitchen}$. After that, we place the robot in a bedroom where it takes the following measurements: $\mathrm{Length} = 10\ \mathrm{m}$, $\mathrm{Width} = 3\ \mathrm{m}$, and $\mathrm{Height} = 2.5\ \mathrm{m}$. The robot then keeps this data together under the term $\mathbf{Bedroom}$.

Now it should be easy for the robot to recognize the kitchen in the house. It only has to take the previous three measurements in a room and compare them with the ones in the database, which are related to the term $\mathbf{Kitchen}$. If they are the same, then the place is labeled as $\mathbf{Kitchen}$.

This is a simple method for recognizing places: to save all measurements of each place in a database under its relative term. This kind of method is called "supervised learning" as the robot is told which label is associated to each measurement.

Now suppose that we want to learn information about the kitchens in all the houses in a country. In this case we will need to save the measurements of all the kitchens in the country together with the label $\mathbf{Kitchen}$. In order to classify a new room, the robot will have to compare the measurements taken in this room with all the measurements taken from all the kitchens in the country. If the measurements taken by the robot fit with any other in the database under the term $\mathbf{Kitchen}$, then the robot will label the new room as $\mathbf{Kitchen}$. This comparison can be quite time consuming. One way to improve this comparison is to extract rules from the data that can help to recognize a kitchen without the need of comparing all the measurements in the database. For example, if we have the following two different sets of measurements, taken from two different kitchens $\mathrm{kitchen}_1$ and $\mathrm{kitchen}_2$:

$$\mathrm{kitchen}_1 = \begin{cases} \mathrm{Length} &= 10\ \mathrm{m} \\ \mathrm{Width} &= 5\ \mathrm{m} \\ \mathrm{Height} &= 2.5\ \mathrm{m} \end{cases}$$

$$\mathrm{kitchen}_2 = \begin{cases} \mathrm{Length} &= 10\ \mathrm{m} \\ \mathrm{Width} &= 3\ \mathrm{m} \\ \mathrm{Height} &= 3\ \mathrm{m} \end{cases},$$

we can extract a rule in the form:

if Length = $10\ \mathrm{m} \Rightarrow$ the place is a $\mathbf{Kitchen}$

This rule was extracted using the information shared between the two kitchens. In this case the value of the measurement $\mathrm{Length}$. Thus, the next time the robot is in a room and takes the three measurements, it only has to check this rule to recognize this room as a $\mathbf{Kitchen}$, avoiding the need to comparing the taken measurements with the ones from the two kitchens. In this same way, the robot can extract other simplified rules with the measurements from all the kitchens in a country.

This is the key idea of learning: to extract simplified concepts from a given set of examples in order to use these rules later on to classifiy new unseen examples.

## 2.2. Description of a Learning Task

The example in the previous Section 2.1 is called a supervised learning task and can be formally described as follows (see Mitchell [24] for more details):

- There is some space of possible instances $X$ over which different target concepts can be defined. In our case $X$ represents the set of all rooms that can be found in a house. The target concepts are the places we want to recognize, like $\mathbf{Kitchen}$ and $\mathbf{Bedroom}$.
- Each instance $x \in X$ is described by a set of attributes. In our case these attributes are "Length", "Width" and "Height".
- Let $\mathcal{C}$ refer to some set of target concepts that our learner (our service-robot) might be called upon to learn. Each target concept $c \in \mathcal{C}$ corresponds to some subset of $X$, or equivalently to some boolean-valued function $c : X \to \{0, 1\}$. For example, the target concept $c = \mathbf{Kitchen}$ corresponds to the subset of places in a house which are kitchens. If one instance $x \in X$ is a positive example of $c$, that is, the place represented by $x$ is a kitchen, then we will write $c(x) = 1$; and $c(x) = 0$ if $x$ is a negative example (a different place).

- We assume that different instances in $X$ may be encountered with different frequencies, for example we can suppose that in a house there are more bedrooms than kitchens. A convenient way to model this is to assume that there is some unknown probability distribution $\mathcal{D}$ that defines the probability of encountering each instance in $X$ (e.g., $\mathcal{D}$ might assign a lower probability to encountering a kitchen than a bedroom). Notice that $\mathcal{D}$ says nothing about whether $x$ is a positive or negative example; it only determines the probability that $x$ will be encountered.

- The learner $L$ considers some set $H$ of possible hypotheses when attempting to learn a target concept. For example, $H$ might be the set of all hypotheses describable by our attributes. For example:

$$h_1(x) = \begin{cases} 1 & \text{if} \begin{cases} \text{Length of } x & > & 12 \text{ m} \\ \text{Width of } x & > & 2 \text{ m} \\ \text{Height of } x & = & 3 \text{ m} \end{cases} \\ 0 & \text{Otherwise} \end{cases} \tag{2.1}$$

$$h_2(x) = \begin{cases} 1 & \text{if Height of } x = 3 \text{ m} \\ 0 & \text{Otherwise,} \end{cases} \tag{2.2}$$

  where $h_1$ represents the set of rooms in a house which are longer than 12 meters, wider than 2 meters and have a height of 2 meters. And $h_2$ represents the rooms with a height of 3 meters. As we can see, the sets represented by each hypothesis do not need to be exclusive.

- After observing a sequence of training examples $S \subset X$, the learner $L$ must output some hypothesis $h$ from $H$, which is an estimate from $c$. The output can also be a subset of $H$.

- The hypotheses output by the learner will be somehow used to create a classifier to classify new instances.

There are some concepts that must be clear in a learning process. On one hand, the learner $L$ must learn from a training set $S$ which is a subset of $X$. But on the other hand, the target concept $c$ is defined over the whole set $X$. That means that the output $h(x)$ generated by the learner must always be considered as an approximation of the target concept $c(x)$. It may be possible that $h(x) = c(x) \; \forall x \in X$, but we can only be sure of that if $S = X$, which implies that the learner uses all the possible instances of $X$ to learn the target concept. Obtaining a training set with all possible instances is not possible for the majority set of problems. As a result, the set $S$ must be sufficiently representative of the whole set of instances $X$. Some guidelines for obtaining a good training set are given in the book of Witten and Frank [44].

After describing the learning task we are ready to define the learning algorithm. This algorithm is responsible for the selection of the hypothesis $h$ which best approximates the target concept. Different algorithms and methods exist for the task of learning. Some of them are decision trees, neural networks, bayesian networks, inductive learning, algorithm based in version spaces, genetic algorithms or inductive, learning among others. In his book, Mitchel [24] gives a good introduction to this area.

In our service-robot example, suppose that we have selected a learning algorithm $L$, a training set $S$ and we have the following hypothesis for the target concept $\mathbf{Kitchen}$ as output:

$$h(x) = \begin{cases} 1 & \text{if} \begin{cases} \text{Length of } x & > & 12 \text{ m} \\ \text{Width of } x & > & 2 \text{ m} \end{cases} \\ 0 & \text{Otherwise} \end{cases} \tag{2.3}$$

This hypothesis is an approximation for describing a kitchen after applying the learning algorithm to the training set. Thus, this hypothesis is only an approximation of all the training examples, which implies that it classifies the training examples with a low error. However, if the robot finds a new bedroom which is 13 meters long and 3 meters wide, it will classify it as a kitchen, which is an error. The question here is wether the hypothesis $h$ will also have a low classification error in a set of unseen examples.

An answer to this question is given by the *inductive learning* assumption. Informally, any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples. We can also face the question from a different point of view. That is, we can try to estimate the error of our hypothesis $h$ with respect to a new example (true error) providing the error of $h$ in the training set (training error). These two errors are formally defined as follows:

**True error** The true error $\mathrm{error}_{\mathcal{D}}(h)$ of hypothesis $h \in H$ with respect to target concept $c \in \mathcal{C}$ and distribution $\mathcal{D}$ is the probability that $h$ will misclassify an instance $x \in X$ drawn at random according to $\mathcal{D}$:

$$\mathrm{error}_{\mathcal{D}}(h) = \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)] \tag{2.4}$$

**Training error** The training error (also called sample error) $\mathrm{error}_S(h)$ of hypothesis $h$ with respect to target concept $c \in \mathcal{C}$ and training set $S$, is the fraction of misclassified instances $x \in S$:

$$\mathrm{error}_S(x) = \frac{1}{|S|} \sum_{x \in S} \delta(c(x), h(x)), \tag{2.5}$$

where:

$$\delta(c(x), h(x)) = \begin{cases} 1 & \text{if } c(x) \neq h(x) \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Different statistical approximations can be used to calculate the true error provided the training error. For more detail see Mitchell [24] and Witten and Frank [44].

## 2.3. PAC Learning

Some other important questions related to a specific learning task are:

- What is the number of training examples needed to assure that the hypothesis output by the learner has a low true error?
- Can this number be bounded in some way?
- Can we bound the time the learner needs to output a hypothesis?

These questions can be answered for certain sets of problems using the probably approximately correct (PAC) framework [41]. The idea of the PAC learnability is to characterize classes of target concepts that can be reliably learned from a polynomial number of randomly drawn training examples and a polynomial amount of computation. PAC learnability is defined formally as follows:

**PAC learnability** Consider a concept class $C$ over a set of instances $X$ of length $n$ and a learner $L$ using hypothesis space $H$. $C$ is "PAC learnable" by $L$ using $H$, if for all $c \in \mathcal{C}$, distribution $\mathcal{D}$ over $X$, $\epsilon$ such that $0 < \epsilon < 1/2$, and $\delta$ such that $0 < \delta < 1/2$, the learner $L$ will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such as that $error_D(h) \leq \epsilon$, in time polynomial in $1/\epsilon$, $1/\delta$, $n$, and $size(c)$.

Here $n$ is the size of the instances in $X$. In our servant-robot task each $x$ is described by $3$ attributes: "Length", "Width", "Height", so $n = 3$. The second space parameter, $size(c)$, is the encoding length of $c$ which in our case is bounding also by 3, the maximum number of attributes that can be used for describing a room.

The previous definition of PAC learning may appear at first to be concerned only with the computational time, whereas we are usually interested in the number of training examples. However they are closely related. In fact, a typical approach to show that some class $\mathcal{C}$ of target concepts is PAC learnable, is to first demostrate that each target concept $c \in \mathcal{C}$ can be learned from a polynomial number of training examples and then show that the processing time per example is also polynomial bounded.

A learner $L$ following the previous definition is also known as *strong PAC learning algorithm*. On the other hand, a *weak PAC learning algorithm* is defined analogously except that it is only required to satisfy the condition $\epsilon \geq 1/2 - \gamma$, where $\gamma > 0$ is either a constant, or decreases as $1/p$ where $p$ is a polynomial in the relevant parameters. A hypothesis learned from a weak PAC learning algorithm is called a weak hypothesis. The term "weak hypothesis" is also used for a hypothesis which performs just slightly better than a random guessing.

Other various extensions and generalizations of the basic PAC concept can be found in Anthony and Bartlett [4], Haussler [14] and Kearns and Vazirani [18].

## 2.4. Boosting

Boosting is a general method which attempts to improve the accuracy of a given learning algorithm (see Freund and Shapire [8], Meir and Rätsch [23] and Shapire [31]). It has its roots in the PAC framework (see Section 2.3). Kearns and Valiant [17, 16] were the first to pose the question of whether a weak learning algorithm, which performs just slightly better than a random guessing in the PAC model, can be combined into an arbitrarily accurate strong learning algorithm. Later, Schapire [32] demonstrated that any weak learning algorithm can be efficiently transformed or boosted into a strong learning algorithm.

The underlying idea of boosting is to combine a set of weak hypotheses $\{h_1, h_2, \ldots, h_T\}$ to form a strong hypothesis $h_s$ such that the performance of the strong hypothesis is better than the peformance of each of the single weak hypothesis $h_t$. Formally:

$$h_s(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}), \tag{2.7}$$

here $w_t$ denotes the weight of hypothesis $h_t$. Both $w_t$ and the hypothesis $h_t$ are to be learned within the boosting procedure. The resulting strong hypothesis $h_s$ has the form of a weighted majority vote classifier.

Formally, boosting proceeds as follows: the boosting algorithm is provided with a set of labeled training examples $(x_1, y_1), ..., (x_n, y_n)$, where $y_i$ is the label associated with instance $x_i$. In the binary case, $y_i = 1$ if $x_i$ is a positive example and $y_i = 0$ if $x_i$ is a negative example. On each round $t = 1, \ldots, T$, the boosting algorithm devises a distribution $D_t$ over the set of examples, and requests (from an unspecified oracle) a weak hypothesis $h_t$ with low error $\epsilon_t$ with respect to $D_t$, that is $\epsilon_t = \Pr[h_t(x_i) \neq y_i]$. Thus, distribution $D_t$ specifies the relative importance of each example for the current round. After $T$ rounds, the booster must combine the weak hypotheses into a strong one. The intuitive idea is to alter the distribution over the training examples in a way that increases the probability of the “harder” elements, thus forcing the weak learner to generate new hypotheses that make less mistakes on these elements.

To summarize, this chapter described the key idea of supervised learning, which is the extraction of rules from certain data, and the use of these rules for the classification of new examples. The PAC model was also described, which allows us to calculate the number of examples needed for learning a particular task. Finally, the basic idea of boosting was introduced. Boosting is the basis for the algorithms explained in the next chapters.

# 3. The AdaBoost Algorithm

## 3.1. AdaBoost as a Binary Classifier

In this section we will concentrate on the case of binary classification.

The **AdaBoost** algorithm, introduced by Freund and Schapire [9], is one of the most popular boosting algorithms. Following the idea of boosting (see Section 2.4), the **AdaBoost** algorithm takes as an input a training set of examples $(x_1, y_1), ..., (x_N, y_N)$, where each $x_i$ belongs to some domain space $X$ and each label $y_i$ pertains to the label set $Y$. In the case of binary classification we assume $Y = \{0, 1\}$. On each round $t = 1, \ldots, T$, **AdaBoost** call a weak learning algorithm repeatedly to select a weak hypothesis. Following the notation from Freund and Schapire [9], we will call this weak learning algorithm **WeakLearn**. This weak learner takes as input the training set of examples and outputs a weak hypothesis for their classification. In Sections 4.3 and 4.4 we will describe in more detail the implementation of the **WeakLearn** algorithm.

Unlike previous boosting algorithms (Freund [10, 7] and Shapire [32]), the **AdaBoost** algorithm needs no prior knowledge of the accuracies of the weak hypotheses. Rather, it adapts to these accuracies and generates a weighted majority hypothesis in which the weight of each weak hypothesis is a function of its accuracy.

The complete algorithm is described in Algorithm 1. In this algorithm the distribution $D$ is applied over the instances of the training set and is controlled by the learner. This distribution can be set initially as the uniform distribution so that $D_1(i) = 1/N$. On each round $t$, the algorithm maintains a set of weights $\mathbf{w}^t = \{w_1^t, \ldots, w_N^t\} \in D_t$ over the training examples and computes a distribution $\mathbf{p}^t$ by normalizing these weights. The distribution $\mathbf{p}^t$ is fed to the weak learner **WeakLearn** which generates a hypothesis $h_t$ that, hopefully, has a small error with respect to the distribution $\mathbf{p}^t$. The accuray of the weak hypothesis $h_t$ is measured by its error:

$$\epsilon = \mathrm{Pr}_{i \sim D_t}\left[h_t(x_i) \neq y_i\right] = \sum_{i:h_t(x_i) \neq y_i} p^t(i). \tag{3.1}$$

Notice that the error is measured with respect to the distribution $\mathbf{p}^t$ on which the weak learner **WeakLearn** was trained. In practice, the **WeakLearn** algorithm may be able to use the weights $\mathbf{p}^t$ on the training examples. Alternatively, when this is not possible, a subset of the training examples can be sampled according to $\mathbf{p}^t$, and these resampled examples can be used to train the weak learner.

Using the new hypothesis $h_t$, the boosting algorithm generates the next weight vector $\mathbf{w}^{t+1}$, and the process is repeated. After $T$ iterations, the final hypothesis $h_f$ is generated. The hypothesis $h_f$ combines the outputs of the $T$ weak hypotheses using a weighted majority vote.

Freund and Schapire [9] proved that, for binary classification problems, the training error of the final hypothesis $h_f$ generated by the **AdaBoost** algorithm is bounded by:

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} \leq \exp\left(-2\sum_{t=1}^{T}\gamma^2\right), \tag{3.2}$$

where $\epsilon_t = 1/2 - \gamma_t$ is the error of the $t$th weak hypothesis. Since a hypothesis that makes an entirely random guess has error 1/2, $\gamma_t$ measures the accuracy of the weak hypothesis $h_t$ relative to random guessing. This bound shows that the final training error drops exponentially if each of the weak hypotheses is better than a random guess.

## 3.2. Sequential AdaBoost

In the previous Section 3.1 we dealt only with binary classification problems in which the set $Y$ of possible labels contained only two elements, say $\{0, 1\}$. In this section we describe a structure for using binary **AdaBoost** classifiers in a multi-class case in which $Y = \{1, 2, \ldots, K\}$ is a finite set of $K$ labels.

A way to construct a multi-class classifier is to arrange several binary classifiers into a decision list. If we have $K$ possible labels for the classification, we can construct a decision list using $K-1$ binary classifiers $\{c_1, c_2, \ldots, c_{K-1}\}$. Each binary classifier $c_k$ uses a binary hypothesis of the form $h_k : X \to Y = \{0, 1\}$ to classify an example. Figure 3.1 illustrates the structure of such a decision list classifier. Each element $c_k$ in the list is a binary classifier that determines if an example $x$ pertains to the corresponding class $k \in Y$ (i.e. the label of $x$ is $k$). If the classifier $c_k$ returns a positive result $h_k(x) = 1$, then the example $x$ is assumed to be correctly classified and is assigned the label $k$. Otherwise, it is recursively passed on to the next element in the list and the process is repeated. The last classifier $c_{K-1}$ assigns label $K-1$ to the example $x$ if $h_{K-1}(x) = 1$, and label $K$ if $h_{K-1}(x) = 0$. Algorithm 2 shows the general procedure used to classifiy a new example using a decision list.

This method for constructing multi-class classifiers is quite general and can be used with any binary classifier. In this work, however, we will limit our use to binary classifiers obtained using the binary version of the **AdaBoost** algorithm (see Section 3.1).

One important question in the context of a sequential classifier is how to choose the order in which the individual binary classifiers are arranged. This order can have a major influence on the overall classification performance, because the individual classifiers typically are not error-free and classify with different accuracies. Since the first element of such a sequential classifier processes more data than subsequent elements, it is a good strategy to order the classifiers according to their estimated error in ascending order. We will explain this strategy further with the following example.

**Algorithm 1** AdaBoost

**Input:** sequence of $N$ labeled examples $(x_1, y_1), \dots, (x_N, y_N)$,
distribution $\mathcal{D}$ over the $N$ examples,
weak learning algorithm **WeakLearn**,
integer $T$ specifying number of iterations.

**Initialize** the weight vector $w_i^1 = D(i)$ for $i = 1, \dots, N$.

**for** $t = 1, \dots, T$ **do**

1. Set
$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$
2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$ and get back a hypothesis $h_t : X \to [0, 1]$.
3. Calculate the error of $h_t$:
$$\epsilon_t = \sum_{i=1}^N p_i^t \, |h_t(x_i) - y_i|$$
4. Set
$$\beta_t = \frac{\epsilon_t}{(1 - \epsilon_t)}$$
5. Set the new weights:
$$w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$$

**end for**

**Output:** The final strong hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \left(\log \frac{1}{\beta_t}\right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

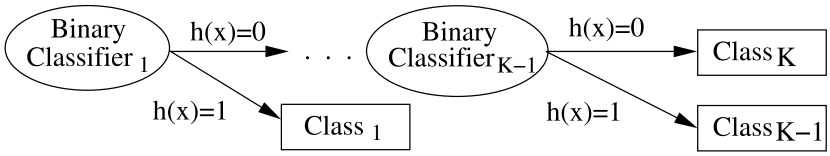



Figure 3.1.: A decision list classifier for $K$ classes using binary classifiers.

**Algorithm 2** Classification procedure for a decision list

**Input:** a new example $x$,
set of labels $Y = \{1, \ldots, K\}$,
set of binary classfiers $\{c_1, \ldots, c_{K-1}\}$

```
for k = 1, ..., K − 1 do
  if h_k(x) = 1 then
    return label k
  end if
end for
return label K
```

Suppose we have three binary classifiers $c_1$, $c_2$ and $c_3$ corresponding to the three classes $y_1$, $y_2$ and $y_3$. Now suppose that the classifiers $c_1$ and $c_2$ have an error $\epsilon_{1,2} = 0$ in the classification, whereas $c_3$ is a random classifier ($\epsilon_3 = 0.5$). We will start with a configuration in which the classifiers $c_1$, $c_2$ are situated in positions 1 and 2 in the decision list respectively and $c_3$ is situated in position 3. Now we want to classify a set of $n$ examples, from which $n_1$ examples pertain to class $y_1$, $n_2$ examples to class $y_2$ and $n_3$ examples to class $y_3$, so that $n = n_1 + n_2 + n_3$. In the first element of the list the $n$ examples are classified by $c_1$. The examples classified as positive are labeled as $y_1$ and the rest are passed to element 2. Because $c_1$ has an error $\epsilon_1 = 0$ the examples passed to the classifier $c_2$ are $n_2 + n_3$. In the second element of the list, the examples classified as positive by $c_2$ are labeled as $y_2$ and the rest are passed to the element 3. In this last element of the decision list, $n_3$ examples are classified by $c_3$ with an error of $\epsilon_3 = 0.5$. Thus, the final number of misclassified examples is $n_3 \cdot \epsilon_3$.

If we now change the configuration and place $c_3$ in the first position of the decison list, then the number of misclassifications in the first classification will be $n \cdot \epsilon_3$ , with $n = n_1 + n_2 + n_3$. Although the elements in positions 2 and 3 of the decison list, that is $c_1$ and $c_2$, have a classification errors $\epsilon_{1,2} = 0$, the final error of the decision list will be $n \cdot 0.5$, which is greater than $n_3 \cdot \epsilon_3$.

In general, the problem of finding the optimal order of binary classifiers that minimizes the classification error is NP-hard [33]. In our application, however, we are only ever confronted with a small number of classes, therefore we can easily enumerate all potential permutations to

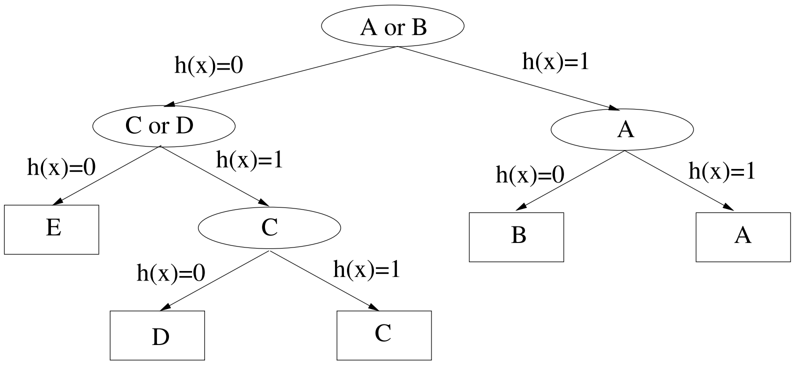



Figure 3.2.: An Or-Tree configuration for the classes $Y = \{A, B, C, D, E\}$.

determine the optimal sequence.

## 3.3. Or – Tree AdaBoost

A different structure for a multi-class classifier can be obtained if we arrange the binary classifiers in a decision tree in which some nodes try to recognize more than one class at the same time. Figure 3.2 shows the structure of a possible decision tree for classes $Y = \{A, B, C, D, E\}$. In this example the first node tries to recognize wether an example $x$ pertains to the classes $A$ or $B$. If this is the case, then the decision tree tries to detect the exact class $A$ or $B$ in the right descent node. If the example $x$ does not pertain to any of the classes $A$ or $B$, then it it passed to a node with different classifiers. Different configurations can be used for constructing decision trees, in which certain nodes can have two or more classifiers.

## 3.4. AdaBoost.M2

The previous Sections 3.2 and 3.3 explained how to use binary **AdaBoost** classifiers to construct a multi-class classifier. In this section we describe an extension of the **AdaBoost** algorithm to the multi-class case, called **AdaBoost.M2**, which was originally described by Freund and Shapire [9]. This extension generates multi-class hypotheses of the form $h : X \rightarrow Y = \{1, \ldots, K\}$ which can be used directly to classify an example into the different $K$ classes.

In **Adaboost.M2**, the weak learner **WeakLearn** generates hypotheses whose output is a vector in $[0, 1]^K$, rather than a single label in $Y$. The $y$th component of this vector represents the belief that the correct label is $y$. The components with values close to 1 are assigned to the more plausible labels. Likewise, labels considered implausible are assigned a value near 0, and questionable labels may be assigned a value near 1/2. To formalize the goal of the weak learner **WeakLearn**, a pseudoloss is defined which measures the accuracy of the weak hypotheses.

To explain this definition, we consider the following setup described by Freund and Shapire [9]. For each example $(x_i, y_i)$ we create a set of incorrect labels $I = Y - \{y_i\}$, which is composed by all labels except the correct one $y_i$. Therefore, the set $I$ contains $K-1$ labels. Now we use a given hypothesis $h$ to ask $K-1$ questions. In each question we compare the probability that the label of the example $x_i$ is the correct label $y_i$, with the probability that the label of $x_i$ is the incorrect label $y \in I$. For this purpose we split the hypothesis $h$ into $K$ weak binary hypotheses in the form:

$$h(x_i, y_i) = \begin{cases} h_1(x_i) \\ h_2(x_i) \\ \vdots \\ h_K(x_i) \end{cases} . \tag{3.3}$$

Here each binary hypothesis $h_y(x_i)$ with $y \in Y = \{1, \ldots, K\}$ is a function $h_y : X \to \{0, 1\}$ which determines wether $y$ is the label of $x_i$. If $h_y(x_i) = 0$ and $h_{y_i}(x_i) = 1$, then the label for $x_i$ is $y_i$. In the same way, if $h_y(x_i) = 1$ and $h_{y_i}(x_i) = 0$ then the label for $x_i$ is $y$. If $h_y(x_i) = h_{y_i}(x_i)$, then one of the two labels is chosen uniformly at random.

In the more general case where $h$ takes values in $[0, 1]$, the interpretation of $h(x, y)$ is a randomized decision. That is, we first choose a random bit $b(x, y)$ which is 1 with probability $h(x, y)$ and 0 otherwise. We then apply the above procedure to the stochastically chosen binary function $b$. According to Freund and Shapire [9], the probability of choosing the incorrect answer $y$ to the question above is:

$$\Pr[b(x_i, y_i) = 0 \wedge b(x_i, y) = 1] + \frac{1}{2}\Pr[b(x_i, y_i) = b(x_i, y)] = \frac{1}{2}(1 - h_{y_i}(x_i) + h_y(x_i)). \tag{3.4}$$

If the answers to all $K-1$ questions are considered equally important, then the loss of the hypothesis is defined to be the average over all $K-1$ questions:

$$\frac{1}{K-1}\sum_{y \in I}\frac{1}{2}(1 - h_{y_i}(x_i) + h_y(x_i)) = \frac{1}{2}\left(1 - h_{y_i}(x_i) + \frac{1}{K-1}\sum_{y \in I} h_y(x_i)\right). \tag{3.5}$$

We can also give different importance to each question depending on the situation, assigning a different weight to each question. So, for each instance $x_i$ and incorrect label $y \neq y_i$ , we assign a weight $q(i, y)$ which we associate with the question that discriminates label $y$ from the correct label $y_i$ . We then replace the average used in Equation 3.5 with an average weighted according to $q(i, y)$. The resulting formula, described by Freund and Shapire [9], is called the pseudoloss of $h$ on training instance $i$ with respect to $q$:

$$\text{ploss}_q(h, i) = \frac{1}{2}\left(1 - h_{y_i}(x_i) + \sum_{y \in I} q(i, y) h_y(x_i)\right). \tag{3.6}$$

The function $q : \{1, \ldots, N\} \times Y \rightarrow [0, 1]$ assigns to each example $x_i$ a probability distribution over the $K - 1$ discrimination problems defined above. So, for all $i$:

$$\sum_{y \in I} q(i, y) = 1. \tag{3.7}$$

The complete $\mathbf{AdaBoost.M2}$ algorithm is shown in Algorithm 3. The weights $w^t_{i,y}$ are maintained for each instance $x_i$ and each label $y \in I$. The weak learner $\mathbf{WeakLearn}$ must be provided with a distribution $D_t$ and a label weight function $q_t$. Both of these are computed using the weight vector $\mathbf{w}^t$ as shown in Steps 1-3. The weak learner's goal is then to minimize the pseudoloss $\epsilon_t$, as defined in Step 5. The weights are updated as shown in Step 7. The final hypothesis $h_f$ outputs, for a given instance $x$, the label $y$ that maximizes a weighted average of the weak hypothesis values $h^t_y(x)$.

Freund and Schapire [9] proved that, for multi-class classification problems, the training error of the final hypothesis $h_f$ generated by $\mathbf{AdaBoost.M2}$ is bounded by:

$$\epsilon \leq (k - 1)2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1 - \epsilon_t)} \leq (k - 1)\exp\left(-2 \sum_{t=1}^{T} \gamma_t^2\right), \tag{3.8}$$

where $\epsilon_t$ is the pseudo-loss of the $t$th weak classifier. As we can see the error increases linearly with the number of classes if we compare it with the error of the binary case (Equation 3.2).

To summarize, this chapter described the $\mathbf{AdaBoost}$ boosting algorithm. In its original form, this algorithm was designed for binary classification. Therefore, three extensions of $\mathbf{AdaBoost}$ for multi-class classification were presented: $\mathbf{Sequential\ AdaBoost}$, $\mathbf{Or - TreeAdaBoost}$ and $\mathbf{AdaBoost.M2}$.

**Algorithm 3** AdaBoost.M2

**Input:** sequence of $N$ labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$ with labels $y_i \in Y = \{1, \ldots, K\}$,
distribution $\mathcal{D}$ over the $N$ examples,
weak learning algorithm **WeakLearn**,
integer $T$ specifying number of iterations.

**Initialize** the weight vector $w^1_{i,y} = D(i)/(K-1)$ for $i = 1, \ldots, N, y \in Y - \{y_i\}$.

**for** $t = 1, \ldots, T$ **do**

1. Set
$$W^t_i = \sum_{y \neq y_i} w^t_{i,y}$$

2. Set
$$q_t(i, y) = \frac{w^t_{i,y}}{W^t_i}$$
for $y \neq y_i$.

3. Set
$$D_t(i) = \frac{W^t_i}{\sum_{i=i}^{N} W^t_i}.$$

4. Call **WeakLearn** providing it with the distribution $D_t$ and label weighting function $q_t$. Get back a hypothesis $h^t : X \times Y \to [0, 1]$.

5. Calculate the pseudo-loss of $h^t$:
$$\epsilon_j = \frac{1}{2} \sum_{i=1}^{n} D_t(i) \left( 1 - h^t_{y_i}(x_i) + \sum_{y \neq y_i} q_t(i, y) h^t_y(x_i) \right).$$

6. Set
$$\beta_t = \frac{\epsilon_t}{(1 - \epsilon_t)}$$

7. Set the new weights vector
$$w^{t+1}_{i,y} = w^t_{i,y} \beta_t^{(1/2)(1 + h^t_{y_i}(x_i) - h^t_y(x_i))}$$
for $i = 1, \ldots, N$ and $y \in Y - \{y_i\}$.

**end for**

**Output:** The hypothesis
$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left( log \frac{1}{\beta_t} \right) h^t_y(x).$$

# 4. Learning Indoor Places from Range Data

## 4.1. Description of the Learning Task

This section describes the learning task of this work: the learning of indoor places using laser range data.

As introduced in Chapter 1, indoor environments are typically distributed in different living and working spaces. Each of those places has a different functionality and structure. An example of an indoor environment is shown in Figure 4.1. Most of these places have a similar structure, but in different environments. For example, a corridor is usually longer, a room is more square, doorways are small, etc.

The aim of this work is to build a system in a mobile robot, so that the robot is able to recognize the place where it is. For example, if the robot is in a corridor and we ask it, where it is, the robot must be able to answer “corridor”. Figure 4.2 explains this situation.

Here we describe this problem as a learning task. According to Section 2.2, this task can be formally described as follows:

- The space of possible instances $X$ is composed of the poses of the robot in the environment. Each instance $x$ is represented by one observation $z$ made by the robot, and one label $y \in Y$ representing the place in which the pose is. In this work we will restrict the set to:

$$Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}, \mathbf{Hallway}\}. \tag{4.1}$$

  As we will see later in Section 4.2, each observation is composed of a laser range scan together with certain features that we calculated from this scan.

- Our target concept $c \in \mathcal{C}$ is represented by a function $c : X \times Y \rightarrow \{1, 0\}$ of the form:

$$c(x, y) = \begin{cases} 1 & \text{if } x \text{ is in place labeled } y \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

- Each hypothesis $h \in H$ has the form:

$$h(x, y) = \begin{cases} 1 & \text{if } x \text{ is classified as } y \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

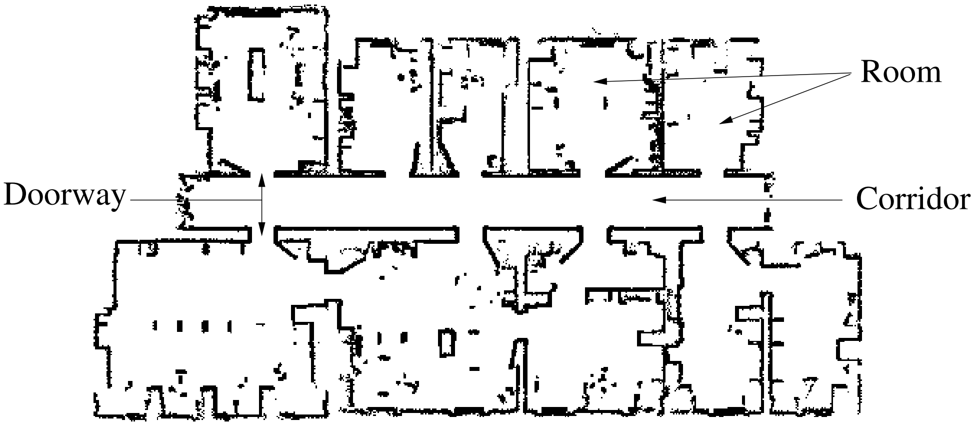



Figure 4.1.: Example environment containing rooms, doorways and a corridor.

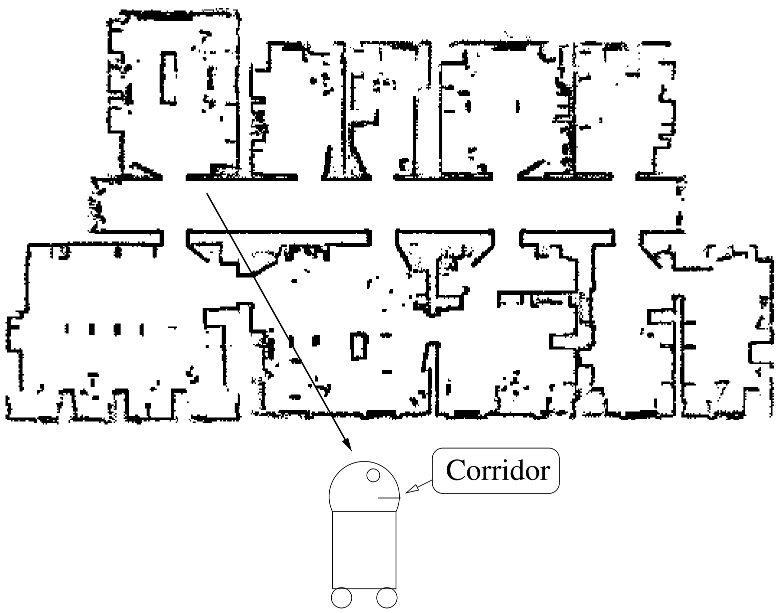



Figure 4.2.: After taking an observation, a robot classifies its pose as **Corridor**.

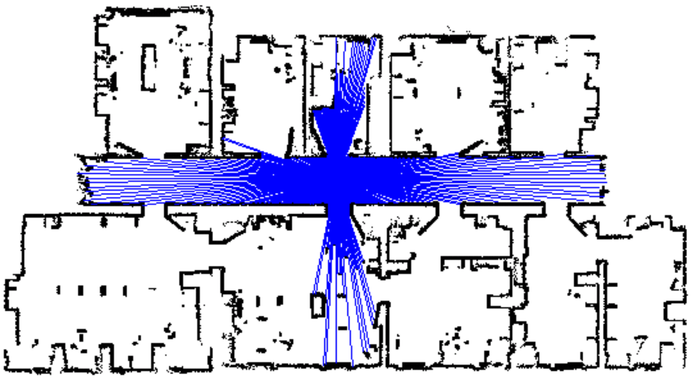


Figure 4.3.: Beams obtained by the robot in a corridor.

- The learner $L$ generates a final hypothesis $h_f$ which is used for subsequent classifications. For the implementation of the learner $L$ we will use an adaptation of two different algorithms: $\mathbf{AdaBoost}.\mathbf{M2}$ and $\mathbf{Sequential\ AdaBoost}$, which we will describe in detail in Sections 4.3 and 4.4.

## 4.2. Features from Laser Range Scans

This section describes the set of simple features that are calculated from each observation using the beams of a laser range scan. These features will be used later as weak hypotheses to form a final strong classifier.

As we showed in Section 4.1, the aim of this work is to create a classifier for a mobile robot in such a way that it is able to recognize each different place. We assume that the robot is equipped only with a 360$^o$ field of view range laser. Therefore, the final classifier is constructed using the data provided by the laser, without any other sensor information. That means, the observation made by the robot at a specific pose is composed only of the set of beams from the laser range. Formally, each observation $z = \{b_0, ..., b_{M-1}\}$ contains a set of $M$ beams $b_i$. Each beam $b_i$ consists of a tuple $(\alpha_i, d_i)$ where $\alpha_i$ is the angle of the beam relative to the robot, and $d_i$ is the length of the beam. Each observation used in this wotk has anumber of $M = 360$ beams, with an angle step of $1^o$. An example of the beams taken by the robot in a corridor is shown in Figure 4.3.

To correctly classify a pose in an indoor environment, the robot must classify the observation $z$ obtained in this pose into one of the places $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}, \mathbf{Hallway}\}$ that we want to learn. As an example, Figure 4.4 shows three observations taken in different places in an indoor environment: in a room, a doorway and a corridor. In the first instances, it seems hard to find features in these beams that can be related with the place where they were taken from. But by looking more carefully, we can find some geometric characteristics which usually appear in the beams taken from the same place. For example, looking at the range taken in the corridor (Figure 4.4(a)), a central horizontal bounding box can be extracted, which represents the corridor. Moreover, two consecutive beams inside this rectangular box have a small difference in length, due to the continuity of the walls that form a corridor. We can also find two perpendicular *ears* corresponding to two doors that were open in the moment of registering the beams. On the other hand, the beams taken inside a room (Figure 4.4(b)) are more scattered than the ones from a corridor. This is due to the different objects that we can find inside like tables, chairs or coat stands. As a consequence more *gaps* appear in the scan. A gap is a difference in length between two consecutive beams considered as significant. With respect to the beams obtained from a doorway (Figure 4.4(c)), two opposite sets exist with length significantly smaller than others. These beams represent the doorframe. Figure 4.5 graphically shows the different characteristics obtained from the three set of beams.

Thus, it seems reasonable to represent each observation $z$ not by the raw beams $\{b_0, ..., b_{M-1}\}$, but by a set of geometrical features which can be more representative of the complete range scan. In this work we use a set of single-valued geometrical features that are calculated from each set of beams that compose an observation. Formally, we define a feature $f$ as a function that takes as an argument one observation and returns a real value:$f : Z \rightarrow \mathbb{R}$, where $Z$ is the set of all possible observations. Apart from that, it is desirable that these features are rotational invariant. The reason for this constraint is that we are interested in classifying the pose where the robot is, independently of its orientation. Most of the features we use in this work are standard geometrical features often used in shape analysis (see Gonzalez and Wintz [11], Haralick and Shapiro [13], Loncaric [22], O'Rourke [27], and Russ [29]).

Two sets of single-valued features are calculated from each observation. The first set $B$ is calculated using the raw beams in $z$ as shown in Figure 4.4. The following is a list of the single-valued features pertaining to this set:

1. The average difference between the length of consecutive beams.

2. The standard deviation of the difference between the length of consecutive beams.

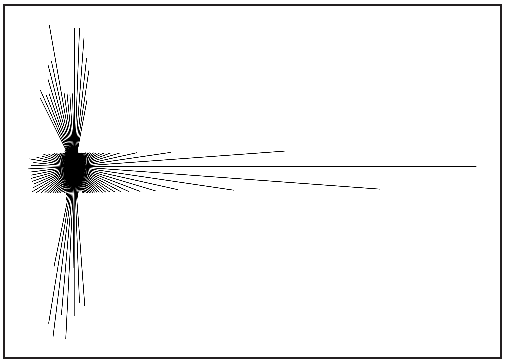
3. Same as 1), but considering different max-range values.

4. The average beam length.

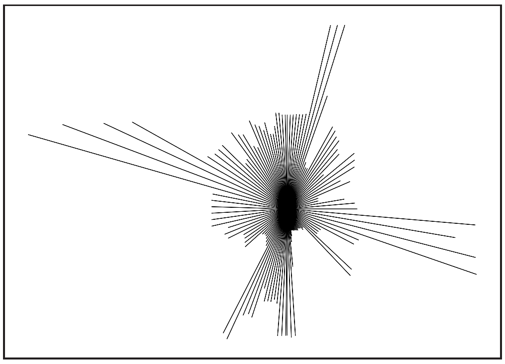
5. The standard deviation of the length of the beams.

6. Number of gaps in the scan. Two consecutive beams build a gap if their difference is greater than a given threshold. Different features are used for different threshold values.

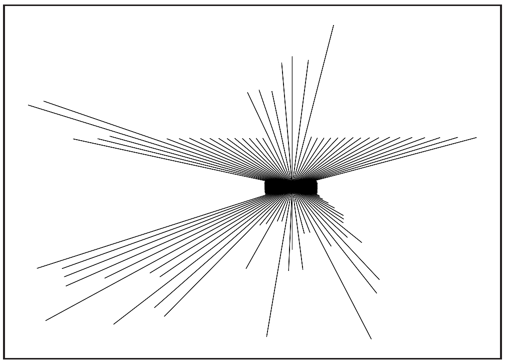
7. Number of beams lying on lines that are extracted from the range scan (see Sack and Burgard [30]).



(a) Corridor

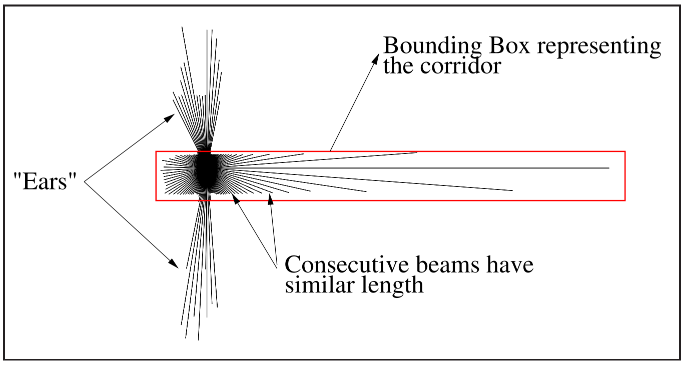


(b) Room

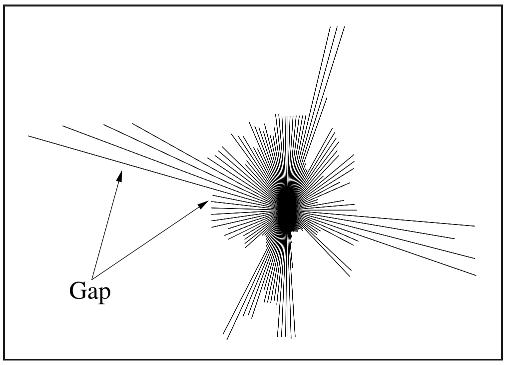


(c) Doorway

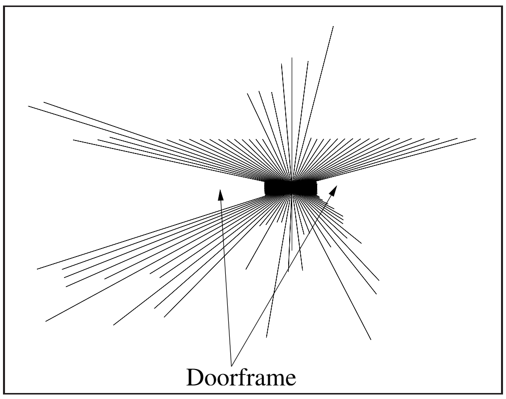
Figure 4.4.: Example raw beams recorded in a corridor, a room and a doorway.




(a) Corridor




(b) Room




(c) Doorway

Figure 4.5.: Possible features extracted from the raw beams of Figure 4.4.

8. Euclidean distance between the two points corresponding to the two consecutive local minima.

9. The angular distance between the beams corresponding to the two consecutive local minima in feature 8).

A second set $P$ of features is calculated from a polygonal approximation $\mathbf{P}(z)$ of the area covered by the observation $z$. The vertices of the closed polygon $\mathbf{P}(z)$ correspond to the coordinates of the end-points of each beam $b_i$ of $z$ relative to the robot:

$$\mathbf{P}(z) \quad = \quad \{(d_i \cos \alpha_i, d_i \sin \alpha_i) \mid i = 0, \ldots, M-1\} \tag{4.4}$$

As an example, the polygonal representations of the laser range scans depicted in Figure 4.4 are shown in Figure 4.6. The following is a list containing the single-valued features pertaining to the $P$ set:

1. Area of $\mathbf{P}(z)$.

2. Perimeter of $\mathbf{P}(z)$.

3. Area of $\mathbf{P}(z)$ divided by Perimeter of $\mathbf{P}(z)$.

4. Mean distance between the centroid to the shape boundary.

5. Standard deviation of the distances between the centroid to the shape boundary.

6. 200 similarity invariant descriptors based in the Fourier transformation.

7. Major axis $\mathrm{Ma}$ of the ellipse that approximates $\mathbf{P}(z)$ using the first two Fourier coefficients.

8. Minor axis $\mathrm{Mi}$ of the ellipse that approximate $\mathbf{P}(z)$ using the first two Fourier coefficients.

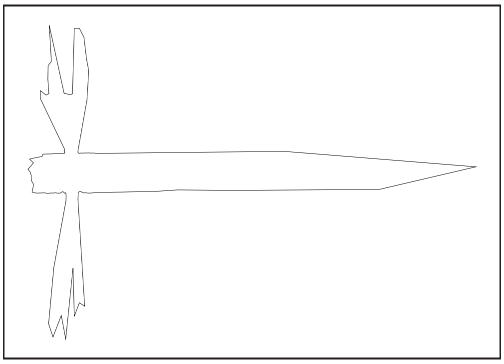
9. $\mathrm{Ma}/\mathrm{Mi}$.

10. Seven invariants calculated from the central moments of $\mathbf{P}(z)$.

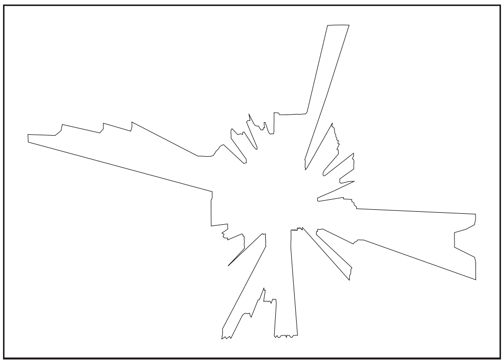
11. Normalized feature of compactness of $\mathbf{P}(z)$.

12. Normalized feature of eccentricity of $\mathbf{P}(z)$.

13. Form factor of $\mathbf{P}(z)$.



(a) Corridor



(b) Room

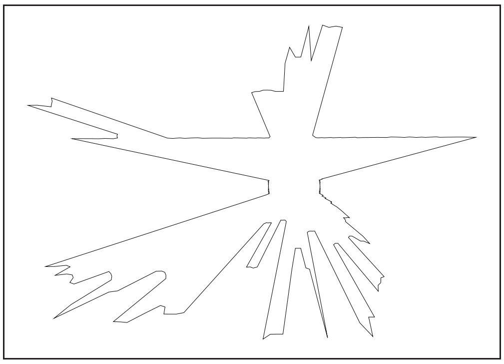


Figure 4.6.: Polygonal representations of the scans shown in Figure 4.4.

Appendix A.1 and A.2 describe in detail how to calculate each of the features for both $B$ and $P$ sets.

Finally, each training example $x_i$ is represented by a set of $J$ features $\{f_1, \ldots, f_J\}$ together with its label $y_i$, in the form:

$$x_i = (\{f_1, \ldots, f_J\}, y_i); \text{ with } y_i \in Y \text{ and } f_j \in \{B \cup P\}. \tag{4.5}$$

Here $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}, \mathbf{Hallway}\}$ is the set of classes that we want to learn. We assume that the label of the training examples is given in advance. In practice this can be achieved by manually labeling places in the map as we will see in Section 5.1, or instructing the robot while it is exploring its environment.

## 4.3. The Final Sequential AdaBoost Algorithm

As shown in Section 3.2, a multi-class classifier can be constructed linking several binary classifiers into a decision list. To create each of the binary classifiers needed in this work we apply the variant of the **AdaBoost** algorithm presented by Viola and Jones [42]. This variant restricts the weak classifiers to depend on single-valued features only. In our case we will use the two sets $B$ and $P$ of features from Section 4.2.

Following the notation from Viola and Jones [42], each weak classifier $h_j(x)$ consists of a feature $f_j$, a threshold $\theta_j$ and a parity $p_j$ indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

As an example, a weak classifier $h_{\text{Area}}$ created from the feature "Area of $\mathbf{P}(x)$", $f_{\text{Area}}(x)$, will have the form:

$$h_{\text{Area}}(x) = \begin{cases} 1 & \text{if } f_{\text{Area}}(x) > \theta_{\text{Area}} \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

meaning that if the area of $\mathbf{P}(x)$ is bigger than the threshold $\theta_{\text{Area}}$, then the example $x$ will be classified as positive.

The parameters $\theta$ and $p$ must be learned by each weak classifier in the training process. One possible method for learning these parameters for a specific weak classifier $h_j$ with feature $f_j$ and set of $N$ examples consists of creating a search space $S_j$ whose elements are the pair values $(\theta_i, p_i)$ with $\theta_i = f_j(x_i)$, in the form:

$$S_j = \{(\theta_1, -1), (\theta_1, +1), \ldots, (\theta_N, -1), (\theta_N, +1)\} \tag{4.8}$$

Now we have only to select each pair $(\theta_i, p_i)$ and compute the number of misclassifications using the Expression 4.6. Finally, we select the parameters which implies lower number of misclassifications. Formally:

$$(\theta^*, p^*) = \arg\min_{\theta,p} \sum_{i=1}^{N} |h_j(x_i) - y_i| \tag{4.9}$$

with

$$h_j(x_i) = \begin{cases} 1 & \text{if } p \cdot f_j(x_i) < p \cdot \theta \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

This process can be carried on in linear time is the elements of $S_j$ are previously ordered by the values $\theta_i$.

Once we have a method for training weak classifiers we can implement the final version of the binary **AdaBoost** algorithm which is shown in Algorithm 4. As we can see, Steps 2 and 3 correspond to the **WeakLearn** algorithm in the original **AdaBoost** algorithm (see Algorithm 1). In this case a brute force search is done among all the possible hypotheses and then, the one with lowest error is selected.

To create the final **Sequential AdaBoost** algorithm we only have to arrange a decision list in the same way as explained in Section 3.2.

## 4.4. The Final AdaBoost.M2 Algorithm

As we showed in Section 3.4, the **AdaBoost.M2** algorithm creates a final strong classifier which is able to discrimate among $K$ different classes. In our case, each multi-class weak hypothesis $h_j(x, y)$ with feature $f_j$ is split into $K$ binary hypotheses of the form:

$$h_j(x_i, y_i) = \begin{cases} h_{j,1}(x) \\ h_{j,2}(x) \\ \vdots \\ h_{j,K}(x) \end{cases} \tag{4.11}$$

with

$$h_{j,k}(x_i) = \begin{cases} 1 & \text{if } p_j \cdot f_j(x_i) < p_j \cdot \theta_j \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

To train the multi-class hypothesis $h_j(x_i, y_i)$ we train each of the binary hypotheses $h_{j,k}(x_i)$ using the same method as in Section 4.3, but using as positive training examples thoses with label $k$ and the rest as negatives. Formally:

$$(\theta^*, p^*) = \arg\min_{\theta,p} \sum_{i=1}^{N} \left|h_{j,k}(x_i) - y'_i\right| \tag{4.13}$$

with

$$h_{j,k}(x_i) = \begin{cases} 1 & \text{if } p \cdot f_j(x_i) < p \cdot \theta \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

and

$$y_i' = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k. \end{cases} \tag{4.15}$$

The complete final $\mathbf{AdaBoost.M2}$ algorithm is shown in Algorithm 5.

This chapter presented the learning task of this thesis: the supervised learning of places from range data using boosting. Moreover, the different simple features extracted from proximity data were described. These features will be used as weak hypotheses by the boosting procedure. Finally, the particular versions of $\mathbf{Sequential\ AdaBoost}$ and $\mathbf{AdaBoost.M2}$ for this work were introduced.

**Algorithm 4** AdaBoost version by Viola and Jones

**Input:** sequence of $N$ labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$ where $y_i = 1$ for positive examples and $y_i = 0$ for negative examples,
a set of features $f_1, \ldots, f_J$,
integer $T$ specifying the number of iterations.

**Initialize** the weight vector $w_i^1 = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where $m$ is the number of negative examples and $l$ is the number of positve examples.

**for** $t = 1, \ldots, T$ **do**

1. Set
$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. For each feature $f_j$, train a weak classifier $h_j$ which is restricted to using this single feature. The error of $h_j$ is evaluated with respect $\mathbf{p}^t$:
$$\epsilon_j = \sum_{i=1}^N p_i^t \, |h_j(x_i) - y_i|$$

3. Choose the classifier $h_t$ with lowest error $\epsilon_t$.

4. Set
$$\beta_t = \frac{\epsilon_t}{(1 - \epsilon_t)}$$

5. Set the new weights:
$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$

**end for**

**Output:** The final strong hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \left(\log \frac{1}{\beta_t}\right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

**Algorithm 5** Final AdaBoost.M2 algorithm

**Input:** sequence of $N$ labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$ with labels $y_i \in Y = \{1, \ldots, k\}$, integer $T$ specifying number of iterations.

**Initialize** the weight vector $w^1_{i,y} = D(i)/(k-1)$ for $i = 1, \ldots, N, y \in Y - \{y_i\}$, where $D(i) = \frac{1}{2m}$ and $m$ is the number of training examples with label $y_i$.

**for** $t = 1, \ldots, T$ **do**

1. Set
$$W^t_i = \sum_{y \neq y_i} w^t_{x,y}$$

2. Set
$$q_t(i, y) = \frac{w^t_{i,y}}{W^t_i}$$
for $y \neq y_i$.

3. Set
$$D_t(i) = \frac{W^t_i}{\sum_{i=i}^{N} W^t_i}.$$

4. For each feature $j$ train multi-class weak classifier $h_j(x, y)$ which is restricted to using the single feature $j$. The error is evaluated with respect $D(i)$:
$$\epsilon_j = \frac{1}{2} \sum_{i=1}^{n} D_t(i) \left( 1 - h_{j,y_i}(x_i) + \sum_{y \neq y_i} q_t(i, y) h_{j,y}(x_i) \right).$$

5. Choose the classifier $h^t$ with lower error $\epsilon_t$.

6. Set
$$\beta_t = \frac{\epsilon_t}{(1 - \epsilon_t)}.$$

7. Set the new weights vector
$$w^{t+1}_{i,y} = w^t_{i,y} \beta_t^{(1/2)(1 + h^t_{y_i}(x_i) - h^t_y(x_i))}$$
for $i = 1, \ldots, N$ and $y \in Y - \{y_i\}$.

**end for**

**Output:** The hypothesis
$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left( log \frac{1}{\beta_t} \right) h^t_y(x).$$

# 5. Experiments

The aim of the following experiments is to demonstrate that the single-valued features described in Section 4.2 can be boosted to a robust classifier which is able to recognize different places in an indoor environment. Additionally we analyze whether the resulting classifier can be used to classify places in environments for which no training data was available.

## 5.1. Experimental Setup

In order to test our approach on real data as well as in simulation, we use the Carnegie Mellon Robot Navigation Toolkit (CARMEN) [25]. This simulation environment allows us to navigate with a robot in different maps whereas recording all sensor information (Figure 5.1). The robot used in the simulation was an ActivMedia Pioneer 2-DX8 equipped with two SICK laser range finders as depicted in Figure 5.2. To generate the training and test set of examples, a specific software was developed which communicates with the CARMEN simulator and permits the selection of different poses where the robot will have to take the observations. A more detailed description of this software is given in Appendix B.

## 5.2. Results with $\text{Sequential AdaBoost}$

One important parameter of the $\mathbf{AdaBoost}$ as well as the $\mathbf{AdaBoost}.\mathbf{M2}$ algorithm is the number of weak classifiers $T$ used to form the final strong classifier. We performed several experiments with different numbers of weak classifiers and analyzed the classification error. Throughout our experiments, we found that anumber of $100$ weak classifiers provides a low error in the classification when using a robot in real time. Therefore, this value is used in all experiments presented in this work.

The first experiment was performed using data from our office environment in Building 79 at the University of Freiburg (see Figure 5.4(a)). This environment was divided into three different types of places: room, doorway, and corridor. For the sake of clarity we present our results by separating the environment into two parts. The left half of the environment contains the training examples (see Figure 5.4(b)), and the right half of the environment was then used as a test set (see Figure 5.4(c)).

In this experiment we used the sequential classifier shown in Figure 3.1. We tested all the possible combinations of binary classifiers with labels $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$

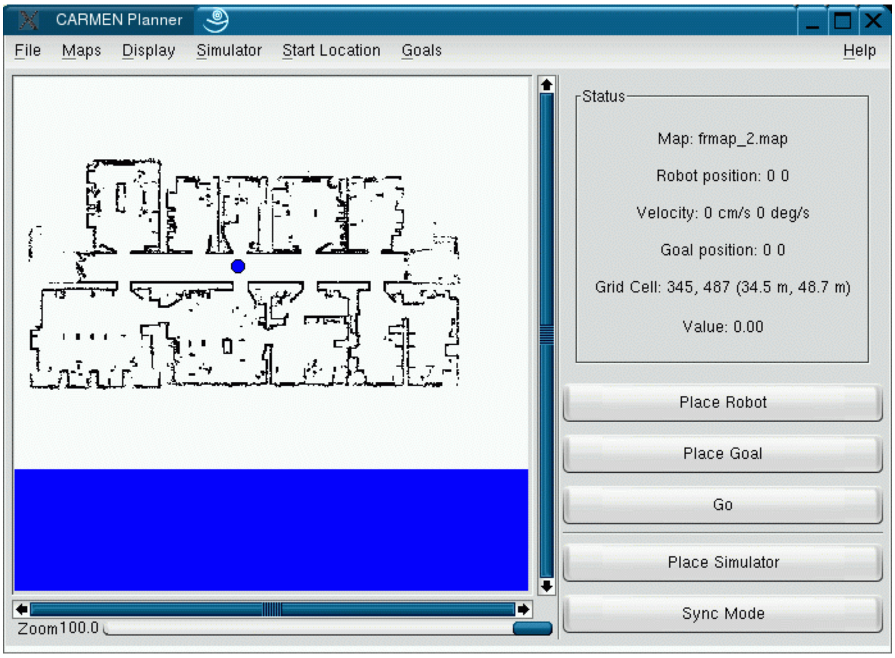



Figure 5.1.: Interface of the Carnegie Mellon Robot Navigation Toolkit (CARMEN).


Figure 5.2.: The robot used in the experiments: a Pioneer 2-DX8 equipped with two SICK laser range finders.

in a decision list, and calculated the success of the final classification for each of them. Table 5.1 shows the correct classification rate for all the possible decision lists. According to this table the optimal decision list for this classification problem is given by the sequence **Room** – **Doorway** whose configuration is shown in Figure 5.3. This decision list correctly classifies 93.94% of all test examples. The classification results are also depicted as colored/grey-shaded areas in Figure 5.4(c).

| Classifier Sequence | Correct Classifications % |
|---|---|
| **Room** – **Doorway** | 93.94 |
| **Room** – **Corridor** | 93.31 |
| **Corridor** – **Room** | 93.16 |
| **Doorway** – **Corridor** | 80.68 |
| **Doorway** – **Room** | 80.49 |
| **Corridor** – **Doorway** | 80.10 |

Table 5.1.: Percentage of correctly classified examples for the 6 configurations of a sequential multi-class classifier with $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$ used in the map of Building 79 (see Figure 5.4).

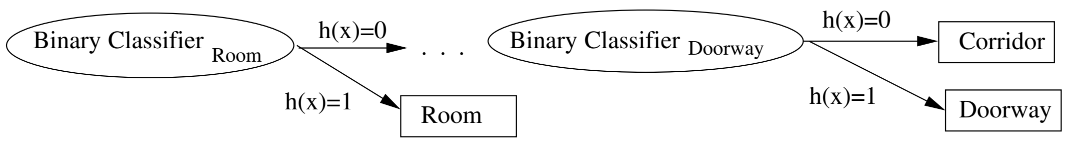



Figure 5.3.: The best decision list configuration for the classification of building 79 at the University of Freiburg.

In a second experiment we repeat the process but use the map of Building 52 at the University of Freiburg (see Figure 5.5(a)). Table 5.2 shows the results for the different configurations using the set of labels $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$. Figure 5.5 depicts the training and test classification for Building 52 using the best decision list **Room** – **Corridor** according to Table 5.2.

A third experiment was performed using a map containing four different classes of places $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}, \mathbf{Hallway}\}$. The complete map is shown in Figure 5.6(a). Table 5.3 shows the resulting classification of all the possible decision lists. The optimal decision list for this experiment was **Corridor** – **Hallway** – **Doorway**, with a success rate of 89.52%.

It is also interesting to calculate the error of the different binary classifiers that form the fi-

| Classifier Sequence | Correct Classifications % |
|---|---|
| **Room – Corridor** | 92.10 |
| **Corridor – Room** | 91.57 |
| **Doorway – Corridor** | 91.13 |
| **Corridor – Doorway** | 91.03 |
| **Room – Doorway** | 90.94 |
| **Doorway – Room** | 90.30 |

Table 5.2.: Percentage of correctly classified examples for the 6 configurations of a sequential multi-class classifier with $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$ used in the map of Figure 5.5(a).

nal sequential classifier. Table 5.4 contains the error rates of the individual binary classifiers on the training data of map in Figure 5.6. The error rates differ between .7% and 1.5%. The binary **Doorway** classifier yields the highest error. We believe that this is due to several factors. First, a doorway is typically a very small region so that only a few training examples are available. Furthermore, if a robot stands in a doorway the scan typically covers nearby rooms or corridors which makes it hard to distinguish the doorway from such places. If we compare the classification rates of the **Sequential AdaBoost** for four classes in Table 5.3 and the errors of the binary classfiers shown in Table 5.4, we can see that most of the decision list with best results (the first twelve) have as first binary classifier the one with less error, that is, **Corridor** or **Hallway**. This coincides with our strategy from Section 3.2, in which we explained that it is a good heuristic to place the binary classifiers with less error in the first positions of the decision list.

## 5.3. Results with Or – Tree AdaBoost

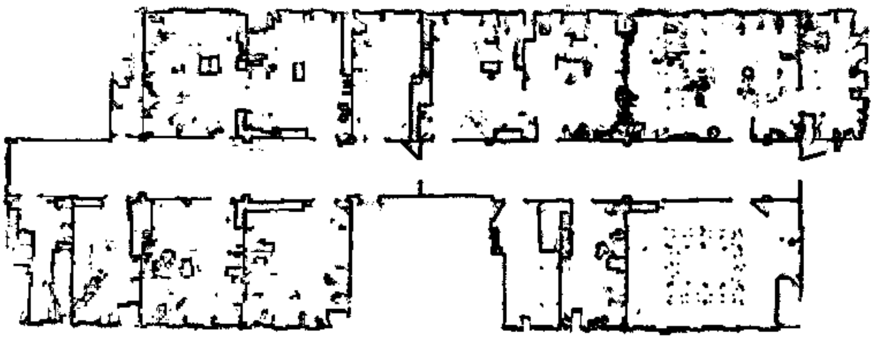
This experiment was designed to compare the two types of structures for a multi-class classifier using binary classifiers: **Sequential Adaboost** and **Or – Tree Adaboost**. Figure 5.7 shows the training and test maps used in this experiment. Table 5.5 shows the results of the classification of the test set for the different configurations of a sequential classifier using labels $Y = \{\mathbf{Corrdior}, \mathbf{Room}, \mathbf{Doorway}\}$. Table 5.6 depicts the results in the classfication of different configurations for an Or-tree classifier. As we can see the classification rates are very similar. This is due to the fact that an Or-tree configuration is similar to a sequential configuration if the first two binary classfifiers in the decision list are used in the first node of the Or-tree.

| Classifier Sequence | Correct Classifications % |
|---|---|
| **Corridor − Hallway − Doorway** | 89.52 |
| **Corridor − Room − Hallway** | 89.41 |
| **Corridor − Hallway − Room** | 89.36 |
| **Room − Corridor − Hallway** | 89.29 |
| **Hallway − Room − Doorway** | 88.95 |
| **Room − Hallway − Doorway** | 88.78 |
| **Hallway − Corridor − Doorway** | 88.69 |
| **Hallway − Doorway − Corridor** | 88.68 |
| **Hallway − Doorway − Room** | 88.59 |
| **Hallway − Corridor − Room** | 88.53 |
| **Hallway − Room − Corridor** | 88.53 |
| **Room − Hallway − Corridor** | 88.36 |
| **Corridor − Room − Doorway** | 86.88 |
| **Room − Corridor − Doorway** | 86.81 |
| **Room − Doorway − Corridor** | 86.80 |
| **Corridor − Doorway − Room** | 86.60 |
| **Doorway − Room − Corridor** | 86.59 |
| **Doorway − Corridor − Room** | 86.57 |
| **Corridor − Doorway − Hallway** | 85.82 |
| **Doorway − Corridor − Hallway** | 85.74 |
| **Room − Doorway − Hallway** | 84.98 |
| **Doorway − Hallway − Room** | 84.76 |
| **Doorway − Hallway − Corridor** | 84.75 |
| **Doorway − Room − Hallway** | 84.68 |

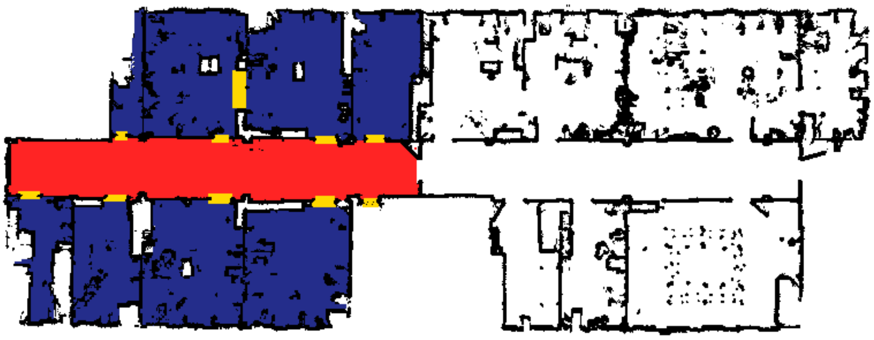
Table 5.3.: Percentage of correctly classified examples for the 24 configurations of a sequential multi-class classifier with labels $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}, \mathbf{Hallway}\}$ for the map in Figure 5.6(a).

| Binary Classifier | Training error % |
|---|---|
| **Corridor** | 0.7 |
| **Hallway** | 0.7 |
| **Room** | 1.4 |
| **Doorway** | 1.5 |

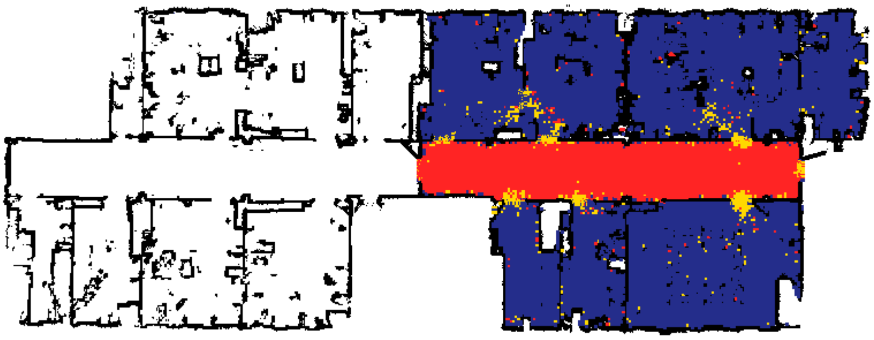
Table 5.4.: Error in the training data for the individual binary classifiers learned from the map depicted in Figure 5.6.



(a) Map of building 79



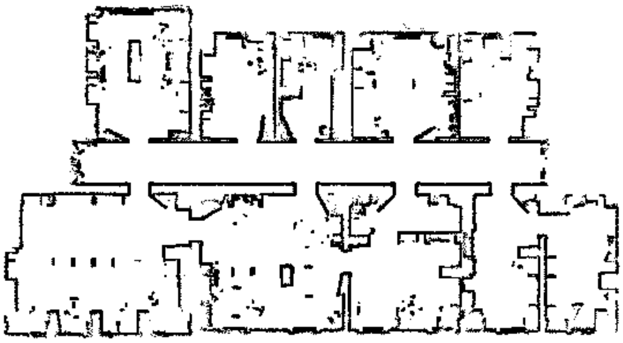
(b) Training examples



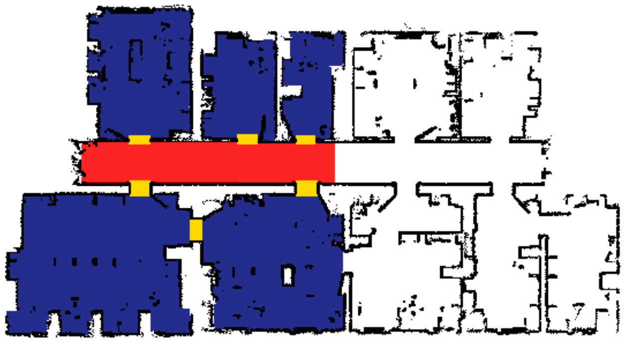
(c) Classification of test examples

Corridor Room Doorway

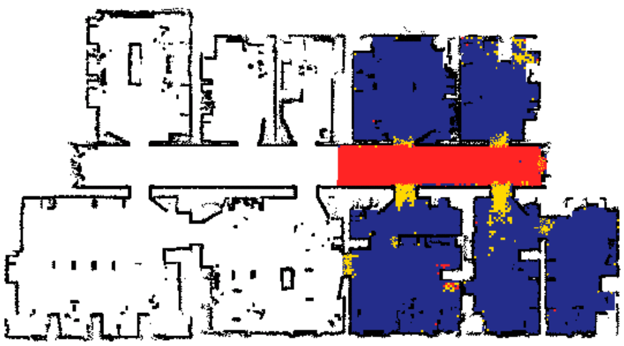
Figure 5.4.: The map of Building 79 at the University of Freiburg is shown in Figure 5.4(a). Figure 5.4(b) shows the training set. Finally, Figure 5.4(c) shows the classified test data using the best sequential classifier $\mathbf{Room} - \mathbf{Doorway}$.



(a) Map of building 52

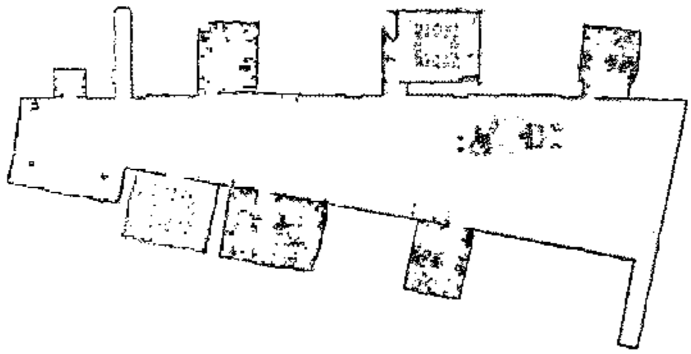


(b) Training examples

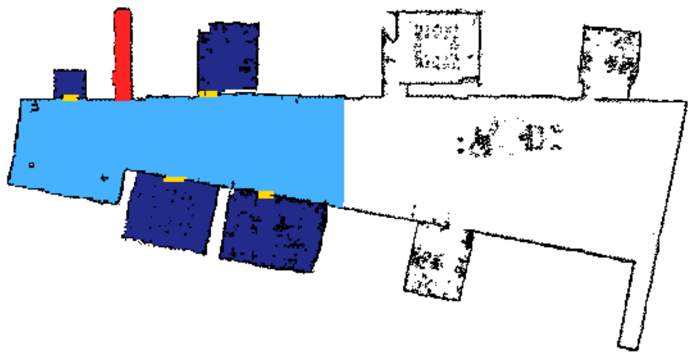


(c) Classification of test examples

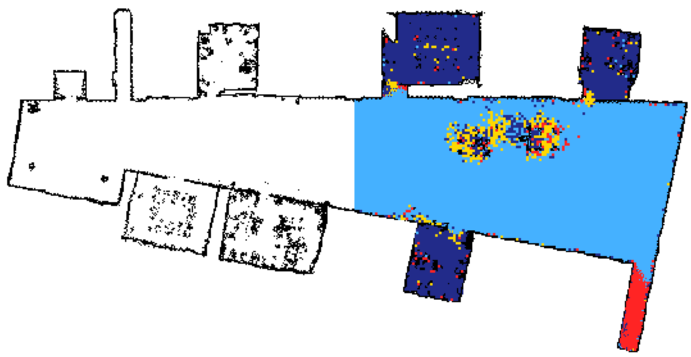

Figure 5.5.: The map of Building 52 at the University of Freburg is shown in Figure 5.5(a). Figure 5.5(b) shows the training set and Figure 5.5(c) shows the classified test data using the best sequential classifier **Room − Corridor**.



(a) Complete Map



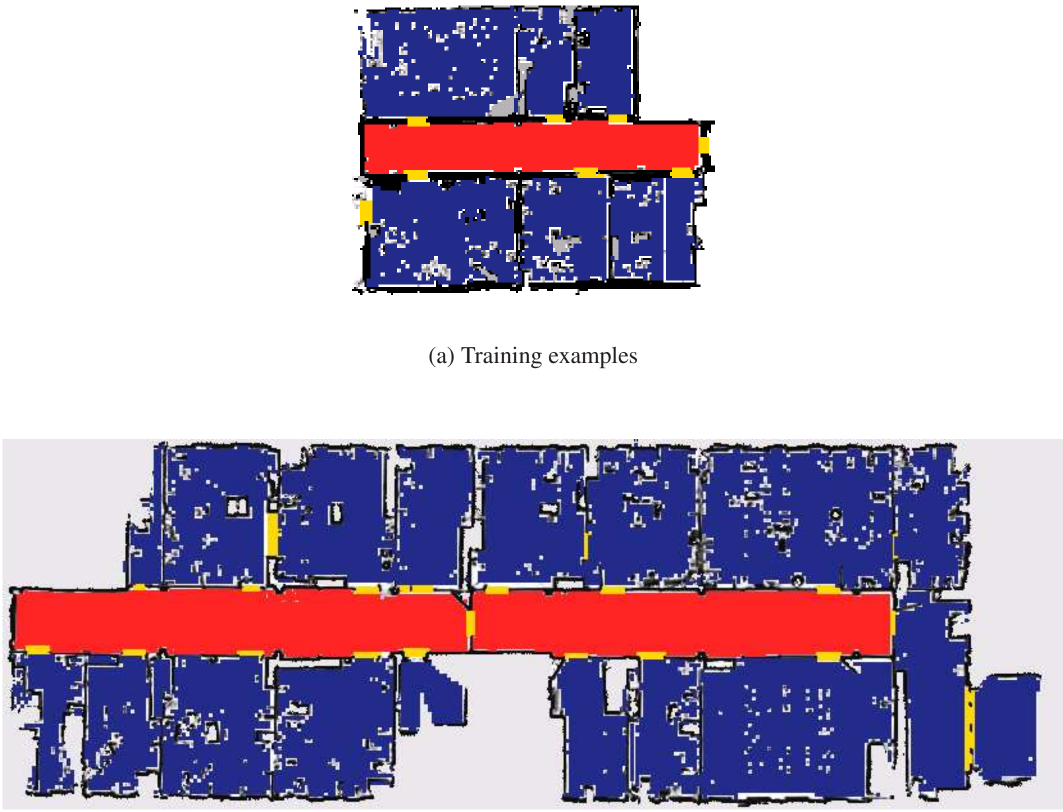
(b) Training examples



(c) Test classification

Corridor Room Doorway Hallway

Figure 5.6.: The complete map with 4 types of places is shown in Figure 5.6(a). Figure 5.6(b) shows the training set, and Figure 5.6(c) shows the classified test data with the best sequential configuration **Corridor** – **Hallway** – **Doorway**.



(a) Training examples

(b) Test examples

Corridor Room Doorway

Figure 5.7.: Training and test sets used for the comparison of the Or-tree and sequential configurations.

| Classifier Sequence | Correct Classifications % |
|---|---|
| $\mathbf{Corridor} - \mathbf{Room}$ | 88.83 |
| $\mathbf{Room} - \mathbf{Corridor}$ | 88.82 |
| $\mathbf{Corridor} - \mathbf{Doorway}$ | 88.33 |
| $\mathbf{Room} - \mathbf{Doorway}$ | 87.04 |
| $\mathbf{Doorway} - \mathbf{Room}$ | 86.17 |
| $\mathbf{Doorway} - \mathbf{Corridor}$ | 85.82 |

Table 5.5.: Percentage of correctly classified examples for the 6 configurations of a sequential multi-class classifier with $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$ using the data from Figure 5.7.

| Classifier Tree | Correct Classifications % |
|---|---|
| $(\mathbf{Room\ or\ Corridor}) - \mathbf{Doorway}$ | 88.82 |
| $(\mathbf{Corridor\ or\ Room}) - \mathbf{Doorway}$ | 88.72 |
| $(\mathbf{Corridor\ or\ Doorway}) - \mathbf{Room}$ | 87.91 |
| $(\mathbf{Room\ or\ Doorway}) - \mathbf{Corridor}$ | 86.94 |
| $(\mathbf{Doorway\ or\ Room}) - \mathbf{Corridor}$ | 86.19 |
| $(\mathbf{Doorway\ or\ Corridor}) - \mathbf{Room}$ | 85.74 |

Table 5.6.: Percentage of correctly classified examples for 6 configurations of an Or-tree multi-class classifier with $Y = \{\mathbf{Corridor}, \mathbf{Room}, \mathbf{Doorway}\}$ using the data from Figure 5.7.

## 5.4. Comparison of the Sequential AdaBoost and AdaBoost.M2 Algorithms

The former experiments used the **Sequential AdaBoost** algorithm to create a strong classifier, used to classify the different environments. In this section we compare this approach to **Adaboost.M2**, the multi-class variant of **AdaBoost**.

### 5.4.1. Comparison with Half Maps

The first comparison was calculated using as training and test examples the same sets used in Section 5.2. Figures 5.8, 5.9 and 5.10 show the results of the classification of the test examples for the three different maps used in this work. As we can see the **Sequential AdaBoost** algorithm performs better than the **AdaBoost.M2** algorithm. Table 5.4.1 shows a quantitative analysis of the classification performance for the three maps.

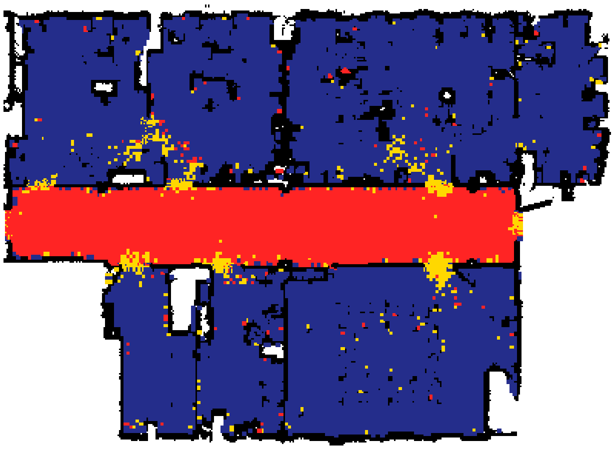
| Map depicted in | Sequential Classifier % | Adaboost.M2 % |
|---|---|---|
| Figure 5.8 | 93.94 | 83.89 |
| Figure 5.9 | 92.10 | 91.83 |
| Figure 5.10 | 89.52 | 82.33 |

Table 5.7.: Results of classification for different indoor maps and different classifiers.

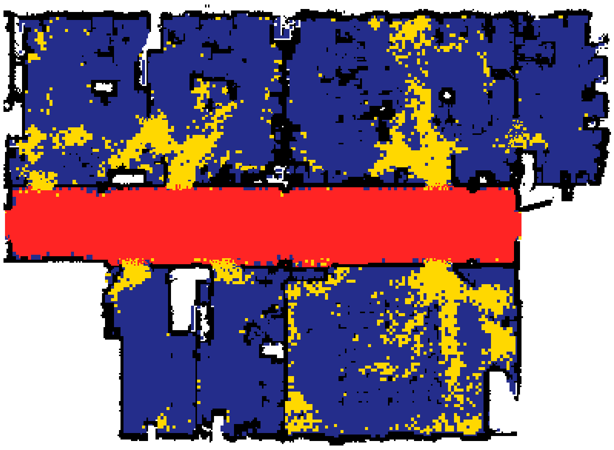
### 5.4.2. Comparison using K-Fold Cross Validation

K-fold cross validation is a standard method for comparing learning algorithms. This method divides the training set into $k$ subsets of equal size. Each of these subsets must contain a representative amount of examples of each class. The process leaves one of the subsets *out* and applies the learning algorithm to the other $k-1$ subsets. The learned classifier is then used to classify the left subset and to calculate its error. The process repeats $k$ times leaving out a different subset each time. The final error is then calculated as the mean of the $k$ partial errors. To compare different learning algorithms the whole process is carried out each time and their final errors are used as a comparison value. A value of $k = 10$ has become a standard for this method.

In our case, we are interested in comparing our **Sequential AdaBoost** algorithm with the **AdaBoost.M2**. For the experiment, we used the map of Building 79 (Figure 5.4(a)). We use as a training set the 3487 poses distributed randomly on the whole map as shown in Figure 5.11. This set is divided into 10 subsets of equal size. Each subset contains a set of randomly selected poses, so that they contain representative examples of each class. Table 5.8 shows the resulting errors of applying 10-fold cross validation with **Sequential AdaBoost**. As we can see in this table, the best three configurations coincide with the ones in Table 5.1, in which a comparison



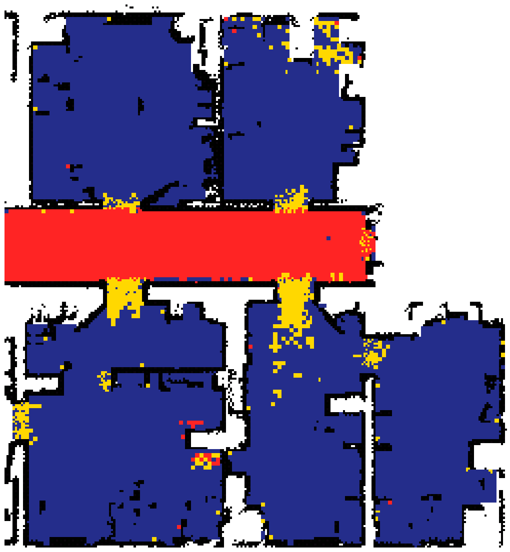
(a) **Sequential AdaBoost**



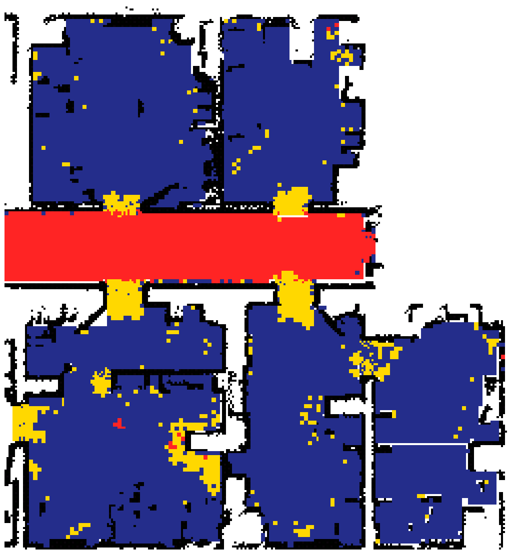
(b) **AdaBoost**.**M2**

Corridor Room Doorway

Figure 5.8.: Figure 5.8(a) shows the classified test data of our **Sequential Adaboost** algorithm in Building 79. Figure 5.8(b) depicts the result obtained with **Adaboost**.**M2**. As can be seen, the error of **Adaboost**.**M2** is much higher compared to our approach.



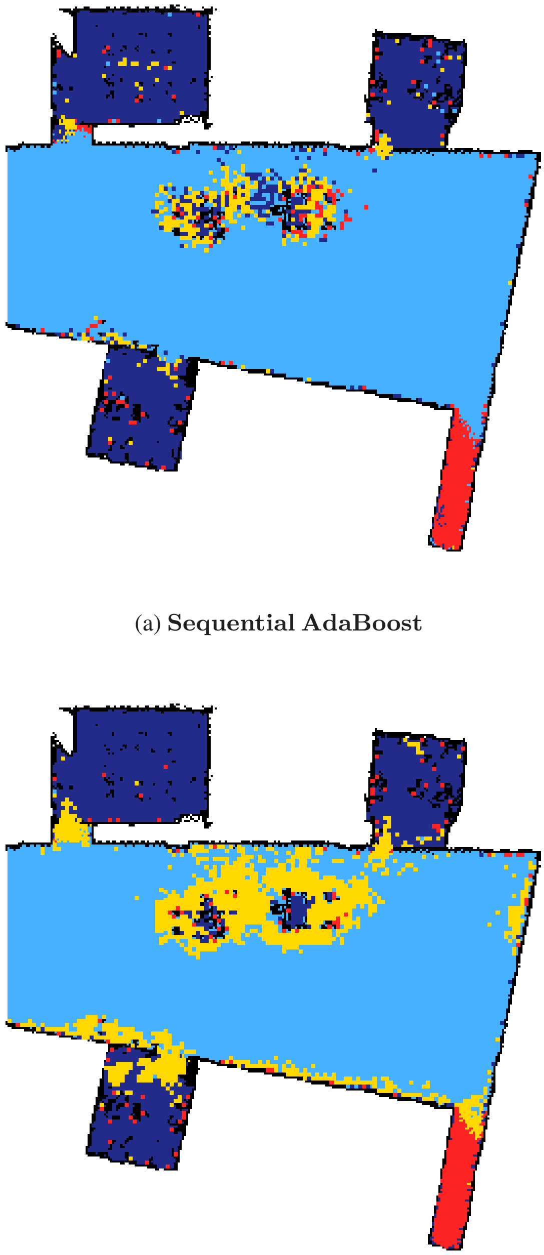
(a) **Sequential AdaBoost**



(b) **AdaBoost.M2**

Corridor Room Doorway

Figure 5.9.: The classified test data of our **Sequential Adaboost** algorithm in Building 52 (Figure 5.9(a)) and the result obtained with **Adaboost.M2** (Figure 5.9(b)). As can be seen, the error of **Adaboost.M2** is higher compared to our approach.



(a) **Sequential AdaBoost**

(b) **AdaBoost.M2**

Corridor Room Doorway Hallway

Figure 5.10.: Classification results of our **Sequential Adaboost** algorithm (Figure 5.10(a)) and **Adaboost.M2** (Figure 5.10(b)). Again the sequential approach outperforms **Adaboost.M2**.

with half maps were done. The error of applying 10-fold cross validation to **AdaBoost.M2** is 9.66%. That means, the three best configurations of **Sequential AdaBoost** are better than **AdaBoost.M2**.

| Classifier Sequence | Error % |
|---|---|
| **Room – Corridor** | 5.84 |
| **Corridor – Room** | 5.96 |
| **Room – Doorway** | 6.39 |
| **Corridor – Doorway** | 9.60 |
| **Doorway – Room** | 10.13 |
| **Doorway – Corridor** | 10.44 |

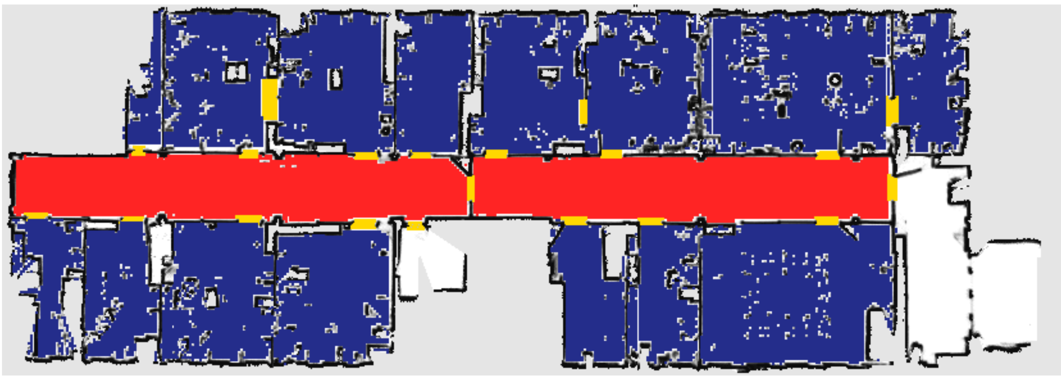
Table 5.8.: Percentage of error of **Sequential AdaBoost** using 10-fold cross validation.

## 5.5. Place Recognition with a Moving Robot

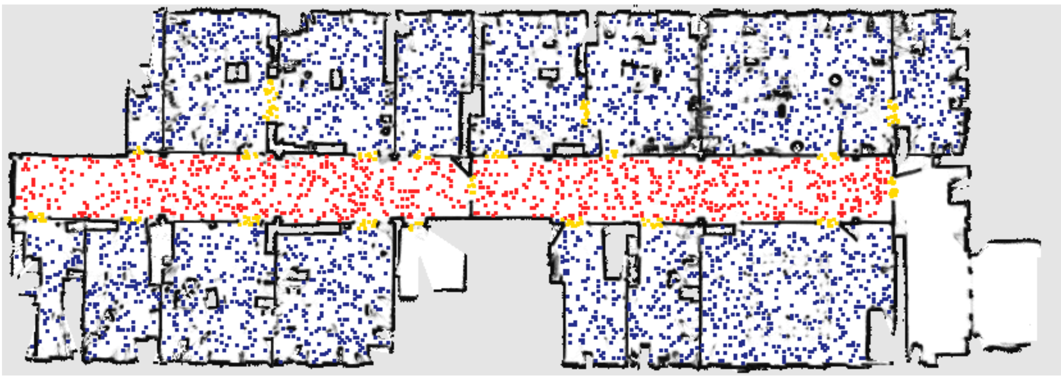
In the following experiment we use the most suitable classifier **Room – Doorway** for our office building (see Table 5.1) to classify the current pose of a mobile robot. We placed our Pioneer2-DX8 robot in our office building and steered it through the corridor, different rooms, and several doorways. Whilst the robot was moving, we logged its trajectory and the classifications obtained for the different range scans. The results are depicted in Figure 5.12. Again, the different colors/grey levels of the points on the trajectory indicate the classification of the corresponding scan. As can be seen, the robot reliably identifies the type of place. Only in a few places is the classification wrong. These failures are most often caused by clutter in the environment, which make the sequential **Room – Doorway** classifier believe that the corresponding place is a doorway.

## 5.6. Transferring the Classifiers to New Environments

This experiment is designed to analyze whether a classifier trained in a particular environment can be used to successfully classify the places of a new environment. To carry out this experiment we trained our **Sequential AdaBoost** classifier on the map shown in Figure 5.5(a). In this environment our approach was able to correctly classify 92.1% of all places with a configuration **Room – Corridor** (see Table 5.2). The resulting classifier was then evaluated on scans simulated given the map of the Intel Research Lab in Seattle. For these scans the classification rate decreased to 82.23% (see Figure 5.13). This indicates that our sequential version algorithm yields good generalizations which can then also be applied to correctly label places of so far unknown environments. Note that a success rate of 82.23% is quite high for this environment, since even humans typically do not consistently/correctly classify the places in this environment.



(a) Random poses



(b) An example subset



Figure 5.11.: The training set used for 10-fold cross validation is shown in Figure 5.11(a). Figure 5.11(b) depicts an example subset used for the calculation of a partial error.


Figure 5.12.: Classification results are obtained with a mobile robot moving through our office environment. Colors/grey levels in the image indicate the classification of the corresponding places of the trajectory.

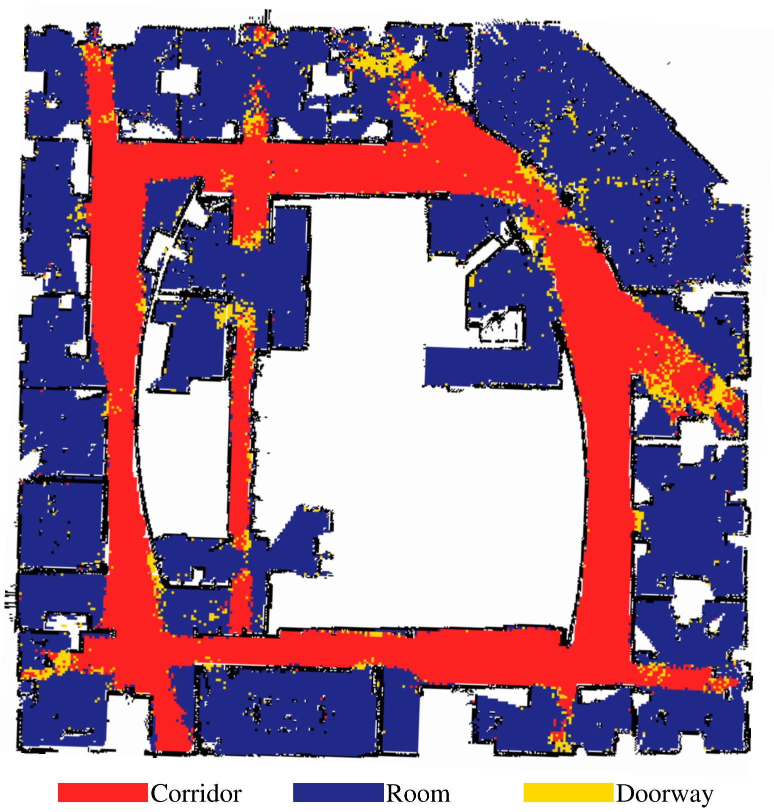



Figure 5.13.: Classification results obtained by applying the classifier learned for the environment depicted in Figure 4.1 to the map of the Intel Research Lab in Seattle. The fact that 82.23% of all places could be correctly classified illustrates that resulting classifiers can be applied to so far unknown environments.

| binary classifier | seven best features |
|---|---|
| **Corridor** | B.6, B.1, P.7, P.6, P.6, B.1, B.1 |
| **Room** | P.2, B.1, P.4, P.6, P.7, B.6, P.5 |
| **Doorway** | B.8, B.1, B.9, B.4, B.2, P.6, B.1, |
| **Hallway** | P.1, B.1, B.8, P.1, P.12, P.6, B.1 |

Table 5.9.: The best seven features for each binary classifier.

## 5.7. Important Simple Features

In this section we analyze the importance of the individual weak features in the final strong classifier. As shown in Section 4.2, we calculated two sets of features: the set $B$, which is formed by features calculated from the raw beams, and the set $P$, with features extracted from the polygon approximation $\mathbf{P}(z)$ of the beams. Table 5.7 lists the seven best features for each binary classifier with the leftmost feature the most important. In this table an entry B.i represents the i-th feature of the $B$ feature set $z$ whereas an entry P.j represents the j-th feature of the P feature set. Note that often identical features occur. As we saw in Section 4.3, on each round $t$ of the $\mathbf{AdaBoost}$ algorithm, each feature $f_j$ is used to create a weak hypothesis $h_j$. In different rounds, the same feature $f_j$ creates different hypotheses $h_j$ which differ in its parameters $\theta$ and $p$. These hypotheses are different, although they correspond to the same feature.

As the table shows, several features like the average difference between consecutive beams (feature B.1) appears to be quite important. Furthermore, the number of gaps (feature B.6), which represents how cluttered the environment is, appears quite often. Whereas feature P.1, which corresponds to the area of the polygon, is most important for the detection of hallways, the feature B.8, which measures the distance between the smallest local minima in the range scan, has the highest weight in the classifier for doorways. In fact, this last feature was designed specifically for the detection of doorways.

In this chapter we presented several experiments which demonstrate that our simple features can be boosted to a robust classifier of places. Additionally, we analyze whether the resulting classifier can be used to classify places in environments for which no training data were available. Finally, we showed the most important geometrical features used in the classifier for each specific place.

# 6. Conclusion

This thesis presented an approach for a mobile robot that learns the relation between the proximity measurements taken in a pose and the semantic term related to this pose. Additionally, our approach creates a classifier that allows the posterior classification of unseen poses into a set of different semantic terms like “corridor”, “room”, “hallway” or “doorway”. Our approach is supervised, which has the advantage that the resulting semantic labels correspond to user-defined classes.

The different set of beams taken as observations are represented by a set of single-valued geometrical features, which are easy and quicker to calculate. These features represent different properties of the beams, as well as of a polygon approximation of the complete scan.

The algorithm used for supervised learning is the boosting **AdaBoost**, which permits us to boost the single-valued features into a strong classifier. In its original form, the **AdaBoost** algorithm generates a binary classifier which can discriminate between two classes: positive examples and negative examples. To determine between more than two classes we devised a structure that uses a set of binary classifiers to form a multi-class classifier. The structure has the form of a decision list, in which each element is represented by one binary classifier. We called this multi-class classifier **Sequential AdaBoost**. To test the performance of the **Sequential AdaBoost** we carried out experiments using different indoor environments, obtaining classification rates of more than 90%. We also compared our **Sequential AdaBoost** algorithm with **AdaBoost.M2**, a multi-class extension of **AdaBoost**. The experimental results showed that our sequential approach outperforms **AdaBoost.M2** in the environments used in this work.

We also analyzed whether a classifier trained in a particular environment can be used to successfully classify the places of a new environment. To carry out this experiment we trained our **Sequential AdaBoost** classifier on one map, and the resulting classifier was then evaluated on scans obtained in a different environment. The experimental results show that our sequential version algorithm yields good generalizations which can then also be applied to correctly label places of so far unknown environments.

Moreover, experiments carried out with a real robot in an office indoor environment demonstrated that the **Sequential AdaBoost** approach is fast enough to do a real time classification of the different poses of the robot. This is due to the use of simple features which are very fast to calculate. Also the number of features used was very low. In the experiments we only used 100 features.

Nevertheless our method also has limitations. Due to the fact that we only use 2-dimensional information from the laser range finder, the method is not well-suited to distinguish the different between rooms which have similar structure.

Our method for semantic classification of places uses observations composed of only 2-dimensional information. This allows us to do a real time classification of places, but with some limitations. The use of only 2-dimensional information permits us to create only features in the plane. However, rooms are 3-dimensional spaces, and as a result, higher dimensional geometric features can improve the information taken from each observation. For example, the use of 3-dimensional scans using a laser range finder would add more information to each observation and allows to extract more complex features, which can be more representative of each room. Although the laser range finder gives very precise proximity measurements, they can not supply other kinds of data which are also representative of an environment, like for example color information. For this reason, the use of a camera can also improve the information that a robot can extract from an observation.

The classification of the different places in a indoor environment is a task that a human can perform without difficulty. One reason for this is that a person is able to recognize the different objects that are situated in a room, and then is able to quickly relate these objects to a place. For example, a bedroom is very easy to recognize if we can locate a bed in it. Hence, the recognition of objects can be a considerable improvement to the task of place classification. And conversely, the classification of places can help the recognition of objects. If a robot has a good estimation of the place where it is, it can use a specific object classifier. For instance, if a robot knows that the area where it is placed is a bedroom, then it can use a specific classifier to look for a bed.

Nevertheless, our method provides a very good estimate of the kind of room as long as the different places are distinguishable based on sensor information obtained in a 2D-plane. Even for more complex places, our approach can be used as an initial guess for the classification technique.

# A. Calculation of the Single-Valued Features

## A.1. Set $B$ of simple features

From now on we will use $z = \{b_0, \ldots, b_{M-1}\}$ to define the set of beams taken as one observation. Each beam $b_i$ is represented by a tuple $(\alpha_i, d_i)$, where $\alpha_i$ is the angle of the beam relative to the robot and $d_i$ is the length of the beam. We also define the function $\mathrm{length}(b_i)$ which returns the length $d_i$ of the beam $b_i$. In the same way we define the function $\mathrm{angle}(b_i)$ which returns the angle $\alpha_i$ of the beam $b_i$.

### A.1.1. The average difference between the length of consecutive beams

The feature representing the average difference between the length of consecutive beams is defined as:

$$f_{\text{average}} = \frac{1}{M} \sum_{i=0}^{M-1} \mathrm{length}(b_i) - \mathrm{length}(b_{i+1 \mod M}) \tag{A.1}$$

### A.1.2. The standard deviation of the difference between the length of consecutive beams

The feature representing the standard deviation of the difference between the length of consecutive beams is defined as:

$$f_{\text{std}} = \frac{1}{M} \sum_{i=0}^{M-1} (\mathrm{length}(b_i) - \mathrm{length}(b_{i+1 \mod M})) - f_{\text{average}}, \tag{A.2}$$

where $f_{\text{average}}$ is the feature defined in A.1.1.

### A.1.3. The average difference between the length of consecutive beams considering max-range

The value max-range is a threshold $\theta$ indicating the maximum length of a beam. Using this threshold $\theta$, we define the function $\mathrm{length}_\theta(b_i)$ as follows:

$$\text{length}_\theta(b_i) = \begin{cases} \text{length}(b_i) & \text{if length}(b_i) \leq \theta \\ \theta & \text{otherwise} \end{cases} \tag{A.3}$$

The feature representing the average difference between the length of consecutive beams using max-range is defined as:

$$f_{\text{average},\theta} = \frac{1}{M} \sum_{i=0}^{M-1} \text{length}_\theta(b_i) - \text{length}_\theta(b_{i+1 \mod M}) \tag{A.4}$$

### A.1.4. The standard deviation of the difference between the length of consecutive beams considering max-range

The standard deviation of the difference between the length of consecutive using max-range beams is defined as:

$$f_{\text{std},\theta} = \frac{1}{M} \sum_{i=0}^{M-1} (\text{length}_\theta(b_i) - \text{length}_\theta(b_{i+1 \mod M})) - f_{average,\theta} \tag{A.5}$$

where $f_{\text{average},\theta}$ is the feature defined in A.1.3.

### A.1.5. The average beam length

The feature representing the average beam length is defined as:

$$f_{\text{average}-\text{length}} = \frac{1}{M} \sum_{i=0}^{M-1} \text{length}(b_i) \tag{A.6}$$

### A.1.6. The standard deviation of the length of the beams

The feature representing the standard deviation of the beam length is defined as:

$$f_{\text{std}-\text{length}} = \frac{1}{M} \sum_{i=0}^{M-1} \text{length}(b_i) - f_{\text{average}-\text{length}} \tag{A.7}$$

where $f_{\text{average}-\text{length}}$ is the feature defined in A.1.5.

### A.1.7. Number of gaps in the scan.

Two consecutive beams build a gap if their length difference is greater than a given threshold $\theta$. An example of a gap is shown in Figure A.1. Formally a gap is defined as:

$$\mathrm{gap}_\theta(b_i, b_{i+1}) = \begin{cases} 1 & \text{if } \mathrm{length}(b_i) - \mathrm{length}(b_{i+1}) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{A.8}$$

The feature representing the total number of gaps is calculated as:

$$f_{\mathrm{gaps},\theta} = \sum_{i=0}^{M-1} \mathrm{gap}(b_i, b_{i+1 \mod M}) \tag{A.9}$$

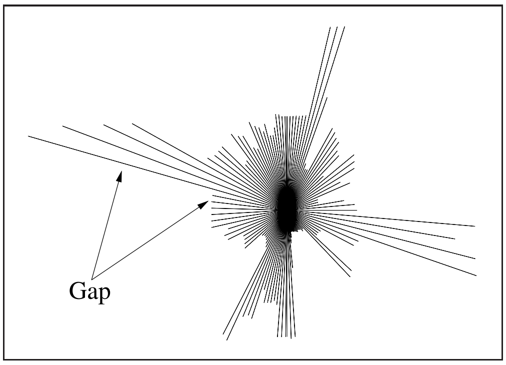



Figure A.1.: An example of a gap in an scan.

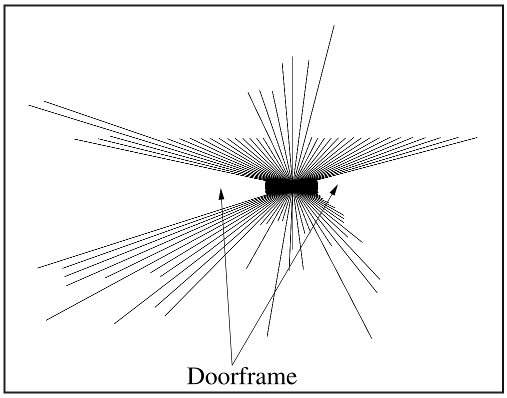
### A.1.8. Number of beams lying on lines that are extracted from the range

This feature is calculated using the method by Sack and Burgard [30].

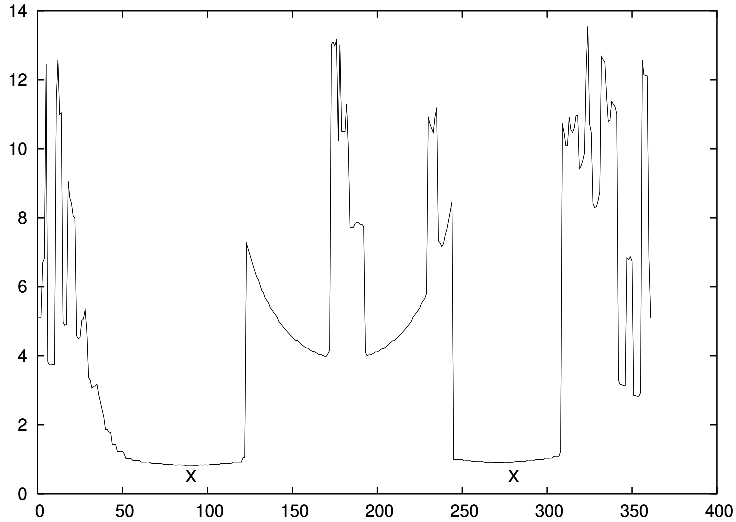
### A.1.9. Euclidean distance between the two points corresponding to two consecutive local minima in the beam length

This feature was designed to help in the classification of doors. If we plot the length of the beams of an observation $z$, we obtain a graph like the one shown in Figure A.2(b). We can look in this graph for two consecutive local minima $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ which can be the representation of a doorframe (Figure A.2(a)). The Euclidean distance between $p_1$ and $p_2$ is then calculate:

$$f_{\mathrm{distance-minima}} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{A.10}$$




(a)




(b)

Figure A.2.: Representation of a set of beams collected where the robot were in a door. In Figure A.2(a) the arrows indicate the two minima which can indicate a doorframe. Figure A.2(b) is a plot of the length of the beams. The two marks **X** indicate the same two minima ($p_1$ and $p_2$ in the text).

### A.1.10. The angular distance between the beams corresponding to two consecutive local minima in the beam length

The two local minimal $p_1, p_2$ calculated in A.1.9 correspond to the end of two beams, i.e $b_1$ and $b_2$. The angular distance between these two beams is used as a feature, in the form:

$$f_{\alpha-\text{minima}} = |\text{angle}(b_1) - \text{angle}(b_2)| \tag{A.11}$$

## A.2. Set $P$ of simple features

This set of features is calculated from a polygonal approximation $\mathbf{P}(z)$ of the area covered by the observation $z = \{b_0, \ldots, b_{M-1}\}$. The vertices $v_i$ of the closed polygon $\mathbf{P}(z)$ correspond to the coordinates of the end-points of each beam $b_i$ of $z$ relative to the robot:

$$\mathbf{P}(z) \quad = \quad \{v_0, \ldots, v_{M-1}, v_M = v_0\}\,, \tag{A.12}$$

where $v_i = (x_i, y_i)$ with $x_i = d_i \cos \alpha_i$ and $y_i = d_i \sin \alpha_i$.

### A.2.1. Area of $\mathbf{P}(z)$

The area of $\mathbf{P}(z)$ is given by:

$$\text{Area}_{\mathbf{P}(z)} = \frac{1}{2} \sum_{i=0}^{M-1} (x_i y_{i+1} - x_{i+1} y_i) \tag{A.13}$$

### A.2.2. Perimeter of $\mathbf{P}(z)$

The area of $\mathbf{P}(z)$ is given by:

$$\text{Perimeter}_{\mathbf{P}(z)} = \sum_{i=0}^{M-1} \text{dist}(v_i, v_i + 1) \tag{A.14}$$

where:

$$\text{dist}(v_i, v_i + 1) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{A.15}$$

### A.2.3. Mean distance between the centroid to the shape boundary

The centroid $c = (c_x, c_y)$ of $\mathbf{P}(z)$ is defined as:

$$c_x = \frac{1}{6 \cdot \text{Area}_{\mathbf{P}(z)}} \sum_{i=0}^{M-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{A.16}$$

$$c_y = \frac{1}{6 \cdot \text{Area}_{\mathbf{P}(z)}} \sum_{i=0}^{M-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{A.17}$$

The mean distance between the centroid to the shape boundary of $\mathbf{P}(z)$ is calculated as:

$$f_{\text{mean-shape}} = \frac{1}{M} \sum_{i=0}^{M-1} \text{dist}(v_i, c), \tag{A.18}$$

where:

$$\text{dist}(v_i, c) = \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} \tag{A.19}$$

### A.2.4. Standard deviation of the distances between the centroid to the shape boundary

The standard deviation of the distances between the centroid to the shape boundary of $\mathbf{P}(z)$ is given by:

$$f_{\text{std-shape}} = \frac{1}{M} \sum_{i=0}^{M-1} \text{dist}(v_i, c) - f_{\text{mean-shape}} \tag{A.20}$$

where $f_{\text{mean-shape}}$ is the feature defined in A.2.3.

### A.2.5. Similarity invariant descriptors based on the Fourier transformation

To calculate the Fourier coefficients we transform each vertex $v_i \in \mathbb{R}^2$ of $\mathbf{P}(z)$ into a complex number $\tilde{v}_i \in \mathbb{C}$ in the form:

$$v_i = (x_i, y_i), \Longrightarrow \tilde{v}_i = x_i + y_i j, \;\; j = \sqrt{-1} \tag{A.21}$$

The Fourier coefficients $\{c_{-n}, \ldots, c_{-1}, c_0, c_1, \ldots, c_n\}$ of $\mathbf{P}(z)$ are then calculated as:

$$c_0 = \frac{1}{2T} \sum_{k=0}^{M-1} (\tilde{v}_k + \tilde{v}_{k+1}) \mid \triangle\tilde{v}_k \mid \tag{A.22}$$

$$c_n = \frac{T}{(2\pi n)^2} \sum_{k=0}^{M-1} (\triangle s_{k+1} - \triangle s_k) e^{-j\pi(\frac{2\pi}{T})t_k} \tag{A.23}$$

with:

$$\begin{aligned}
\triangle\tilde{v}_i &= \tilde{v}_{i+1} - \tilde{v}_i && \text{(A.24)}\\
\triangle s_i &= \triangle\tilde{v}_i / |\triangle\tilde{v}_i| && \text{(A.25)}\\
t_k &= \sum_{i=0}^{k-1} |\triangle\tilde{v}_i| \quad k > 0, t_0 = 0 && \text{(A.26)}\\
T &= \text{perimeter of } \mathbf{P}(z) && \text{(A.27)}
\end{aligned}$$

The Fourier descriptors $\{\tilde{x}_{-n}, \ldots, \tilde{x}_0, \ldots, \tilde{x}_n\}$, which are invariant to similarity, that is, translation, rotation and scale, are calculated as:

$$\left\{\tilde{x}_n := \tfrac{|c_n|}{|c_1|} e^{j(\Phi_n + (1-n)\Phi_2 - (2-n)\Phi_1)}\right\} \quad \text{(A.28)}$$

$$\Phi_n = \text{phase of } c_n \quad \text{(A.29)}$$

### A.2.6. Major axis $\mathrm{Ma}$ of the ellipse that approximates $\mathrm{P}(z)$ using the first two Fourier coefficients

Having the two Fourier coefficients $(c_{-1}, c_1)$ of $\mathbf{P}(z)$ (see A.2.5), we can calculate the major axis of an ellipse that approximates the polygon $\mathbf{P}(z)$ as:

$$\mathrm{Ma} = |c_1| + |c_{-1}| \quad \text{(A.30)}$$

### A.2.7. Minor axis $\mathrm{Mi}$ of the ellipse that approximates $\mathrm{P}(z)$ using the first two Fourier coefficients

Having the two Fourier coefficients $(c_{-1}, c_1)$ of $\mathbf{P}(z)$ (see A.2.5), we can calculate the minor axis of an ellipse that approximates the polygon $\mathbf{P}(z)$ as:

$$\mathrm{Mi} = ||c_1| - |c_{-1}|| \quad \text{(A.31)}$$

### A.2.8. Invariants calculated from the normalized central moments of $\mathrm{P}(z)$

The central moments $\mu_{pq}$ of $\mathbf{P}(z)$ up to 3 are:

$$\mu_{10} = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} (x_i - \bar{x})^1 (y_j - \bar{y})^0 \quad \text{(A.32)}$$

$$\mu_{01} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^0 (y_j - \bar{y})^1 \quad \text{(A.33)}$$

$$\mu_{11} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^1 (y_j - \bar{y})^1 \tag{A.34}$$

$$\mu_{20} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^2 (y_j - \bar{y})^0 \tag{A.35}$$

$$\mu_{02} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^0 (y_j - \bar{y})^2 \tag{A.36}$$

$$\mu_{30} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^3 (y_j - \bar{y})^0 \tag{A.37}$$

$$\mu_{03} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^0 (y_j - \bar{y})^3 \tag{A.38}$$

$$\mu_{12} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^1 (y_j - \bar{y})^2 \tag{A.39}$$

$$\mu_{21} = \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} (x_i - \bar{x})^2 (y_j - \bar{y})^1 \tag{A.40}$$

with:

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M-1} x_i \tag{A.41}$$

and:

$$\bar{y} = \frac{1}{M} \sum_{j=1}^{M-1} y_j \tag{A.42}$$

The normalized central moments, denoted $\eta_{pq}$ of $\mathbf{P}(z)$ are defined as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{A.43}$$

where:

$$\gamma = \frac{p+q}{2} + 1 \tag{A.44}$$

for $p + q = 2, 3, \ldots$

A set of seven invariant moments with respect translation, rotation and scale can be derived from the second and third moments:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{A.45}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{A.46}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{A.47}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{A.48}$$

$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\ & +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} - \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned} \tag{A.49}$$

$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\ & +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned} \tag{A.50}$$

$$\begin{aligned}\phi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} - \eta_{12})\left[(\eta_{30} - \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\ & +(3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned} \tag{A.51}$$

### A.2.9. Normalized feature of compactness of $\mathrm{P}(z)$

The normalized feature of compactness $\mathrm{M}_{\mathrm{cmp}}$ of $\mathbf{P}(z)$ is calculated as:

$$\mathrm{M}_{\mathrm{cmp}} = \frac{\mathrm{Area}_{\mathbf{P(z)}}}{\mu_{20} + \mu_{02}}, \quad 0 \leq \mathrm{M}_{\mathrm{cmp}} \leq 1 \tag{A.52}$$

where $\mu_{20}, \mu_{02}$ are the central moments of second order calculated in A.2.8.

### A.2.10. Normalized feature of eccentricity of $\mathrm{P}(z)$

The normalized feature of eccentricity $\mathrm{M}_{\mathrm{ect}}$ of $\mathbf{P}(z)$ is calculated as:

$$\mathrm{M}_{\mathrm{ect}} = \frac{\sqrt{(\mu_{20} + \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02}}, \quad 0 \leq \mathrm{M}_{\mathrm{ect}} \leq 1 \tag{A.53}$$

where $\mu_{20}, \mu_{02}, \mu_{11}$ are the central moments of second order calculated in A.2.8.

### A.2.11. Form factor of $\mathrm{P}(z)$

The form factor of the $\mathbf{P}(z)$ is given by:

$$f_{\mathrm{f-factor}} = \frac{4\pi \mathrm{Area}_{\mathbf{P}(\mathrm{z})}}{\sqrt{\mathrm{Perimeter}_{\mathbf{P}(\mathrm{z})}}} \tag{A.54}$$

# B. Developed Software

To generate the training and test sets of examples used in the experiments of this work, a specific program was developed that communicates with the CARMEN server to obtain the map in which the simulation is running. Once the software is connected to CARMEN, its graphic user interface (GUI) permits the user to select points in the map where the robot must be situated to obtain the observations taken from the laser (Figure B.1). There exit two modes to obtain laser beams information. In the first mode, the simulated robot is situated in a pose in the map and its beams are recorded as can be seen in Figure B.2. The second mode permits a massive selection of points with their corresponding label. This points are used later on to pose the robot and record the laser information. Figure B.3 shows how can we select a set of points with a specific label.

The software also permits the visualization of a set of examples that were classified by any classifier used in this work. The points are drawn using different colors for each label (Figure B.4). The images used in this work were obtained using this software.

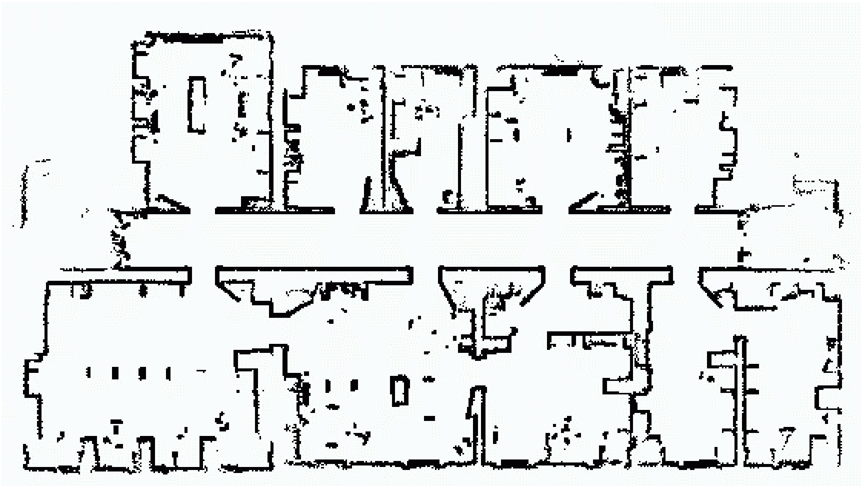


Figure B.1.: Example of a map obtained from CARMEN.

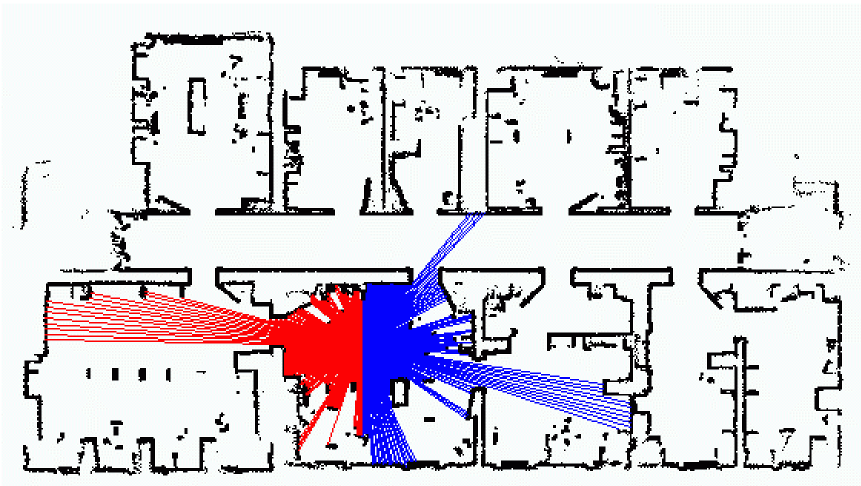


Figure B.2.: Positioning of a robot in a map together with its beams. Blue rays represent the front laser and red ones represent the rear laser.

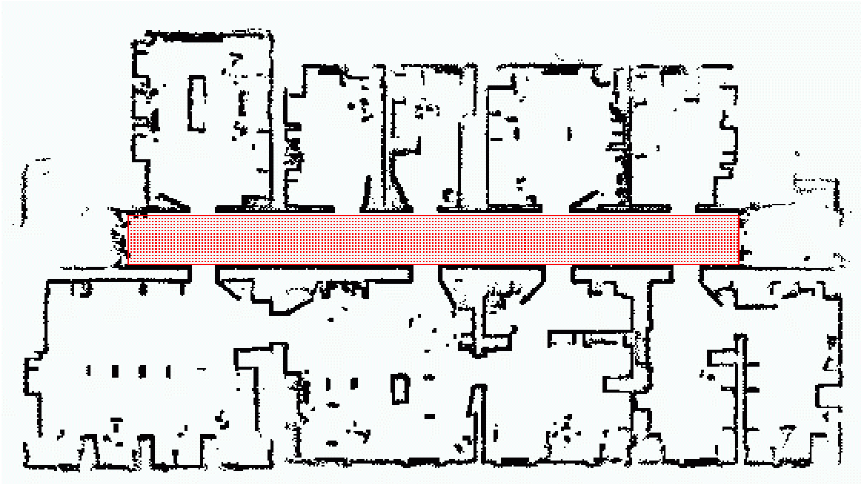


Figure B.3.: Selection of a set of poses that correspond to the corridor.

Figure B.4.: Visualization of the classification of a set of poses corresponding to the rigth part of the map. Each class is drawn with a different color.

# List of Algorithms

# Bibliography


[1] P. Althaus and H.I. Christensen. Behaviour coordination in structured environments. *Advanced Robotics*, 17(7):657–674, 2003.

[2] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[3] D. Anguelov, D. Koller, Parker E., and S. Thrun. Detecting and modeling doors with mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[4] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[5] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *ArtificialIntelligence*, 114(1-2), 2000.

[6] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 637–642, 2002.

[7] Freund. Boosting a weak learning algorithm by majority. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1990.

[8] Y. Freund and R. Schapire. A short introduction to boosting. *Journal of Japanase Society for Artificial Intelligence*, 14(5):771–780, 1999.

[9] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory (Eurocolt)*, 1995.

[10] Yoav Freund. Data filtering and distribution modeling algorithms for machine learning. Technical Report UCSC-CRL-93-37, 1993.

[11] R.C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Inc., 1987.

[12] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[13] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Publishing Inc., 1992.

[14] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

[15] Japan Robot Association. http://www.jara.jp/.

[16] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[17] Michael J. Kearns and L.G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory, August 1988.

[18] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[19] S. Koenig and R. Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122. MIT-Press, 1998.

[20] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, 2002.

[21] Whitcomb L, Yoerger D, Singh H, and Howland J. Advances in underwater robot vehicles for deep ocean exploration: Navigation, control and survey operations. In *9th Internation Symposium onRobotics Research (ISRR)*, pages 346–353, 1999.

[22] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8), 1998.

[23] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In *Advanced lectures on machine learning*, pages 118–183. Springer-Verlag New York, Inc., 2003.

[24] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[25] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[26] S. Oore, G.E. Hinton, and G. Dudek. A mobile robot that learns its place. *Neural Computation*, 9(3):683–699, 1997.

[27] J. O'Rourke. *Computational Geometry in C (2nd Edition)*. Cambridge University Press, 1998.

[28] Real Academia Española (Spanish Royal Academy). http://www.rae.es/.

[29] J.C. Russ. *The Image Processing Handbook*. CRC Press, 1992.

[30] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. In *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[31] R. Schapire. The boosting approach to machine learning: An overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[32] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[33] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Co., 1996.

[34] C. Stachniss and W. Burgard. Mapping and exploration with mobile robots using coverage maps. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[35] Raj Talluri and J. K. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Trans. on Robotics and Automation*, 8(5):573–584, 1992.

[36] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery Montemerlo, C. Deepayan, and W. Burgard. A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–442, 2003.

[37] K. Tieu and P. Viola. Boosting image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 228–235, 2000.

[38] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2003.

[39] A. Treptow, A. Masselli, and A. Zell. Real-time object tracking for soccer-robots without color information. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2003.

[40] K. Čapek. *R.U.R. (Rossum's Universal Robots)*. Dover Publications, 2001.

[41] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[42] P. Viola and M.J. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.

[43] SB Williams, P Newman, J Rosenblatt, G Dissanayake, and H Durrant-Whyte. Autonomous underwater navigation and control. *Robotica*, 19(5):481–496, 2001.

[44] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.