Probabilistic Modeling of Dynamic Environments for Mobile Robots

Daniel Meyer-Delius Di Vasto

Technische Fakultät
Albert-Ludwigs-Universität Freiburg im Breisgau

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

August 2011

Probabilistic Modeling of Dynamic Environments for Mobile Robots

Daniel Meyer-Delius Di Vasto

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften
Technische Fakultät Albert-Ludwigs-Universität Freiburg im Breisgau

Tag der Disputation: 16.08.2011

Dekan der Technische Fakultät:	Prof. Dr. Bernd Becker
Erstgutachter:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	Prof. Dr. Bernhard Nebel

Abstract

To be truly useful, mobile robots not only need to operate autonomously, they need to operate autonomously over extended periods of time. In realistic environments, this means that they need to be able to cope with potential changes in their surroundings. In this thesis we present different techniques that allow mobile robots to explicitly represent and reason about changes in the environment. The goal is to enable robots to reliably operate over extended periods of time in dynamic environments.

We approach the problem of representing and reasoning about dynamic environments using probabilistic techniques and present solutions to a number of problems associated to the operation of mobile robots in such environments. These solutions include an approach to estimate the pose of the robot in semi-static environments, a model of the environment that represents the occupancy of the space and additionally characterizes how this occupancy changes over time, and a landmark placement approach that can be used to improve robot navigation in dynamic environments. Furthermore, we address the problem of reasoning in dynamic environments. We define a situation as a relevant spatio-temporal configuration of the environment and present two approaches for the modeling and recognition of situations. All solutions presented in this thesis were implemented and tested in simulation as well as with different mobile robotic platforms. The results show that explicitly considering the dynamics of the environment can considerably improve the performance of a robotic system.

With this thesis, we contribute to the field of robotics by providing several novel solutions to a number of relevant problems associated to the operation of mobile robots in dynamic environments. We develop novel probabilistic models that explicitly represent the dynamics of the environment and present efficient methods for probabilistic inference that enable a robotic system to reason about these dynamics. The issues and challenges addressed in this thesis are fundamental for the ultimate goal of long-term autonomous operation of mobile robots.

Zusammenfassung

Damit mobile Roboter in Zukunft sinnvoll eingesetzt werden können, müssen sie in der Lage sein, über längere Zeiträume autonom agieren zu können. Da die meisten interessanten Anwendungsbereiche langfristig gesehen dynamisch sind, müssen mobile Roboter außerdem mit Veränderungen in der Umgebung umgehen können. Manche dieser Veränderungen sind nur von kurzer Dauer — wie etwa eine vorbeilaufende Person —, während sich andere über längere Zeiträume erstrecken — wie zum Beispiel, wenn in einem Büro eine neue Wand errichtet wird. Obwohl es in bestimmten Situationen unproblematisch ist, die Veränderungen zu ignorieren, stellt eine solche Herangehensweise oft keine geeignete Lösung dar. Die vorliegende Arbeit beschreibt verschiedene Methoden, die einem mobilen Roboter die Repräsentation von und das Urteilen über Umgebungsveränderungen ermöglichen, mit dem Ziel, über längere Zeiträume in dynamischen Umgebungen zuverlässig agieren zu können.

Um sowohl sicher als auch effizient zu funktionieren, sind mobile Roboter auf ein Modell angewiesen, welches die wesentlichen Merkmale ihrer Umgebung und deren Veränderung über die Zeit beschreibt. Eine vollständige Modellierung der Umgebung ist offensichtlich nur in sehr einfachen bzw. trivialen Szenarien möglich. Neben der Vielzahl unterschiedlichster Eigenschaften, die für realistische Anwendungen beschrieben werden müssen, ist die Charakterisierung des zeitlichen Verhaltens dieser Aspekte an sich ein komplexes Problem: Einerseits sind viele Veränderungen unberechenbar und andererseits gibt es, selbst bei vorhersehbaren Veränderungen, meistens kein allgemeines Modell, um sie angemessen darzustellen. Aus diesen Gründen setzen viele Ansätze im Bereich der mobilen Robotik eine statische Welt voraus und arbeiten mit Modellen, die ausschliesslich statische Umgebungsmerkmale repräsentieren.

Diese Arbeit behandelt zwei fundamentale Fragestellungen, die sich aus diesem Kontext ergeben. Erstens stellt sich die Frage, wie dynamische Umgebungen modelliert werden können. Zweitens ergibt sich die Frage nach der Inferenz aus diesen Modellen. Hierzu werden in der Arbeit wahrscheinlichkeitstheoretische Verfahren und

Modelle entwickelt, um Lösungen für mehrere relevante Probleme bereitzustellen, die mit dem Agieren mobiler Roboter in diesen Umgebungen zusammenhängen.

Nach einem einführenden Kapitel 2, in dem grundlegende Algorithmen und Modellen erklärt werden, wird in Kapitel 3 ein Lokalisierungsansatz beschrieben, der ausgehend von Beobachtungen nicht-statischer Objekte lokale Karten erstellt, welche die Referenzkarte der Umgebung temporär erweitern. Diese vorübergehenden Karten erlauben dem Roboter, seine Position selbst in Bereichen mit lang anhaltenden Veränderungen robust zu schätzen. Kapitel 4 präsentiert ein wahrscheinlichkeitstheoretisches Modell für die Belegung des Raumes, welches zusätzlich die Veränderung dieser Belegung über die Zeit charakterisiert. Die explizite Darstellung dieses Wechsels der Belegung ermöglicht ein besseres Verständnis der Umgebung und somit eine verbesserte Roboternavigation. Letzteres kann auch durch die Platzierung leicht erkennbarer Landmarken in statischen Bereichen der Umgebung erreicht werden. Insbesondere verbessert dieses Vorgehen die Navigation auch in symmetrischen Umgebungen oder in solchen, die nicht über eine ausreichende Anzahl an wiedererkennbaren Merkmalen verfügen, so dass die Position des Roboters nicht eindeutig bestimmt werden kann. Dieses Problem der Landmarken-Platzierung zur Minderung der strukturellen Mehrdeutigkeit der Umgebung wird in Kapitel 5 behandelt. Hier wird ein Verfahren vorgestellt für das Berechnen einer Konstellation ununterscheidbarer Landmarken, die die strukturelle Mehrdeutigkeit der Umgebung reduziert und somit die Zuverlässigkeit der Lokalisierung erhöht. Die letzten zwei Kapitel beschäftigen sich schlielich mit dem Problem des Schlussfolgerns in dynamischen Umgebungen, konkret in Verkehrsszenarien. Eine ‚Situation' wird als eine Abfolge relevanter Konfigurationen der Umgebung definiert und es wird für jede Klasse von Situationen ein Modell gelernt, welches die Situation charakterisiert. Diese erlernten Modelle werden dann für die Erkennung von sich entwickelnden Instanzen der unterschiedlichen Situationsklassen benutzt, und ermöglichen dem Roboter auf diese Weise anhand der vorkommenden Situationen rational zu agieren. Zur Modellierung und Erkennung von Situationen werden in diesem letzten Teil der Arbeit zwei unterschiedliche Ansätze beschrieben: In Kapitel 6 werden Hidden Markov Modelle verwendet, während in Kapitel 7 ein Regressionsansatz präsentiert wird.

All die Ansätze, die in dieser Arbeit präsentiert werden, wurden sowohl in Simulation wie auch auf unterschiedlichen mobilen Robotern implementiert und evaluiert. Dabei zeigen die in diesen Experimenten erzielten Ergebnisse, dass die explizite Berücksichtigung der Dynamik der Umgebung die Leistungsfähigkeit des Roboters erheblich verbessern kann.

So leistet die vorliegende Arbeit einen Beitrag auf dem Gebiet der mobilen Robotik indem sie innovative Lösungsansätze zu relevanten Problemstellungen liefert, die die Performanz von mobilen Robotern in dynamischen Umgebungen verbessern. Hierzu werden mehrere wahrscheinlichkeitstheoretische Modelle für die Darstellung der Dynamik der Umgebung entwickelt sowie neue wahrscheinlichkeitstheoretische Verfahren eingeführt, die es einem Robotersystem erlauben, aus diesen Umgebungsmodellen effizient zu inferieren. Die Probleme und Herausforderungen, mit denen sich diese Arbeit befasst, sind wesentliche Bestandteile des langfristigen Ziels, mobile Roboter über längere Zeiträume autonom agieren zu lassen. Konkret setzt sich der Beitrag dieser Arbeit zusammen aus:

- einem Lokalisierungsverfahren für „semi-statische“ Umgebungen,
- einem wahrscheinlichkeitstheoretischen Modell für die Beschreibung dynamischer Umgebungen,
- einer Landmarken-Platzierungsmethode für eine verbesserte Lokalisierung des Roboters und
- einem Verfahren für Situationserkennung in Verkehrsszenarien.

Acknowledgments

I would like to thank all the people who helped and supported me during my research; this thesis would not have been possible without their contribution.

First of all, I would like to thank Professor Wolfram Burgard for his motivation, guidance and support. Working under his supervision at the University of Freiburg has been a great experience; I have had the chance to participate in several interesting projects and was always given freedom to follow my own research directions. I also thank Professor Burgard for his fruitful theoretical discussions and practical advices.

I would like to thank my friends and colleagues in Freiburg for their help and companionship. In particular, the collaborations of Maximilian Beinhofer, Giorgio Grisetti, Jürgen Hess, Alexander Kleiner, Christian Plagemann and Jürgen Sturm, have been a great contribution to my research and to this thesis. Christian Plagemann's help during my first research years was especially important. I thank Dominik Joho, Óscar Martínez Mozos and Jürgen Sturm for their friendship and help inside and outside the lab. My thanks to Axel Rottmann for being a great friend and for helping me survive all these years among the Alemannen.

I would also like to express my gratitude to Susanne Bourjaillat and Michael Keser for providing administrative and technical help.

I thank Wendelin Feiten, Thilo Grundmann, Gisbert Lawitzky, Georg von Wichert and the other researchers at Siemens Corporate Technology in Munich for the many insightful and productive discussions about situation recognition.

I am grateful to Sonya Gzyl, Jürgen Sturm, Dominik Joho, Axel Rottmann, Barbara Frank and Maximilian Beinhofer for their corrections and comments on earlier versions of this document.

Finally and most importantly, I would like to thank my family for their love and support. Most of all, I thank Sonya for her care, understanding, patience and for believing in me, and Natalia for the joy and happiness she has brought to my life

Not to be absolutely certain is, I think, one of the essential things in rationality.

Bertrand Russell

Contents

Chapter 1

Introduction

As the research field of robotics continues to develop and related technologies improve, mobile robots find new application domains; performing increasingly challenging tasks in increasingly challenging environments. Robots have been successfully employed in planetary and underwater exploration, search and rescue missions, and are currently jamming our streets and parking lots in the form of autonomous cars.

Together with the task's complexity, operational times are also continually increasing. This emphasizes one of the most difficult challenges faced by mobile robots: dealing with changes in their surroundings. Some of these changes only last for a brief period of time, such as moving people or passing cars. Others last for longer periods, like rearranged furniture and parked cars. And there are some changes that last for extended periods of time, for example, when a new wall is built in an office, or a new road is constructed between two towns. Depending on the task at hand, some of these changes can be irrelevant for the robot and be safely disregarded. This, however, is not always an appropriate solution, since some changes can influence the robot's performance and therefore need to be taken into account. In this thesis we present different techniques that allow a mobile robot to explicitly represent and reason about changes in the environment. The goal is to enable the robot to reliably operate over extended periods of time in dynamic environments.

The problem with dynamic environments is that they are much more complex to describe than static ones. For safe and efficient operation, mobile robots rely on information about both the pertinent aspects of the environment and how these aspects change over time. This information is referred to as the model of the environment. Clearly, specifying a model of the complete environment is practically impossible but for trivial cases. Consider for example the task of modeling the aspects of the world

relevant to an autonomous car. This would require describing far too many aspects like roads, other vehicles, pedestrians, etc. In addition to the number of aspects that need to be described, characterizing the way in which these aspects change over time is a problem on its own. First, changes may be practically or even inherently unpredictable. For instance, the behavior of other vehicles subject to the inherently unpredictable decisions of their drivers. And secondly, even when changes can be predicted to some extend, there is usually no general model to appropriately describe them.

For the reasons illustrated above, most mobile robot systems today make the simplifying assumption of a static environment. A model is provided to the system beforehand and is then used as reference disregarding subsequent changes in the environment. Robust systems are able to handle some inconsistencies between the model and the actual environment. However, a largely inconsistent model can degrade the performance of the system or even lead to a complete failure. Furthermore, the overall performance of the robot is limited by the lack of information about how the environment behaves.

In this thesis we approach the problem of representing and reasoning about dynamic environments using probabilistic techniques and models. Probabilistic techniques have been successfully applied for solving numerous complex problems with many sources of uncertainty. The idea is to explicitly represent the uncertainty present in the system. This uncertainty being the result of the inherent unpredictability of the environment, noisy and limited perception, unreliable robot actuation, algorithmic approximations, etc. The incompleteness and limitations of the models themselves, which are only a partial and approximate description of the environment, are also an important source of uncertainty.

The contributions of this thesis are solutions to several relevant problems associated to the operation of mobile robots in dynamic environments. After introducing in Chapter 2 the fundamental algorithms and models used throughout this thesis, we present in Chapter 3 an approach to estimate the pose of the robot in a dynamic environment. The approach relies on the measurements caused by non-static objects to build local maps that temporarily extend the reference map of the environment. These temporary maps allow the robot to reliably estimate its pose also in regions that are subject to persistent changes. In Chapter 4, we describe a probabilistic model of the environment that represents the occupancy of the space and additionally characterizes how it changes over time. The explicit representation of how the occupancy changes provides a better understanding of the environment that can be used to improve the navigation performance of the robot. Navigation in dynamic environments can also

be improved by attaching easily detectable landmarks to static parts of the environment. This strategy can also improve navigation in environments that are structurally symmetrical or have only few salient features so that the pose of the robot cannot be uniquely determined. In Chapter 5 we investigate how artificial landmarks can be placed to reduce the inherent ambiguity in the environment. We present a practical approach to compute a configuration of indistinguishable landmarks that decreases the overall ambiguity and thus increases the robustness of the pose estimation. In the last two technical chapters of this thesis, we address the problem of reasoning about situations in dynamic environments. We focus on an autonomous car application and consider traffic situations that typically occur in highway-like driving settings. We define a situation as a relevant sequence of configurations of the environment and learn, for each situation type, a model describing the characteristic dynamics of the situation. These models are then used to recognize instances of the corresponding situations as they are developing, allowing the robot to act rationally based on the occurring situations. In Chapter 6 situations are described using hidden Markov models while in Chapter 7 we present a regression-based approach.

All the approaches presented in this thesis were implemented and tested using simulation as well as with different mobile robotic platforms. The results show that explicitly considering the dynamics of the environment can considerably improve the performance and robustness of a robotic system. The issues and challenges addressed in this thesis are fundamental to enable mobile robots to operate over extended periods of time autonomously.

1.1 Contributions of this Thesis

With this thesis, we contribute several novel approaches to the field of robotics by investigating and developing solutions to a number of problems associated to the operation of robotic systems in dynamic environments. We develop suitable probabilistic models to explicitly represent the dynamics of the environment and present probabilistic techniques that enable a robotic system to reason about these dynamics. The goal is to allow robotic systems to reliably operate over extended periods of time in dynamic environments. Concretely, the main contributions of this thesis are:

- A robot localization framework for semi-static environments (Chapter 3)
- A probabilistic model for dynamic environments (Chapter 4)

- A landmark placement approach for improved robot localization (Chapter 5)
- A situation modeling framework for traffic scenarios (Chapter 6 and 7)

1.2 Publications

The work presented in this thesis is partially based on the following publications:

- D. Meyer-Delius, M. Beinhofer, A. Kleiner and W. Burgard, *Using Artificial Landmarks to Reduce the Ambiguity in the Environment of a Mobile Robot*. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 2011.
- D. Meyer-Delius, J. Hess, G. Grisetti and W. Burgard, *Temporary Maps for Robust Localization in Semi-static Environments*. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 2010.
- D. Meyer-Delius, J. Sturm and W. Burgard, *Regression-Based Online Situation Recognition for Vehicular Traffic Scenarios*. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, USA, 2009.
- D. Meyer-Delius, C. Plagemann and W. Burgard, *Probabilistic Situation Recognition for Vehicular Traffic Scenarios*. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 2009.
- D. Meyer-Delius, C. Plagemann, G. von Wichert, W. Feiten, G. Lawitzky, W. Burgard, *A Probabilistic Relational Model for Characterizing Situations in Dynamic Multi-Agent Systems*. In Proc. of the Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications (GFKL), Freiburg, Germany, 2007.

Chapter 2

Basics

In this chapter we review the fundamental algorithms and models used throughout this thesis. First, we introduce the mobile robot localization problem and present the Monte-Carlo localization algorithm — one of the most popular approaches to robot localization. We then discuss gird maps, also a popular technique for representing the environment in the field of mobile robotics. Finally, we present the hidden Markov model, a probabilistic graphical model for state estimation in dynamic systems that allows for simple and elegant implementations of the basic inference tasks.

2.1 Mobile Robot Localization

Mobile robot localization consists in estimating the pose of the robot relative to a given map of the environment using sensor data. Accurate and robust localization is essential for the successful navigation of the robot in the environment.

The localization problem can be divided into two types depending on the information available to the robot about its initial pose in the reference map (see [Thrun *et al.*, 2005] for an in-depth discussion about the different classes of localization problems). In case that a good estimate is available, the localization problem is referred to as *position tracking*. Since the initial pose is assumed to be known, position tracking consists in adjusting the pose of the robot as new sensor data is obtained. If, on the other hand, no information about the initial pose is available, the localization problem is referred to as *global localization*. This problem is much harder because a larger number of potential initial poses has to be considered. Position tracking is a special case of the global localization problem in which the initial pose of the robot is given. There is also a variant of the global localization problem known as the *kidnapped robot* prob-

lem. While in global localization the pose of the robot is known to be unknown, in the kidnapped robot problem, the pose of the robot is incorrectly believed to be known. This is a much harder problem since in certain situations, for example in dynamic environments or environments that are structurally symmetrical or have only few recognizable features, it is not possible for the robot to determine that its estimated pose is incorrect.

Besides the available information about the initial pose of the robot, localization problems can again be divided into two types depending on the behavior of the environment over time. Localization in *static* environments assumes that the environment does not change over time while localization in *dynamic* environments addresses the more general case where other objects besides the robot can change their location or configuration. Although environments are, in general, dynamic, the majority of existing localization approaches assume a static environment. Dealing with dynamic environments is, in most of the cases, more difficult than dealing with static ones.

In the next section we describe the Monte-Carlo localization algorithm [Dellaert *et al.*, 1999] for estimating the pose of the robot in static environments. Monte-Carlo localization is one of the most popular localization approaches in robotics. It is a probabilistic algorithm robust to some unmodeled dynamic in the environment and capable of addressing the global localization problem.

2.1.1 The Monte-Carlo Localization Algorithm

As already mentioned, the mobile robot localization problem consists in estimating the pose of a robot relative to a given map of the environment using sensor data. In this thesis, we consider only robots restricted to planar environments, thus the pose of the robot $x_t = (r_t^x, r_t^y, r_t^\theta)^T$ at time t is given by its two location coordinates r_t^x and r_t^y in the plane and its orientation r_t^θ. The sensor data consists in exteroceptive data $z_{1:t} = z_1, z_2, \ldots, z_t$ referred to as measurements or observations, and proprioceptive data $u_{1:t} = u_1, u_2, \ldots, u_t$, which in this thesis corresponds to odometry measurements. An odometry measurement u_t represents the relative motion of the robot between the poses x_{t-1} and x_t. An observation z_t represents some information about the state of the environment at time t. Throughout this thesis we consider only laser range scanners. This type of sensors generate more than one measurement at any given time. Thus, an observation $z_t = \{z_t^1, \ldots, z_t^K\}$ consists of a set of K individual measurements $z_t^i = (d_t^i, \varphi_t^i)$, where d_t^i is the distance to the nearest object along the beam of the laser, and φ_t^i is the orientation of the beam in the local coordinate system of the sensor.

The goal of the Monte-Carlo localization (MCL) algorithm is to estimate the posterior distribution $p(x_t \mid z_{1:t}, u_{1:t}, m)$ of the robot's pose x_t at time t conditioned on the observations $z_{1:t}$, odometry measurements $u_{1:t}$, and map of the environment m. The Monte Carlo principle consists in drawing a set of N independent and identically distributed samples x^i from a target distribution $p(x)$ to approximate the target distribution as

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}(x)\,, \tag{2.1}$$

where $\delta_{x^i}(x)$ denotes the Dirac delta located at x^i. Under mild assumptions, the estimate $\hat{p}(x)$ will almost surely converge to the target distribution $p(x)$ as the number of samples N approaches infinity. One of the main advantages of this non-parametric approximation is that it is appropriate to represent complex multimodal distributions, like the ones that arise during global localization, for example.

Unfortunately, in the context of localization, drawing samples directly from the posterior $p(x_t \mid z_{1:t}, u_{1:t}, m)$ can be difficult to realize. To solve this problem, a technique called *importance sampling* (see [Andrieu *et al.*, 2003]) is used. The idea is to draw the samples from an easy-to-sample proposal distribution $\pi(x)$ and approximate the target distribution $p(x)$ as

$$\hat{p}(x) = \sum_{i=1}^{N} w(x^i)\, \delta_{x^i}(x)\,, \tag{2.2}$$

where $w(x)$ is referred to as the *importance weight*, and is defined as

$$w(x) = \frac{p(x)}{\pi(x)}\,. \tag{2.3}$$

The importance weight is a value greater than zero and the sum of all the weights must add up to 1. Furthermore, in order for Equation (2.2) to be correct, the support of the proposal distribution $\pi(x)$ must include the support of the target distribution $p(x)$. In essence, the importance weight accounts for the dissimilarity between the target and the proposal distribution. Importance sampling allows us to sample from an easy-to-sample distribution to approximate another not-so-easy-to-sample by attaching an importance weight according to Equation (2.3) to each sample.

The MCL algorithm is, as its name implies, based on the Monte Carlo principle. It represent the posterior $p(x_t \mid z_{1:t}, u_{1:t}, m)$ of the pose x_t of the robot at time t us-

ing a set S_t of N weighted samples or particles, where each particle corresponds to a potential pose of the robot. The general MCL approach is described in Algorithm 2.1. The input of the algorithm is the particle set S_{t-1} at the previous time step $t-1$, the most recent odometry measurement u_t, and the most recent observation z_t. The algorithm consists basically in three steps: *sampling*, *importance weighting*, and *resampling*. In the sampling step (line 3 in Algorithm 2.1) the next generation of particles is created by drawing N samples from the proposal distribution $p(x_t \mid x_{t-1}, u_t)$. This distribution corresponds to a probabilistic motion model of the robot that describes a posterior density over possible poses x_t given the previous pose x_{t-1} and most recent odometry measurement u_t. In the importance weighting step (line 4) the importance weight w_t^i for each previously drawn particle x_t^i is calculated according to the observation model $p(z_t \mid x_t, m)$. The observation model represents the likelihood of the most recent observation z_t given the map of the environment m and the pose x_t. In the final resampling step (lines 8 through 11) the resulting set of particles S_t is created by drawing with replacement N particles from the temporary set $\bar{S}_t$. The probability of drawing a sample x_t^i is proportional to its weight w_t^i, therefore particles with high weights are more likely to be included in the resulting set than particles with low weights. The resampling step can be though of as a probabilistic version of the natural selection mechanism of evolution. Figure 2.1 illustrates the MCL approach in the context of global localization. The particles, corresponding to potential robot poses, are initially distributed uniformly over the free space in the map. As the robot moves through the environment and new evidence, i.e., odometry information and observations, becomes available, unlikely particles are discarded and the remaining particles concentrate around the most likely poses.

Odometry Motion Model

The sampling step of the MCL algorithm requires a motion model which is used as proposal distribution to draw the next generation of samples based on the previous pose of the robot and most recent odometry measurement. The motion model describes probabilistically how the pose of the robot changes as the result of control actions. The model represents the posterior distribution $p(x_t \mid x_{t-1}, u_t)$ over robot poses given the robot pose x_{t-1} at the previous time step and control action u_t. The motion model of the robot strongly depends on the type of locomotion and the hardware of the robotic platform. Throughout this thesis, odometry readings are interpreted as control actions. They describe the relative motion of the robot between two poses and are commonly

Algorithm 2.1 The Monte Carlo localization algorithm

Input: Particle set S_{t-1}, odometry measurement u_t, and observation z_t.

1: $\bar{S}_t = \emptyset$
2: **for** $i = 1$ to N **do**
3: draw $x_t^i \sim p(x_t \mid x_{t-1}^i, u_t)$
4: $w_t^i = p(z_t \mid x_t^i, m)$
5: $\bar{S}_t = \bar{S}_t \cup \{\langle x_t^i, w_t^i \rangle\}$
6: **end for**
7: $S_t = \emptyset$
8: **for** $i = 1$ to N **do**
9: draw x_t^i from $\bar{S}_t$ with probability proportional to w_t^i
10: $S_t = S_t \cup \{\langle x_t^i, 1/N \rangle\}$
11: **end for**
12: **return** S_t

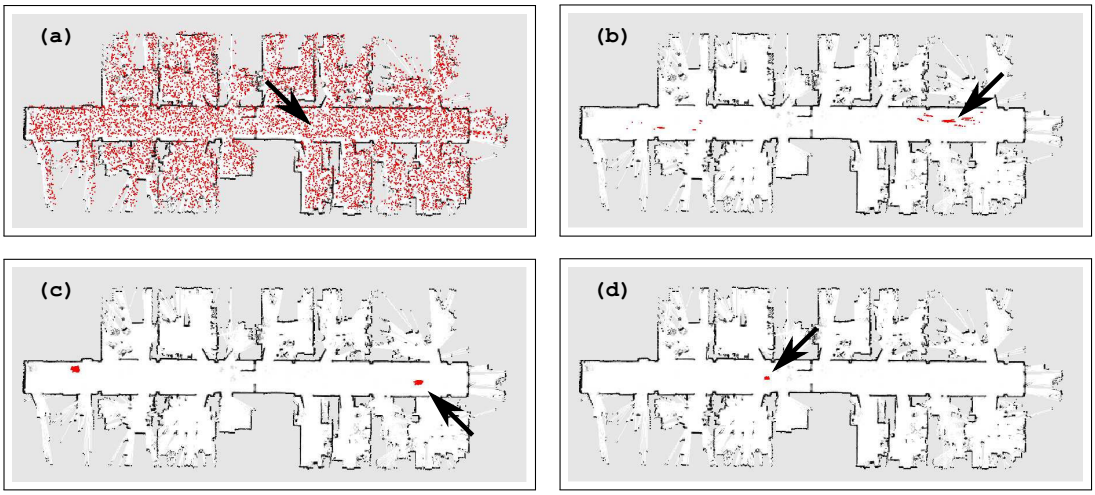


Figure 2.1: Illustration of the Monte-Carlo localization approach in the context of global localization. The arrow indicates the true pose of the robot and the red dots represent the particles corresponding to potential robot poses. **(a)** The particles are initially distributed uniformly over the free space in the map. **(b) - (d)** As the robot moves through the environment unlikely particles are discarded and the remaining particles concentrate around the most likely poses.

obtained from the wheel encoders of the robot.

One way to represent the relative motion u_t of the robot between two poses x_{t-1} and x_t, proposed by [Hähnel *et al.*, 2003a], is to decompose the movement into a sequence of three independent steps or actions: an initial rotation δ_{rot1}, a straight line

motion or translation δ_{trans}, and a final rotation δ_{rot2}. These independent components can be computed as

$$\begin{aligned}\delta_{rot1} &= \operatorname{atan2}(r_t^y - r_{t-1}^y, r_t^x - r_{t-1}^x) - r_{t-1}^\theta \\ \delta_{trans} &= \sqrt{(r_t^x - r_{t-1}^x)^2 + (r_t^y - r_{t-1}^y)^2} \\ \delta_{rot2} &= r_t^\theta - r_{t-1}^\theta - \delta_{rot1}\,.\end{aligned}$$

Assuming that these three parameters are corrupted by independent noise allows us to easily sample poses from the distribution $p(x_t \mid x_{t-1}, u_t)$ given an initial pose x_{t-1} and odometry reading u_t. This motion model is particularly well-suited for robots equipped with a differential drive that, for example, cannot move sideways. Other motion models for differential drives and other types of drives have been proposed in the literature, but if the pose of the robot is estimated frequently enough, the effect of using a different model is usually insignificant.

Observation Model

Besides the motion model, the MCL algorithm also requires an observation model to compute the importance weights of the particles. The observation model describes the probability $p(z_t \mid x_t, m)$ of making an observation z_t given the pose x_t of the robot and the map m of the environment. As the motion model, the observation model also strongly depends on the type of sensor used by the robot to perceive its environment. In this thesis, we consider only laser range scanners. These type of sensors are very common in robotics and state-of-art for distance measurements given their high accuracy. Laser scanners provide, at any given point in time t, a set of N individual measurements z_t^i. These are assumed to be independent from each other and, therefore, the probability $p(z_t \mid x_t, m)$ of an observation z_t can be computed as the product of the probabilities of the individual measurements

$$p(z_t \mid x_t, m) = \prod_{i=1}^{N} p(z_t^i \mid x_t, m)\,. \tag{2.4}$$

Each individual measurement $z_t^i = (d_t^i, \varphi_t^i)$ consists of a range measurement d_t^i with an associated orientation φ_t^i in the local coordinate system of the sensor. The value of d_t^i corresponds to the distance to the nearest object along the beam of the laser. The map m represents the location of the relevant objects in the environment.

The likelihood of each range measurement z_t^i is computed according to a mixture of three distributions corresponding to three different types of measurement errors: *measurement noise*, *measurement failures*, and *random measurements*. The measurement noise is modeled by a narrow Gaussian distribution $p_{hit}(z_t^i) \sim \mathcal{N}(d, \sigma_{hit}^2)$ with mean d and standard deviation σ_{hit}. The value of d corresponds to the distance between the endpoint of the measurement z_t^i and its closest obstacle in the map m. Measurement failures typically result in maximum range readings, that is, the sensor returns its maximum allowable value z_{max}. This type of error is modeled as a point-mass distribution $p_{max}(z_t^i)$ centered at z_{max}. Finally, random measurements are modeled with a uniform random distribution $p_{rand}(z_t^i)$ over the entire allowable range of the sensor. The resulting probability $p(z_t^i \mid x_t, m)$ for an individual range measurement is given by the mixture

$$p(z_t^i \mid x_t, m) = w_{hit} \cdot p_{hit}(z_t^i) + w_{max} \cdot p_{max}(z_t^i) + w_{rand} \cdot p_{rand}(z_t^i)\,, \tag{2.5}$$

where w_{hit}, w_{max}, and w_{rand} are positive weighting factors that sum up to 1. This model is known as the *likelihood field* model and was first proposed by [Thrun, 2001]. Its key advantage is that the resulting distribution $p(z_t \mid x_t, m)$ is smooth: small changes in the pose of the robot produce only small changes in the distribution. This is a desirable property, in particular for the MCL algorithm, since particles that are close to each other will be assigned similar weights. Under the premise that a particle in the correct pose is assigned a high weight, then all near particles will also be assigned a similarly high weight. A highly discontinuous distribution $p(z_t \mid x_t, m)$, on the other hand, would assign nearby particles different — probably considerably lower — weights, and particles that are actually close to the correct pose could be eliminated in the resampling step. The main disadvantage of the likelihood field model is that it explicitly disregards the geometry of the environment and the physics of the range finder. Figure 2.2 shows a map of an environment with its associated likelihood field.

2.2 Grid Maps

Mobile robot localization, as well as path planning and other functionalities fundamental for safe and reliable navigation, required a map or representation of the robot's environment. Many different representations have been proposed in the robotics literature. For robots operating in planar environments, grid maps are one of the most commonly used representations. A grid map is a tessellation of the space into a num-

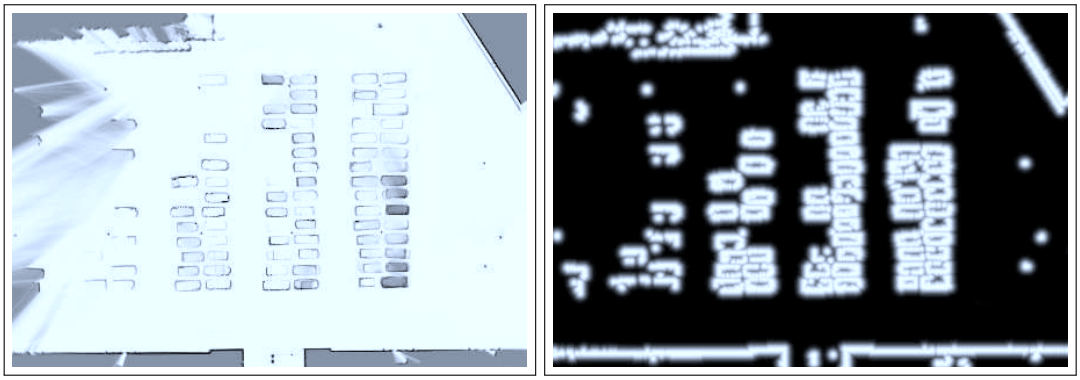


Figure 2.2: Occupancy grid map of the parking lot of the Department of Computer Science at the University of Freiburg (left) and associated likelihood field (right). The darker the color of a location in the likelihood field, the lower the likelihood for a beam to end up in that location.

ber of cells where each cell contains information about some features of its associated space. In the context of robot localization, the tessellation of the space is usually regular, the cells are rectangular, and the presence of an obstacle is the most commonly considered feature of the environment. Figure 2.3 shows two grid maps corresponding to two different environments. The darker the color of an area, the higher the likelihood for it of being occupied by an obstacle. Grid maps are popular because they allow constant time access to the information in a cell and are straightforward to implement. They don't necessarily rely on predefined features that need to be extracted from sensor data. Furthermore, they are also able to represent areas of the environment for which no data is available. The main disadvantages of grid maps is that the discretization process can create artifacts in the representation and they don't scale well to large environments. In the following section we describe an approach called *Occupancy Probability Mapping* for creating grid maps using sensor measurements.

2.2.1 Occupancy Probability Mapping

The goal of the occupancy probability mapping algorithm is to estimate the posterior $p(m \mid z_{1:t}, x_{1:t})$ over the map m given the sequence $z_{1:t}$ of sensor readings obtained by the robot at poses $x_{1:t}$. Grid maps partition the space into a finite number of grid cells, so that $m = \{c_i\}$ where c_i denotes the grid cell with index i. Each cell c_i is assumed to be either free or occupied and can be though of as a binary variable where $p(c_i)$ denotes the probability for the cell of being occupied. In this section we present the occupancy

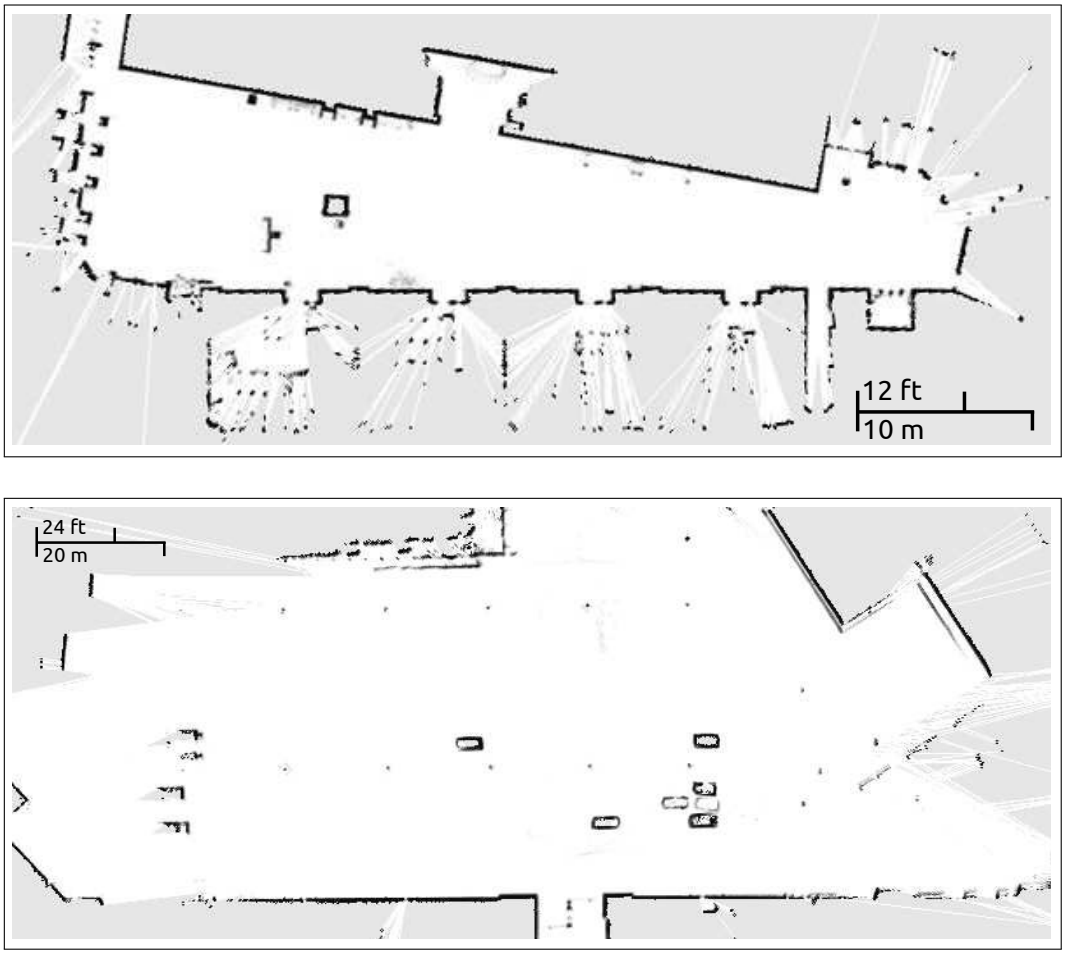


Figure 2.3: Occupancy grid map of the foyer of *building 101* (top) and parking lot (bottom) of the Department of Computer Science at the University of Freiburg. The darker the color of an area, the higher the likelihood of being occupied by an obstacle. A resolution of 0.1 and 0.2 meters were used, respectively, for the cells of the top and bottom grids.

probability mapping algorithm introduced by Moravec and Elfes [Moravec and Elfes, 1985] for computing the the posterior $p(m \mid z_{1:t}, x_{1:t})$. The derivation presented in this section, follows the one presented by [Hähnel, 2004] and [Stachniss, 2006].

The algorithm assumes that the cells in the map are independent from each other and estimates the occupancy probability $p(c_i \mid z_{1:t}, x_{1:t})$ of each cell c_i individually. The posterior $p(m \mid z_{1:t}, x_{1:t})$ over the map is thus approximated as the product of the posteriors $p(c_i \mid z_{1:t}, x_{1:t})$ over the individual cells c_i in the map

$$p(m \mid z_{1:t}, x_{1:t}) = \prod_i p(c_i \mid, z_{1:t}, x_{1:t}) \,, \tag{2.6}$$

where

$$p(c_i \mid z_{1:t}, x_{1:t}) = \frac{p(z_t \mid c_i, z_{1:t-1}, x_{1:t})\, p(c_i \mid z_{1:t-1}, x_{1:t})}{p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.7}$$

The previous equation is obtained by applying Bayes' rule using $z_{1:t-1}$ and $x_{1:t}$ as background knowledge. Assuming that the observation z_t is independent from $z_{1:t-1}$ and $x_{1:t-1}$ leads to

$$p(c_i \mid z_{1:t}, x_{1:t}) = \frac{p(z_t \mid c_i, x_t)\, p(c_i \mid z_{1:t-1}, x_{1:t})}{p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.8}$$

Reformulating the term $p(z_t \mid c_i, x_t)$ in the previous expression according to Bayes' rule and assuming that c_i is independent from the robot's pose x_t if there is no sensor reading z_t we obtain

$$p(c_i \mid z_{1:t}, x_{1:t}) = \frac{p(c_i \mid z_t, x_t)\, p(z_t \mid x_t)\, p(c_i \mid z_{1:t-1}, x_{1:t-1})}{p(c_i)\, p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.9}$$

Since it is assumed that each cell c_i is either free or occupied, the following equation is derived in an analogous way

$$p(\neg c_i \mid z_{1:t}, x_{1:t}) = \frac{p(\neg c_i \mid z_t, x_t)\, p(z_t \mid x_t)\, p(\neg c_i \mid z_{1:t-1}, x_{1:t-1})}{p(\neg c_i)\, p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.10}$$

Dividing Equation (2.9) by Equation (2.10) we obtain

$$\frac{p(c_i \mid z_{1:t}, x_{1:t})}{p(\neg c_i \mid z_{1:t}, x_{1:t})} = \frac{p(c_i \mid z_t, x_t)\, p(\neg c_i)\, p(c_i \mid z_{1:t-1}, x_{1:t-1})}{p(\neg c_i \mid z_t, x_t)\, p(c_i)\, p(\neg c_i \mid z_{1:t-1}, x_{1:t-1})} \,, \tag{2.11}$$

and, given that $p(\neg c_i) = 1 - p(c_i)$, the previous equation can be reformulated as

$$\frac{p(c_i \mid z_{1:t}, x_{1:t})}{1 - p(c_i \mid z_{1:t}, x_{1:t})} = \frac{p(c_i \mid z_t, x_t)}{1 - p(c_i \mid z_t, x_t)} \cdot \frac{1 - p(c_i)}{p(c_i)} \cdot \frac{p(c_i \mid z_{1:t-1}, x_{1:t-1})}{1 - p(c_i \mid z_{1:t-1}, x_{1:t-1})} . \quad (2.12)$$

Finally, using the log odds representation

$$l(x) = \ln \frac{p(x)}{1 - p(x)} ,$$

Equation (2.12) can be written as

$$l(c_i \mid z_{1:t}, x_{1:t}) = l(c_i \mid z_t, x_t) - l(c_i) + l(c_i \mid z_{1:t-1}, x_{1:t-1}) , \quad (2.13)$$

where $l(c_i)$ corresponds to the prior occupancy of the cell c_i represented as log odds. In practice, the prior is often assume to be 0.5. In this case, the term $l(c_i)$ can be omitted from the equation.

Using Equation (2.13), the belief about the occupancy of a cell can be computed incrementally as sensor readings become available. The log odds representations allows the computations to be performed efficiently using sums and avoids numerical instabilities for extreme probabilities.

Inverse measurement model

It remains to describe how to compute the inverse measurement model $p(c_i \mid z_t, x_t)$ of a cell given a single sensor reading z_t and corresponding robot pose x_t. This probability corresponds to the term $l(c_i \mid z_t, x_t)$ in Equation (2.13). In contrast to the observation model specified by Equation (2.4), the inverse measurement model reasons from effects to causes.

This model strongly depends on the sensor used. As already mentioned, in this thesis we deal only with laser range finders that generate, at each time t, a sensor reading $z_t = \{z_t^1, \ldots, z_t^K\}$ consisting of K range measurements which are assumed to be independent from each other. Let $C_t^i = \{c_t^{i,1}, \ldots, c_t^{i,N}\}$ be the cells covered by range measurement z_t^i. The last cell $c_t^{i,N}$, corresponds to the cell where the the measurement z_t^i ends. The occupancy probability $p(c_i \mid z_t^i, x_t)$ of map cell c_i given a

single range measurement z_t^i and pose x_t can be computed as

$$p(c_i \mid z_t^i, x_t) = \begin{cases} p_{occ} & \text{if } c_i = c_t^{i,N} \\ p_{free} & \text{if } c_i \in C_t^i \wedge c_i \neq c_t^{i,N} \\ 0.5 & \text{otherwise,} \end{cases} \tag{2.14}$$

where $0 < p_{free} < 0.5 < p_{occ} \leq 1$. The expression above simply states that a cell covered by the range measurement is considered as free if the measurement does not end in it. On the other hand, a cell in which a range measurement ends is considered occupied. The occupancy probability of the cells not covered by the measurement remain unchanged. Range measurements corresponding to a maximum range reading are ignored.

2.3 Hidden Markov Models

The robot localization problem presented in the first section of this chapter, is an instance of the general problem of state estimation in dynamic systems. In the localization problem the state of the system is the pose of the robot, the motion model describes the evolution of the system over time, and the observation model relates the observations or evidence about the state of the system with the actual state. In this section, we describe the general problem of state estimation in dynamic systems and present a concrete temporal probabilistic model called the hidden Markov model (see [Rabiner, 1989]) for characterizing and reasoning in dynamic systems.

In a hidden Markov model (HMM) the state of the system at time t is represented by a discrete random variable x_t. This random variable represents the relevant aspects of the system at time t and can be though of as a snapshot of the system at that specific time. It is assumed that the state of the system depends only on a finite history of previous states. This is known as the *Markov assumption*. Commonly the state x_t is only conditioned on the previous state x_{t-1}. Furthermore, it is assumed that the changes in the system are caused by stationary processes. These two assumptions are summarized by the state transition model

$$p(x_t \mid x_{1:t-1}) = p(x_t \mid x_{t-1}), \quad \forall t. \tag{2.15}$$

In addition to the Markov and stationary process assumptions, it is also assumed that the observation z_t at time t depends only on the current state of the system x_t.

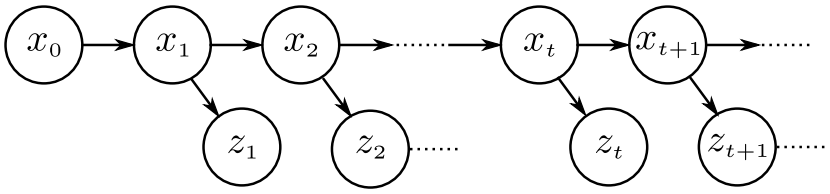


Figure 2.4: Graphical representation of a dynamic system described by a hidden Markov model. A discrete random variable x_t represents the hidden state of the system at time t. The observation or evidence at time t are represented by the random variable z_t. The arrows indicate the dependencies between the variables.

This is expressed by the observation model

$$p(z_t \mid x_{1:t}, z_{1:t-1}) = p(z_t \mid x_t), \quad \forall t. \tag{2.16}$$

The observations are the result of a stochastic process that depends on the state of the system. The actual state of the system is not directly observable, that is, it is *hidden* and can only be inferred through the observations. Although they cannot be observed directly, in many practical applications the states are associated to some meaningful interpretation. Figure 2.4 shows a graphical representation of a dynamic system described by an HMM.

Besides the transition and observation models, an HMM also requires the specification of the prior or initial state distribution $p(x_{t=0})$ to characterize the complete joint distribution $p(x_{0:t}, z_{1:t})$ over all the variables

$$p(x_{0:t}, z_{1:t}) = p(x_{t=0}) \prod_{\tau=1}^{t} p(x_\tau \mid x_{\tau-1})\, p(z_\tau \mid x_\tau)\,. \tag{2.17}$$

Formally, an HMM is characterized by the following elements:

- A set $Q = \{q_1, q_2, \ldots q_N\}$ of N possible model states. As already mentioned, the state of the system at time t is represented by the random variable x_t.
- A set $V = \{v_1, v_2, \ldots v_M\}$ of M possible observations. These correspond to the evidence about the states of the model that can be directly observed.
- The state transition model $p(x_t = q_i \mid x_{t-1} = q_j)$ for all states q_i and q_j. According to the stationary process assumption, the transition probabilities only need to be specified for every pair of states since they do not change over time. For

notational convenience, the probability of changing from state q_j to state q_i is sometimes denoted as a_{ji}.

- The observation model $p(z_t = v_k \mid x_t = q_i)$ for all states q_i and observations v_k. Analogous to the transition probabilities, the observation probabilities need to be specified only for every pair of state and observation. The probability of observing v_k in state q_i is sometimes represented as $b_i(k)$.

- The initial state probability $p(x_{t=0} = q_i)$ for all states. This probability can also be denoted as π_i.

Compactly, a hidden Markov model can be specified as $\lambda = (A, B, \pi)$, where $A = \{a_{ij}\}$, $B = \{b_i(k)\}$, and $\pi = \{\pi_i\}$. The structure of the HMM allows for simple and elegant implementations of the basic inference tasks. In the next section we describe how the filtering and prediction tasks can be solved for HMMs.

2.3.1 Filtering and Prediction

Filtering consists in estimating the state of the system at some specific time given all previous evidence. This corresponds to estimating the *belief* or posterior distribution $p(x_t \mid z_{1:t})$ over the state x_t given the sequence of observations $z_{1:t}$. From the discussion in Section 2.1.1, it can be seen that the MCL approach is a filtering process where the pose of the robot is estimated given all previous observations and odometry measurements.

Filtering is performed recursively. The belief at time t is computed from the belief at time $t-1$. Using Bayes' rule, the belief at time t can be written as

$$p(x_t \mid z_{1:t}) = \eta\, p(z_t \mid x_t, z_{1:t-1})\, p(x_t \mid z_{1:t-1})\,, \tag{2.18}$$

where η is a normalizing constant to make the probabilities sum up to 1. Since it is assumed that the observation z_t at time t depends only on the state of the system x_t at time t, Equation (2.18) can be simplified as

$$p(x_t \mid z_{1:t}) = \eta\, p(z_t \mid x_t)\, p(x_t \mid z_{1:t-1})\,. \tag{2.19}$$

The second term on the right hand side of the previous equation, $p(x_t \mid z_{1:t-1})$ represents a prediction of the belief at time t based on the evidence up to time $t-1$.

Applying the theorem of total probability conditioning on x_{t-1} we obtain

$$p(x_t \mid z_{1:t}) = \eta\, p(z_t \mid x_t) \sum_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1})\, p(x_{t-1} \mid z_{1:t-1})\,. \tag{2.20}$$

According to the Markov assumption, the state x_t is only conditioned on the previous state x_{t-1}. Therefore, Equation (2.20) can be simplified as

$$p(x_t \mid z_{1:t}) = \eta\, p(z_t \mid x_t) \sum_{x_{t-1}} p(x_t \mid x_{t-1})\, p(x_{t-1} \mid z_{1:t-1})\,. \tag{2.21}$$

The first factor within the summation, $p(x_t \mid x_{t-1})$, corresponds to the state transition model, and the second factor corresponds to the belief at time $t-1$. This recursive filtering approach is known as the Bayes filter and for HMMs can be implemented in a very simple and elegant way using matrices and vectors.

Prediction is the task of estimating the posterior distribution $p(x_{t+k} \mid z_{1:t})$ over the future state x_{t+k} at time $t+k$ given the sequence of observations $z_{1:t}$ up to time t. Prediction can be considered as filtering without the processing of evidence. The prediction at $t+k+1$ is computed recursively from the prediction at time $t+k$ as

$$p(x_{t+k+1} \mid z_{1:t}) = \sum_{x_{t+k}} p(x_{t+k+1} \mid x_{t+k})\, p(x_{t+k} \mid z_{1:t})\,. \tag{2.22}$$

2.3.2 Parameter Estimation

As mentioned above, a hidden Markov model is specified by the state transition probabilities $A = \{a_{ij}\}$, the observation model $B = \{b_i(k)\}$, and the initial state probabilities $\pi = \{\pi_i\}$. These parameters are usually determined from data, that is, from one or multiple sequences of observations that are assumed to be generated by the model under consideration.

One of the most popular approaches for estimating the parameters of an HMM is an instance of the expectation-maximization (EM) algorithm. The idea is to start with some initial value for the parameters and, based on the training data, use inference to obtain an estimate of the hidden states that generated the data. Then, the parameters of the model are re-estimated based of the obtained hidden states. These two steps, called the *expectation* and *maximization* steps respectively, are repeated until convergence. Let $\hat{\theta}^n = (\{\hat{a}^n_{ij}\}, \{\hat{b}^n_{jk}\})$ represent the parameters estimated at the n-th iteration and let $z_{1:T}$ be the observation sequence used for estimating the parameters. The EM

algorithm results in the following re-estimation formula for the transition model

$$\hat{a}_{ij}^{n+1} = \frac{\sum_{\tau=1}^{T} p(x_{\tau-1} = q_i, x_\tau = q_j \mid z_{1:T}, \hat{\theta}^n)}{\sum_{\tau=1}^{T} p(x_{\tau-1} = q_i \mid z_{1:T}, \hat{\theta}^n)}\,, \tag{2.23}$$

and observation model

$$\hat{b}_{jk}^{n+1} = \frac{\sum_{\tau=1}^{T} p(z_\tau = v_k, x_\tau = q_j \mid z_{1:T}, \hat{\theta}^n)}{\sum_{\tau=1}^{T} p(x_\tau = q_i \mid z_{1:T}, \hat{\theta}^n)}\,, \tag{2.24}$$

Note that the probabilities on the right-hand side are conditioned on the observation sequence $z_{1:T}$ and the previous parameter estimates $\hat{\theta}^n$. These probabilities can be efficiently computed using the *forward-backward* procedure [Rabiner, 1989].

The algorithm computes the parameters of the model that locally maximize the likelihood of the training data. However, there is no guarantee for this local maximum to be a global maximum too. The results strongly depend on the choice of the initial parameters. The right parameters can lead to a global maximum, while the wrong parameters can even lead to a trivial solution. There is no straightforward approach for selecting good initial parameters. In practice, random or uniformly distributed initial values for the initial state and transition probabilities usually lead satisfactory results. For the observation model, however, better initial parameters are usually needed. One way to obtain more informed initial estimates is, for example, to manually segment the observations according to the states in the model and then average the observations within each segment. Alternatively, some clustering technique, like k-means, could be used to group the observations according to states.

Another problem associated to estimating the parameters is that of insufficient training data. Usually, the number of state transitions and observations is not enough to obtain appropriate estimates. One obvious solution to this problem is to obtain more training data. This is, however, not always possible or practical. Alternatively, the size of the model can be reduced, for example, by reducing the number of states. This solution is, however, can lead to inadequate models.

Part I

Mobile Robot Navigation in Dynamic Environments

Chapter 3

Temporary Maps for Robust Localization in Semi-Static Environments

Most mobile robot localization approaches, including the Monte-Carlo localization approach presented in the previous chapter, rest on the simplifying assumption of a static environment. A reference map of the environment is provided beforehand and is then used as ground truth disregarding potential subsequent changes. To cope with non-static environments, one popular technique is simply to ignore the measurements that are not explained by the reference map. This assumes that objects are either static and represented in the map or dynamic and should be ignored for localization. Whereas this technique has been demonstrated to be robust in highly dynamic environments, it ignores valuable localization information when the changes take place infrequently.

In this chapter, we describe a localization framework that exploits the measurements caused by certain non-static objects to build local maps that temporarily extend the reference map of the environment. Using these temporary maps, the robot can reliably estimate its pose also in regions that are subject to persistent changes. The motivation behind this approach is that many objects, referred to as *semi-static*, change their locations with a relatively low frequency and therefore provide important localization information. For example, consider a parking lot as the one depicted in Figure 3.1. In such an environment there are only a few static objects. Additionally, the parked cars occlude them most of the time. As a result almost no features remain that can be used for localization by standard approaches. The parked cars, however, provide on their own important features for localization.

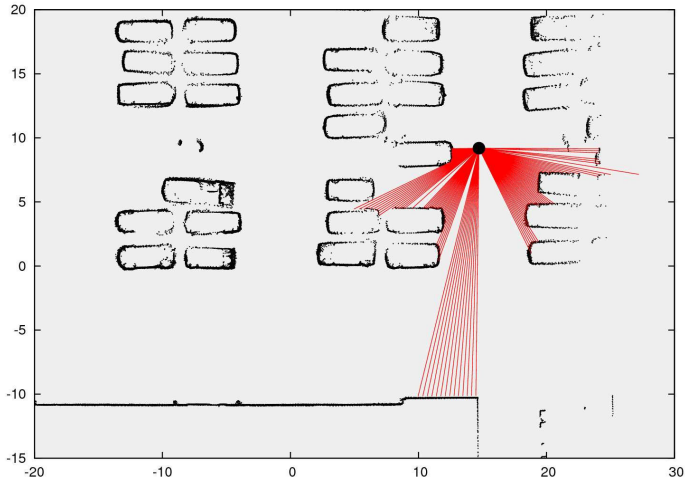


Figure 3.1: In large open spaces like parking lots, semi-static objects (parked cars) provide an abundant source of features for localization. The static objects (walls) are not only few but are also occluded by the non-static objects.

As already mentioned, in many practical mobile robot applications, a reference map representing the static parts of the environment is available beforehand. However, most interesting environments are not static and mobile robots must be able to deal with changes in the environment. Our proposed approach is an extension of the Monte-Carlo localization (MCL) approach for static environments presented in Chapter 2. We assume that a reference map representing the static objects in the environment is given. When the observations of the robot are consistent with this map, the approach corresponds exactly to the standard MCL approach. However, we also use temporary maps to keep track of the inconsistent observations caused by semi-static objects. Whenever the robot enters an area for which a temporary map already exists we try to use this map to improve the localization. Taking advantage of the measurements caused by semi-static objects is particularly important in large open spaces like the parking lot in Figure 3.1 or warehouses, where the static parts of the environment are few and usually occluded, but many semi-static objects provide valuable localization information.

In this chapter we present a localization approach capable of dealing with semi-static environments. We provide a probabilistic formulation of the localization problem where the semi-static aspects of the environment are explicitly modeled. We keep track of the observations caused by the semi-static objects in the environment in the form of local maps that temporarily extend the static map of the environment. At its

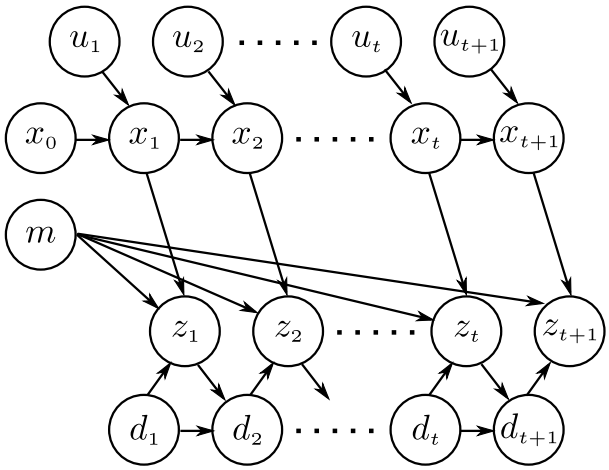


Figure 3.2: Graphical model of the mobile robot localization process in semi-static environments. The observations z_t at time step t are explained by both the static map m and the corresponding semi-static map d_t.

core, our localization framework relies on the MCL approach for estimating the pose of the robot using the extended static map. Experimental results demonstrate that by exploiting the observations caused by semi-static objects our approach is capable of robustly estimating the pose of the robot where standard approaches fail.

3.1 Localization in Semi-Static Environments

As described in Chapter 2, robot localization consists in estimating the probability density $p(x_t \mid z_{1:t}, u_{1:t}, m)$ of the robot's pose x_t in a known map m, given a sequence of observations of the environment $z_{1:t}$ and odometry measurements $u_{1:t}$. Most existing solutions to the localization problem assume that the objects in the environment are either static and represented in the map or dynamic and should be ignored for localization. Accordingly, these approaches classify objects into two classes: static and dynamic. In contrast, we classify the objects in the environment into three different classes according to their dynamics

- ***static objects***: like buildings, that do not change their location.
- ***semi-static objects***: like parked cars, that change their locations with a relatively low frequency. In particular, we assume that these objects do not change their location while the robot is observing them.

- ***dynamic objects***: like moving people, that frequently change their location. Unlike semi-static objects, dynamic objects do change their location while being observed by the robot.

We assume that dynamic objects are detected and filtered out. The map m represents the static objects in the environment whereas the map d_t represents the semi-static objects at time t. Figure 3.2 depicts the dynamic Bayesian network describing the localization process in a semi-static environment. The main difference to standard localization approaches in static environments is that we explicitly model the fact that the observation z_t at time step t is explained by both, the static map m and the semi-static map d_t. Additionally, we model the transition probability $p(d_t \mid d_{t-1}, z_{t-1})$ that characterizes the temporal dependency between semi-static maps.

According to our formulation, robot localization in semi-static environments requires to jointly estimate the probability distribution $p(x_t, d_t \mid z_{1:t}, u_{1:t}, m)$ over robot poses and semi-static maps. To this end one could use one of the many simultaneous localization and mapping (SLAM) algorithms available (see [Thrun, 2002] for a survey on the literature on the field) and initialize it with the known static map. However, the majority of SLAM approaches is based on the assumption that the environment is static and the presence of non-static objects can lead to serious errors in the resulting maps as we demonstrate in the experimental section of this chapter. Exploiting the conditional independence assumptions encoded in the dynamic Bayesian network of Figure 3.2 and applying Baye's rule we can write the SLAM posterior as

$$p(x_t, d_t \mid z_{1:t}, u_{1:t}, m) = \eta\, p(z_t \mid x_t, d_t, m)\, p(x_t, d_t \mid z_{1:t-1}, u_{1:t}, m)\,. \tag{3.1}$$

Applying the theorem of total probability we can rewrite the previous equation as

$$\begin{aligned} &p(x_t, d_t \mid z_{1:t}, u_{1:t}, m) = \eta\, p(z_t \mid x_t, d_t, m) \\ &\iint p(x_t, d_t \mid x_{t-1}, d_{t-1}, z_{t-1}, u_t)\, p(x_{t-1}, d_{t-1} \mid z_{1:t-1}, u_{1:t-1}, m)\, \mathrm{d}d_{t-1}\, \mathrm{d}x_{t-1}\,. \end{aligned} \tag{3.2}$$

Note that the static map m, the odometry measurements $u_{1:t-1}$, and the observations $z_{1:t-2}$ up to time step $t-2$ can be dropped from the evidence in the first term inside the integral because x_t and d_t do not depend on them given x_{t-1}, d_{t-1} and z_{t-1} (see Figure 3.2). Similarly, we drop the odometry measurement u_t at time t from the evidence in the second term inside the integral which then turns into a recursive term that corresponds to the SLAM posterior at the previous time step.

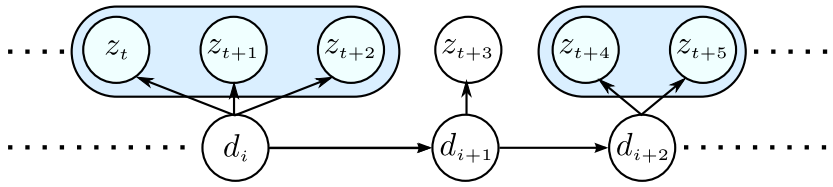


Figure 3.3: Since semi-static objects change their locations with a relatively low frequency, one semi-static map d_i can explain observations at different time steps.

According to the conditional independence assumptions in our model, the first term inside the integral can be written as

$$p(x_t, d_t \mid x_{t-1}, d_{t-1}, z_{t-1}, u_t) = p(x_t \mid x_{t-1}, u_t)\, p(d_t \mid d_{t-1}, z_{t-1})\,. \tag{3.3}$$

This holds by the rules of *d-separation* since given the evidence x_{t-1}, d_{t-1}, z_{t-1}, and u_t, the pose x_t of the robot and the map d_t are conditionally independent from each other. Given x_{t-1}, the map d_{t-1} and observation z_{t-1} at time step $t-1$ provide no additional information about the pose of the robot x_t at time t. Similarly, given d_{t-1} and z_{t-1}, the pose x_{t-1} of the robot at time step $t-1$ and odometry measurement u_t at time t provide no additional information about the map d_t at time t. The distribution $p(x_t \mid x_{t-1}, u_t)$ in Equation (3.3) corresponds to the motion model of the robot and $p(d_t \mid d_{t-1}, z_{t-1})$ represents the temporal dependency between maps d_t and d_{t-1}.

In Chapter 4 we present an approach to learn the temporal dependency between maps at consecutive time steps. In this chapter, however, we assume that a map either consistently explains the measurements z_t at time t or it is not valid anymore and needs to be re-estimated. This transition model for maps can be formulated as

$$p(d_t \mid d_{t-1}, z_{t-1}) = \begin{cases} \delta_{d_{t-1}}(d_t) & \text{if } z_{t-1} \text{ is consistent with the map } d_{t-1} \\ \mathcal{U}(d_t) & \text{otherwise.} \end{cases} \tag{3.4}$$

The expression $\delta_{d_{t-1}}(d_t)$ denotes the Dirac delta located at d_{t-1} and $\mathcal{U}(d_t)$ corresponds to a uniform distribution about potential maps at time t. Figure 3.3 provides an intuitive interpretation for this model; semi-static objects change their location with a relatively low frequency and, therefore, we assume that the map does not necessarily change between consecutive time steps. However, when the semi-static objects do change, that is, when the existing map does not consistently explain the measurements anymore, it is disregarded, and a new map is built from scratch. In the following section we present our approach for robot localization in semi-static environments in more detail.

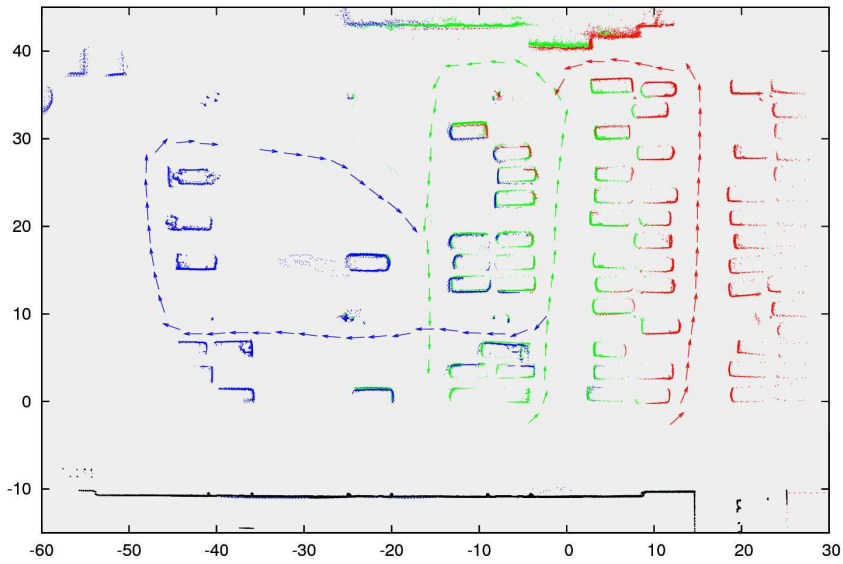


Figure 3.4: Semi-static maps extending the static map of a parking lot. The static map of the environment (structures at the bottom in black) is shown together with three different semi-static maps (colored in blue, green, and red).

3.2 Temporary Maps

The key idea of our approach is to use the measurements caused by semi-static objects to improve the localization capabilities of the robot. With these measurements, we build local semi-static maps that temporarily extend the static map of the environment. Due to the temporal nature of these maps we refer to them both as semi-static or temporary maps.

A semi-static map represents the semi-static objects in the environment as observed by the robot while navigating through it. As illustrated in Figure 3.3, a semi-static map d_i is associated to a sequence of measurements $z_{n:m}$ with their corresponding poses $x_{n:m}$. Semi-static maps are created as the robot navigates through the environment using a maximum-likelihood approach. Figure 3.4 depicts a static map of a parking lot extended by three semi-static maps.

As already mentioned, our localization framework extends the MCL approach described in Chapter 2. Instead of relying on the static map of the environment, our approach uses the extended static map for estimating the pose of the robot. When the observations obtained by the robot are consistent with the static map of the environ-

ment, we use this map as reference to estimate the pose of the robot. However, when the observations are inconsistent, we select the local map closest to the current pose of the robot. If a map is found, we try to use it as reference for localization instead of the static map. In the remainder of this section we describe the different components of our localization framework.

3.2.1 Perceptual Model

We assume that the observations are obtained from a range scanner and that each observation z_t consists of a set of range measurements. To evaluate the likelihood $p(z_t \mid x_t, m)$ of an observation z_t given the pose x_t of the robot and a reference map m, we use the likelihood fields model described in Chapter 2. In this model, the individual range measurements of the observation z_t are assumed to be independent of each other and the likelihood of each one is computed based on the distance between the endpoint of the range measurement and its closest obstacle in the map m.

In our current implementation, we use this distance as a heuristic to decide whether a measurement is explained by the map m or not. Concretely, if the distance to the closest object in the map is larger than a given threshold ϵ_d, we assume that the measurement is inconsistent with the map and consider it an outlier. The outlier ratio $e(z_t, x_t, m)$ for a given observation z_t and pose x_t corresponds to the fraction of range measurements that do not correspond to their expected values according to the map m

$$e(z_t, x_t, m) = \frac{\sum_{i=1}^{K} \mathbb{1}_{\{d_t^i > \epsilon_d\}}}{K}\,, \tag{3.5}$$

where K is the number of range measurements, d_t^i is the distance between the endpoint of the range measurement z_t^i, as observed from pose x_t, and its closest obstacle in the map m, and $\mathbb{1}_{\{d_t^i > \epsilon_d\}}$ is 1 if $d_t^i > \epsilon_d$ and 0 otherwise. Since the belief about the pose x_t of the robot is represented by a set of weighted particles, we use the weighted average outlier ratio $\bar{e}(z_t, m)$ of the particle set given a map as criterion of how well a given observation is explained by the map. This average outlier ratio is computed as

$$\bar{e}(z_t, m) = \frac{\sum_{i=1}^{N} w_t^i \, e(z_t, x_t^i, m)}{\sum_{i=1}^{N} w_t^i}\,, \tag{3.6}$$

where N is the number of particles in the set, and w_t^i and x_t^i are, respectively, the weight and pose associated to the i-th particle.

3.2.2 Constructing Temporary Maps

A temporary or semi-static map consists of a sequence of measurements $z_{n:m}$ with their associated poses $x_{n:m}$ that implicitly represent a part of the environment. To construct such a map we incrementally perform scan matching on consecutive observations. The idea is to compute, for each new observation, the pose that best aligns the observation with respect to a reference map. This map is then extended by incorporating the aligned observation together with its corresponding pose.

The concrete scan matching technique that we use is similar to the correlative scan matching approach proposed by Olson [Olson, 2009]. The idea is to evaluate the observation likelihood $p(z_t \mid x_t, m)$ using a previous scan z_{t-1} as reference map m over a discretized three-dimensional volume of potential poses x_t. The maximum-likelihood pose corresponds to the best alignment between the two scans. Whereas in Olson's approach only a single scan z_{t-1} is used as reference, we use a history of previously aligned scans $z_{t-k:t-1}$. In our experiments, we found that using a history of scans instead of a single one increases the robustness of the scan matching.

One drawback of incrementally constructing maps using scan matching is that the pose estimates are never corrected. To overcome this problem, we adjust the poses of the temporary map whenever the robot enters a known area in the static map or an area for which a temporary map already exists. This problem corresponds to an instance of the graph-based SLAM problem [Grisetti *et al.*, 2010], where the poses of the robot correspond to nodes in a graph. An edge between two nodes represents the relative movement between the corresponding poses as estimated by the scan matcher. In addition to these spatial constraints between consecutive poses we also consider the global constraint that results from the robot entering a known area. This global constraint corresponds to the relative movement of the robot between the first and last pose in the semi-static map, being the last pose the point where the robot reentered a known area.

To efficiently compute the maximum-likelihood semi-static map we use the approach described by Grisetti *et al.* [Grisetti *et al.*, 2010]. Note that the optimization is only performed when reentering a know area. Furthermore, in contrast to the pure SLAM problem, we do not adjust the nodes in the graph that correspond to the robot being in a known area in the static map of the environment.

3.2.3 Extending the Static Map

We assume that a static map of the environment is given. When the observations of the robot are inconsistent with the static map, we select the semi-static map closest to the current pose of the robot and try to use it as reference for localization instead of the static map. Note that our approach is an extension of the standard MCL approach for static environments. Whenever the observations of the robot are consistent with the static map, our approach corresponds exactly to the standard MCL approach.

Semi-static maps are created whenever the following two conditions hold. First, the observations of the robot are inconsistent with the static map of the environment. And second, there is no other semi-static map close to the current pose of the robot that explains the observations and can be used as reference for localization instead of the static map. Whenever these two conditions hold over multiple consecutive time steps a new semi-static map is created as described in the previous section. To determine if an observation is inconsistent with a map or not we use the average outlier ratio of the particle set as described in Section 3.2.1.

We assume that a semi-static map either consistently explains the observations at a given time or it is not valid anymore. Accordingly, semi-static maps are discarded if the average outlier ratio of the particle set is too high over multiple consecutive time steps. Ideally, semi-static maps should only be eliminated if the environment has changed since the moment of its creation. To reduce the problem of incorrectly eliminating a map, the uncertainty in the pose estimate is also taken into account. Maps are discarded if the uncertainty of the particle set is below a given threshold.

3.2.4 Localization Using Temporary Maps

Whenever the static-map of the environment does not explain the observations of the robot, we search for a semi-static map near the current pose of the robot to use as reference for localization instead of the static map. To choose the nearest semi-static map, we use the Mahalanobis distance as proximity measure between the pose of the robot and the poses in the local maps. Figure 3.5 illustrates a situation where the robot has to choose between two neighboring semi-static maps. Relying on the Mahalanobis distance, we can take the uncertainty in the pose estimate into account when selecting an adequate map. We use a kd-tree to store the poses of the local maps to make the search efficient.

Using temporary maps for localization, the weighs of the particles are computed as $w_t = p(z_t \mid x_t, m, d_t) \cdot w_{t-1}$, where d_t corresponds to the nearest semi-static map

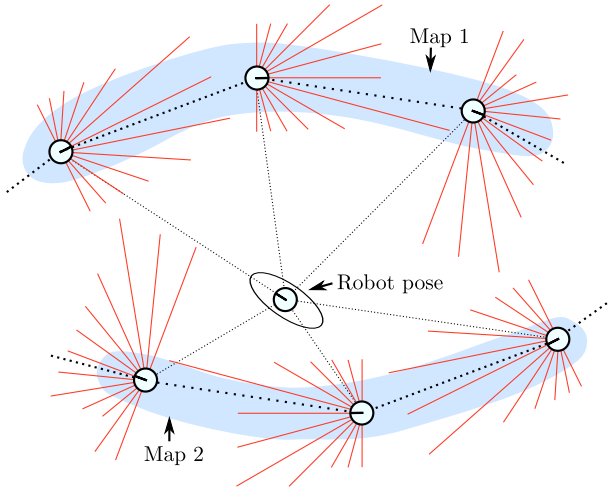


Figure 3.5: Localization using temporary maps. To choose the nearest semi-static map, we use the Mahalanobis distance as proximity measure between the pose of the robot and the poses in the local maps.

and w_{t-1} is the weight of the particle after the last importance weighting step of the localization algorithm. The observation likelihood $p(z_t \mid x_t, m, d_t)$ is computed as

$$p(z_t \mid x_t, m, d_t) = p(z_t \mid x_t, m)^{I(z_t,m)} \cdot p(z_t \mid x_t, d_t)^{I(z_t,d_t)} . \tag{3.7}$$

In the equation above, $I(z_t, m)$ is an indicator function defined as

$$I(z_t, m) = \begin{cases} 1 & \text{if } \bar{e}(z_t, m) < \epsilon_z \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

where $\bar{e}(z_t, m)$ is the average outlier ratio for observation z_t and map m as defined in Equation (3.6). Here ϵ_z represents the threshold at which the observation z_t is considered inconsistent with the map m. Semi-static maps are only created in areas where the observations are inconsistent with the static map of the environment. Thus, Equation (3.7) states that the weights of the particles are computed according to either the static map of the environment m or the closest semi-static map d_t, provided that d_t is consistent with z_t. Note that when no map is consistent with the observations, all particles will be assigned the same weight, and the particle set will evolve exclusively according to the motion model of the robot.

Since our approach is based on a particle filter, the complexity of the algorithm depends mostly on the number of particles used. The construction, including opti-

mization, of a semi-static map is approximately lineal in the number of poses in the map. Adding semi-static maps to the kd-tree and searching for the closest semi-static map is logarithmic in the number of poses of all semi-static maps. However, constructing and adding semi-static maps to the kd-tree is not a frequent operation and searching for the closest semi-static maps takes place only when the observations of the robot are inconsistent with the static map of the environment. More importantly, none of these operations depend on the number of particles. As a result, the overall complexity of the algorithm depends linearly on the number of particles.

To summarize our approach, we assume that a static map of the environment is given. When the observations of the robot are inconsistent with this map we try to find a semi-static map to use as reference for localization instead. The semi-static map is selected based on the distance to the current pose of the robot. If a map is found, we try to localize the robot using the temporary map instead of the reference map of the environment. Temporary maps are eliminated if they are inconsistent with the robot's observation.

3.3 Experimental Evaluation

We implemented and tested our approach using real data gathered with a MobileRobots Powerbot equipped with a SICK LMS laser range finder. The experiments show that by exploiting the observations caused by semi-static objects our method can robustly and accurately estimate the pose of the robot where standard approaches fail.

3.3.1 Localization in Large Open Spaces

To evaluate the robustness and accuracy of our approach we steered the robot through the parking lot of Department of Computer Science at the University of Freiburg and created a map containing only the static elements of the environment. Although there exists approaches for generating static maps in dynamic environments (e.g. [Hähnel *et al.*, 2003b]), this was not the focus of our research so we manually removed the dynamic elements. Furthermore, we removed some areas of the static map to better evaluate the behavior of our approach. Figure 3.6 shows the part of the environment that was used as static map for this experiment.

We evaluated our approach in the task of position tracking and did not consider the problem of global localization. Since our framework is an extension of the MCL approach, global localization is possible as long as enough features of the reference

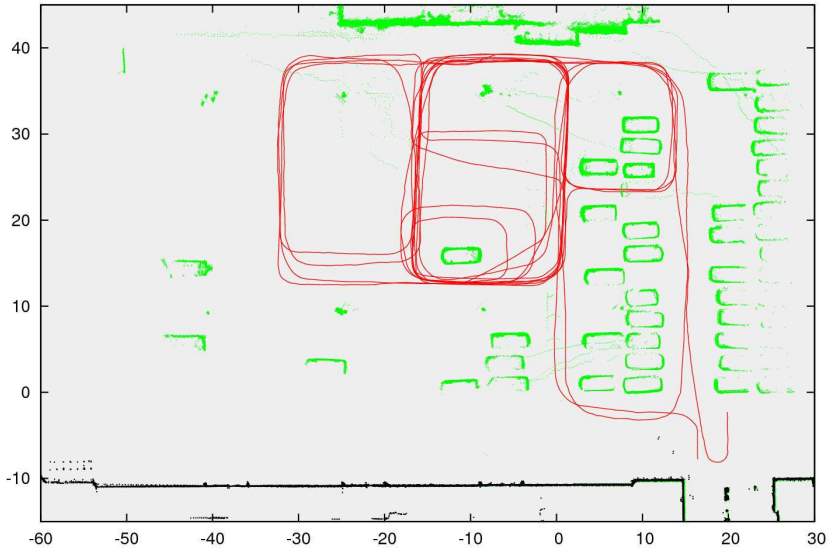


Figure 3.6: Trajectory of the robot obtained during the experiments with our localization approach. The dark colored structures at the bottom correspond to the static parts of the environment used as reference map. Note that by limiting the line-of-sight of the robot to 20 meters the reference map could only be observed sporadically.

map are observed. Figure 3.6 plots the average trajectory of the robot obtained using our localization approach over 10 repetitions of the experiment. In every repetition, 30 particles where used and the maximum range of the laser beams was set to 20 meters. In this way, we reduced the number of observations caused by the reference map. As pose estimate, we used the weighted mean of the particle set. The ground truth map, also shown in Figure 3.6, was computed using a static SLAM approach [Grisetti *et al.*, 2010] considering the full 80 meter depth range of the laser scanner. Note that the reference map was only observed during short time intervals at the beginning and the end of the trajectory. Despite of this, the pose of the robot could be accurately estimated during the whole experiment.

To quantitatively measure the accuracy of our approach, we computed the average error between the estimated poses and the ground truth. We also compared our results against that of the standard MCL approach using the raw odometry of the robot in one case and using an improved odometry based on scan-matching in the other. Figure 3.7 plots the average error and standard deviation of the errors. As can be seen in the figure, our approach only produces a small and relatively constant error along the

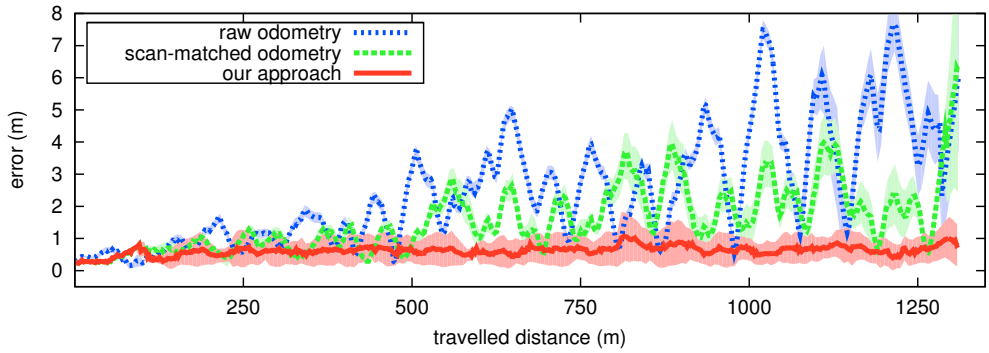


Figure 3.7: Average error and standard deviation of the estimated pose obtained using our approach. We also compared the results against a standard MCL approach using the raw odometry of the robot in one case and using an improved odometry based on scan-matching in the other.

entire trajectory. The standard MCL approach, in contrast, results in a substantially larger error, even when the robot utilized the improved odometry. Thus, the utilization of observations caused by semi-static objects substantially increases the localization capabilities of the robot.

3.3.2 Localization in Non-Static Environments

The goal of this second experiment was to evaluate how our approach handles large changes in the environment. We collected data on two different days on the parking lot so that the configuration of the parked cars would be considerably different. We ran our algorithm on the data of the first day and used the obtained temporary maps as the initial extended map for the data of the second day. The objective of the experiment was to analyze the effect of inconsistent temporary maps on the localization.

Figure 3.8 plots the average error and standard deviation of our approach when using the outdated semi-static maps compared against the error when no semi-static maps were given beforehand. As can be seen, there are no significant differences between both errors. This shows that our approach can correctly identify when a temporary map is not valid anymore and discards it accordingly. As explained in Section 3.2.3, a temporary map is considered invalid if the average outlier ratio of the particle set for that map is above a given threshold. We determined this value empirically and set it to 0.8 in all our experiments. On the one hand, using smaller values sometimes caused maps to be deleted even when the environment had not changed. On the other hand, using

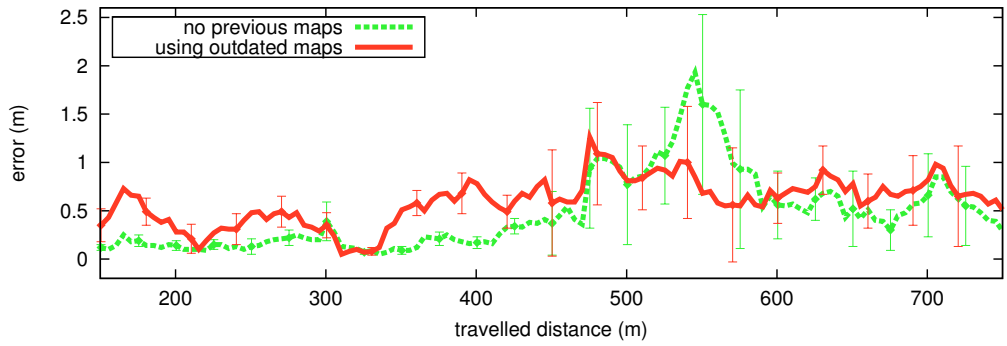


Figure 3.8: Average localization error obtained when using outdated semi-static maps compared against the error when no maps were given beforehand.

larger values made the algorithm overconfident in the available maps which sometimes lead to less accurate results. This explains the slightly higher errors obtained when using the outdated semi-static maps in the first half of the trajectory shown in Figure 3.8.

3.3.3 Standard SLAM in Non-Static Environments

The goal of this experiment was to compare standard SLAM approaches with our approach. We created several artificial maps representing a parking lot in different configurations (see Figure 3.9). Using a simulation environment we switched between the different maps while the robot moved to create the effect of a semi-static environment. For the comparison we considered two state-of-the-art static SLAM techniques: a Rao-Blackwellized Particle Filter (RBPF) [Grisetti *et al.*, 2005] and a graph-based SLAM approach [Grisetti *et al.*, 2010]. To measure the accuracy of the approaches we utilized the error metric described in [Burgard *et al.*, 2009]. The displacements from the initial pose where used to emphasize the overall geometry of the environment.

Figure 3.10 compares the translational error obtained using our approach and the static SLAM techniques. As the number of observations caused by non-static objects increases in the map, it becomes more difficult for the static SLAM approaches to distinguish between inconsistent observations caused by changes in the environment, sensor noise, and localization errors. In particular because changes in the environment are not explicitly considered. This is reflected by the growth in the error as the robot navigates the environment. By relying on an unmodifiable static map and discarding the semi-static maps as they become inconsistent, our approach is robust against changes in the environment as can be seen in the figure by the almost constant error.

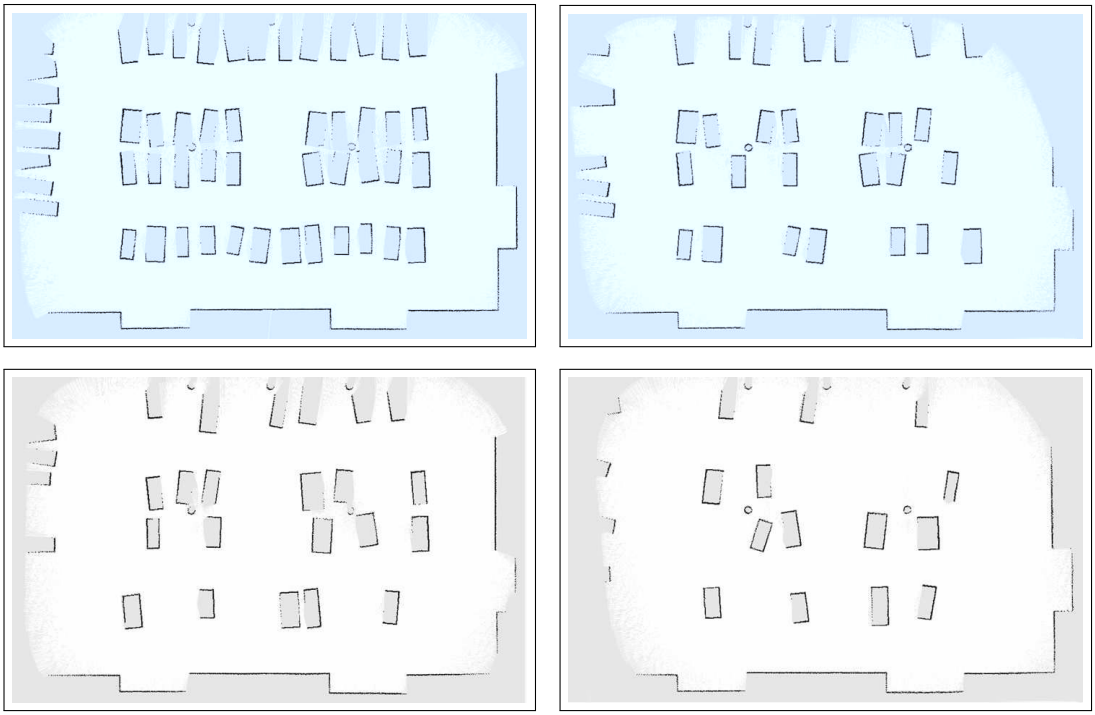


Figure 3.9: Artificial maps representing a parking lot in different configurations used in simulation to create the effect of a semi-static environment.

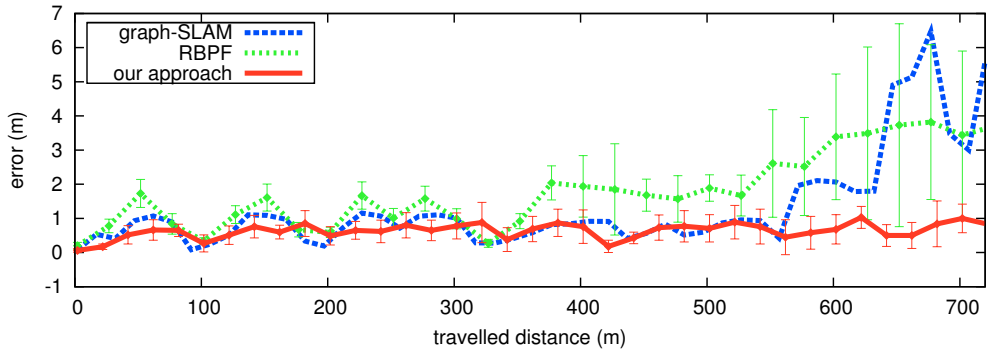


Figure 3.10: Average error and standard deviation of the estimated pose obtained using our approach and two static SLAM techniques: Rao-Blackwellized Particle Filter (RBPF) and graph-based SLAM.

3.4 Related Work

In the past, several authors have studied the problem of mobile robot localization in non-static environments. Fox *et al.* [Fox *et al.*, 1999], for example, use an entropy gain filter to identify the measurements caused by dynamic objects. Burgard *et al.* [Burgard *et al.*, 2000] additionally use a distance filter which selects individual measurements based on the difference between their measured and their expected distance. Montemerlo *et al.* [Montemerlo *et al.*, 2002] propose a method for tracking people while simultaneously localizing the robot which increases the robustness of the robot pose estimation.

The problem of dealing with dynamic objects has also been investigated in the field of simultaneous localization and mapping (SLAM). Wang and Thorpe [Wang and Thorpe, 2002] employ a feature-based heuristic to identify dynamic objects in range measurements and use the filtered result for localizing the robot and building a map at the same time. Hähnel *et al.* [Hähnel *et al.*, 2002] use a probabilistic method for tracking people and filter out the corresponding measurements to improve the map building process. They later extended this approach in [Hähnel *et al.*, 2003b] by considering measurements individually and estimating a posterior about whether it has been generated by an dynamic object. Although these filtering approaches have been shown to be robust in highly dynamic environments, they discard valuable localization information when the changes in the environment occur with a relatively low frequency.

Stachniss and Burgard [Stachniss and Burgard, 2005] approach this problem by estimating typical configurations of dynamic areas in the environment. They show that the integration of this information into a particle filter framework improves the robot pose estimation. Anguelov *et al.* [Anguelov *et al.*, 2002] deal with non-stationary objects representing them using learned geometric models. They apply a hierarchical EM algorithm based on occupancy grid mapping to learn a shape model of the objects and ultimately use this information to correct the mapping process. Andrade-Cetto and Sanfeliu [Andrade-Cetto and Sanfeliu, 2002] describe an approach where landmarks are introduced and removed depending on how often they had been observed. Biber and Duckett [Biber and Duckett, 2005] propose a spatio-temporal map where the environment is represented at multiple time-scales simultaneously. In contrast to mapping approaches, we do not aim at generating a consistent representation of the environment. The local maps constructed by our approach are only temporary and are discarded as soon as inconsistencies are detected.

The idea of using sets of local maps as representation of the environment has been

proposed by many authors in the past. Estrada *et al.* [Estrada *et al.*, 2005], for example, build a graph of local stochastically independent maps and correct the position of the local maps whenever the robot enters an area already visited. Williams *et al.* [Williams *et al.*, 2002] create a local submap of the features around the robot and fuse the local maps regularly with a global map. Gutmann and Konolige [Gutmann and Konolige, 2000] construct locally consistent maps and use them to determine when the robot is entering an area already visited in the global map. All these approaches assume the environment to be static and use the gathered information for constructing globally consistent maps. In contrast, our goal is to improve the localization capabilities of the robot by constructing local maps that temporarily extend the reference map of the environment.

3.5 Conclusions

In this chapter, we presented a localization framework that exploits the measurements caused by semi-static objects to improve the localization capabilities of a mobile robot in dynamic environments. The presented approach constructs local maps using the measurements caused by semi-static objects. These maps temporarily extend the reference map of the environment and are used as fall-back maps whenever the observations of the robot are inconsistent with the reference map. The approach extends the standard MCL approach that only employs a map of the static aspects of the environment to estimate the pose of the robot. We implemented our approach and tested it on data gathered with a real robot. Experimental results demonstrate that by exploiting the observations caused by semi-static objects our approach is capable of robustly and accurately estimating the pose of the robot even in situations in which state-of-the-art approaches fail.

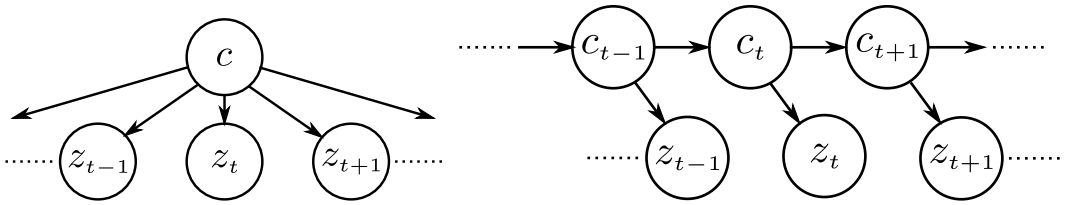
Chapter 4

Grid-Based Models for Dynamic Environments

An accurate model of the environment is essential for many mobile robot navigation tasks. Although the environment generally is dynamic, most existing navigation approaches assume it to be static. They typically build a static map of the environment in an offline phase and then use it without considering potential future changes. There are robust approaches that can handle inconsistencies between the map and the actual measurements. However, a largely inconsistent model can lead to unreliable navigation or even to a complete localization failure.

In Chapter 3 we described a localization framework that uses the measurements caused by semi-static objects to temporarily extend the static map of the environment. The approach, however, relied on a rather simple model of the temporal dependencies between maps at consecutive time steps. In this chapter we consider the problem of modeling a mobile robot's environment taking the dynamics of the environment explicitly into account. We present a probabilistic model that represents the occupancy of the space and characterizes how this occupancy changes over time. The explicit representation of how the occupancy changes in time provides a better understanding of the environment that can be used to improve the robot's navigation performance.

We describe the environment as a spatial grid and use a hidden Markov model (HMM) to represent the belief about the occupancy state and state transition probabilities of each grid cell. Our model, called *dynamic occupancy grid*, is a generalization of a standard occupancy grid. Figure 4.1 illustrates the fundamental difference between these two models: while occupancy grids characterize the state of a cell as static, our representation explicitly models state changes. In addition to the explicit representa-



(a) Occupancy grid (b) Dynamic occupancy grid

Figure 4.1: Bayesian network describing the dependencies between the states of a cell c and observations z in standard and dynamic occupancy grids.

tion of the environment dynamics, the HMM framework provides efficient algorithms for estimating the model parameters. This allows us to learn the dynamics of the environment from observations made by the robot. Furthermore, within the framework we can efficiently estimate the occupancy state of a cell from the observed evidence as it becomes available, making it possible to continuously adapt the representation.

In this chapter, we propose a mapping approach that represents the occupancy of the space and explicitly characterizes how this occupancy changes over time. We describe our model and how the representation can be updated as new observations become available. Furthermore, we present two techniques, one offline and one online, to estimate the state transition probabilities of the model from observed data. We evaluate our approach in simulation and using real-world data. The results demonstrate that our model can represent dynamic environments more accurately than standard occupancy grids. Furthermore, we show how the explicit representation of the environment dynamics can be used to improve the path planning performance of a robot.

4.1 Dynamic Occupancy Grids

Occupancy grids, as they were introduced by Moravec and Elfes [Moravec and Elfes, 1985] (see Chapter 2), are a regular tessellation of the space into a number of rectangular cells. They store in each cell the probability that the corresponding area of the environment is occupied by an obstacle. To avoid a combinatorial explosion of possible grid configurations, the approach assumes that neighboring cells are independent.

Occupancy grids rest on the assumption that the environment is static. As mentioned above, they store for each cell c of an equally spaced grid, the probability $p(c)$ that c is occupied by an obstacle. Thus far, there is no model about how the occupancy changes over time. The approach described in this chapter overcomes this limitation

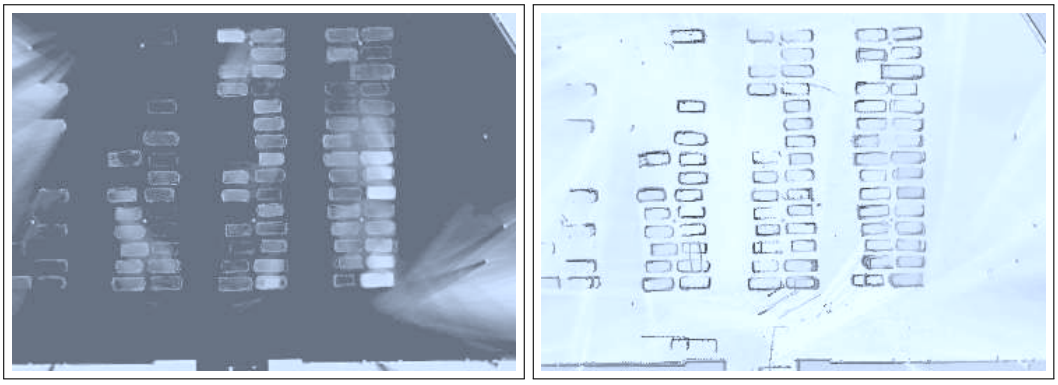


Figure 4.2: State transition probabilities at the parking lot of the Department of Computer Science at the University of Freiburg. The left and right images correspond to the distributions $p(c_t = \textit{free} \mid c_{t-1} = \textit{free})$ and $p(c_t = \textit{occ} \mid c_{t-1} = \textit{occ})$ respectively. The darker the color, the larger the probability for the occupancy to remain unchanged.

by relying on an HMM (see Chapter 2) to explicitly represent both the belief about the occupancy state and state transition probabilities of each grid cell (see Figure 4.1).

An HMM requires the specification of the state transitions, observation probabilities, and initial state distribution. Let c_t be a discrete random variable that represents the occupancy state of a cell c at time t. The initial state distribution $p(c_{t=0})$ specifies the occupancy probability of a cell a the initial time step $t = 0$ prior to any observation.

The state transition model $p(c_t \mid c_{t-1})$ describes how the occupancy state of cell c changes between consecutive time steps. We assume that the changes in the environment are caused by a stationary process, that is, the state transition probabilities are the same for all time steps t. These probabilities are what allows us to explicitly characterize how the occupancy of the space changes over time. Since we are assuming that a cell c is either free (*free*) or occupied (*occ*), the state transition model can be specified using only two transition probabilities, namely $p(c_t = \textit{free} \mid c_{t-1} = \textit{free})$ and $p(c_t = \textit{occ} \mid c_{t-1} = \textit{occ})$. Note that, by assuming a stationary process, these probabilities do not depend on the absolute value of t. Figure 4.2 depicts the transition probabilities for the parking lot of the Department of Computer Science at the University of Freiburg. The darker the color, the larger the probability for the corresponding occupancy to remain unchanged. The figure clearly shows the parking spaces, driving lanes, and static elements such as walls and lampposts as having different dynamics. The "shadows" in the upper left and lower right areas of the maps were mostly caused by maximum range measurements. These are ignored so no information is available for the areas covered by them.

The observation model $p(z \mid c)$ represents the likelihood of the observation z given the state of the cell c. The observations correspond to measurements obtained with a range sensor. In this thesis, we consider only observations obtained with a laser range scanner. The cells in the grid that are covered by a laser beam are determined using a ray-tracing operation. We consider two cases: the beam is not a maximum range measurement and ends up in a cell (a *hit*) or the beam covers a cell without ending in it (a *miss*). Accordingly, the observation model can also be specified using only two probabilities: $p(z = \textit{hit} \mid c = \textit{free})$ and $p(z = \textit{hit} \mid c = \textit{occ})$. We additionally take into account the situation where a cell is not observed at a given time step. This is necessary since the transition model characterizes state changes only for consecutive time steps and not every cell is observed at each time step. Explicitly considering this *no-observation* case allows us to update and estimate the parameters of the model using the HMM framework directly without having to distinguish between observations and no-observations. The concrete observation probability for a *no-observation* is irrelevant as long as the proportion between the other probabilities remains unchanged.

From the discussion above it can be seen that standard occupancy grids are a special case of dynamic occupancy grids where probabilities $p(c_t = \textit{free} \mid c_{t-1} = \textit{free})$ and $p(c_t = \textit{occ} \mid c_{t-1} = \textit{occ})$ are 1 for all cells c.

4.1.1 Occupancy State Update

The update of the occupancy state of the cells in a dynamic occupancy grid follows a Bayesian approach. The goal is to estimate the belief or posterior distribution $p(c_t \mid z_{1:t})$ over the current occupancy state c_t of a cell given all the available evidence $z_{1:t}$ up to time t. The belief update can be formulated as

$$p(c_t \mid z_{1:t}) = \eta\, p(z_t \mid c_t) \sum_{c_{t-1}} p(c_t \mid c_{t-1})\, p(c_{t-1} \mid z_{1:t-1})\,, \tag{4.1}$$

where η is a normalization constant. Exploiting the Markov assumptions in our HMM, this equation is obtained using Bayes' rule with $z_{1:t-1}$ as background knowledge and applying the theorem of total probability on $p(c_t \mid z_{1:t-1})$ conditioning on the state of the cell c_{t-1} at the previous time step $t-1$. Equation (4.1) corresponds to a discrete Bayes filter and describes a recursive approach to estimate the current state of a cell given a current observation and the previous state estimate. Note that the map update for standard occupancy grids is a special case, where the sum in Equation (4.1) is replaced by the posterior $p(c_t \mid z_{1:t-1})$.

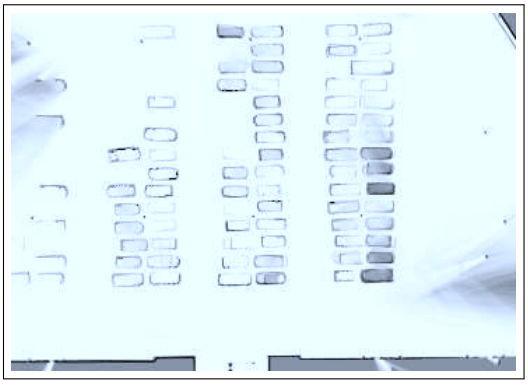


Figure 4.3: Occupancy grid representing the stationary distribution π^s_{occ} for the parking lot of the Department of Computer Science at the University of Freiburg. The state transition probabilities used to compute the stationary distribution correspond to the ones represented in Figure 4.2.

This posterior, corresponds to a prediction of the occupancy state of the cell at time t based on the observations up to time $t-1$. Prediction can be considered as filtering without the processing of evidence. By explicitly considering no-observations as explained in the previous section, the update formula can be used directly to estimate the future state of a cell or estimate the current state of a cell that has not been observed recently.

As the number of time steps for which no observation is available tends to infinity, the occupancy value of a cell converges to a unique stationary distribution π^s. This stationary distribution can be computed for our concrete HMMs as

$$\pi^s_{\mathit{free}} = \frac{p(c_t = \mathit{free} \mid c_{t-1} = \mathit{occ})}{p(c_t = \mathit{occ} \mid c_{t-1} = \mathit{free}) + p(c_t = \mathit{free} \mid c_{t-1} = \mathit{occ})} \tag{4.2}$$

and, accordingly, $\pi^s_{\mathit{occ}} = 1 - \pi^s_{\mathit{free}}$. If a cell is not observed for some time, its occupancy belief will converge to its stationary distribution, regardless of the value after the last observation. Figure 4.3 shows an occupancy grid representing the stationary distribution π^s_{occ} for the parking lot of the Department of Computer Science at the University of Freiburg. The state transition probabilities used to compute the stationary distribution correspond to the ones represented in Figure 4.2.

The time needed for the occupancy value of a cell to converge to its stationary distribution its called the *mixing time*. The concrete definition of mixing time depends on the metric used for measuring the distance between distributions. One simple ap-

proach is to consider the *total variation distance* (see [Levin *et al.*, 2006]) to measure the distance between two distributions. Since our HMMs have only two states, the total variation distance Δ_t between the stationary distribution π^s and the occupancy distribution p_t at time t can be specified as

$$\Delta_t = \left|1 - p(c_t = \mathit{occ} \mid c_{t-1} = \mathit{free}) - p(c_t = \mathit{free} \mid c_{t-1} = \mathit{occ})\right|^t \Delta_0 \,, \tag{4.3}$$

where

$$\Delta_0 = \left|\pi_{\mathit{free}} - \pi^s_{\mathit{free}}\right| . \tag{4.4}$$

is the difference between the initial state π and stationary distribution π^s.

Based on the total variation distance, the mixing time $t_{\mathit{mix}}(\epsilon)$ can then be defined as the smallest t such that Δ_t is less than a given value ϵ. Let

$$\begin{aligned} a &= p(c_t = \mathit{occ} \mid c_{t-1} = \mathit{free}) \\ b &= p(c_t = \mathit{free} \mid c_{t-1} = \mathit{occ}) \,. \end{aligned}$$

The mixing time $t_{\mathit{mix}}(\epsilon)$ for a given π_{free} can be analytically computed as

$$t_{\mathit{mix}}(\epsilon) = \left\lceil \frac{\ln(\epsilon/\Delta_0)}{\ln(|1-a-b|)} \right\rceil , \tag{4.5}$$

where $\lceil n \rceil$ is the ceiling operator that returns the smallest integer larger than n. Equation (4.5) can be derived from Equation (4.3) by straight forward algebraic operations. This computation is only valid when $\Delta_0 > 0$, for the opposite case, $t_{\mathit{mix}}(\epsilon)$ is trivially 0.

Being able to compute the mixing time of a cell has important practical implications, since it renders unnecessary the computation of predicted occupancy values for times steps beyond the mixing time.

4.1.2 Parameter Estimation

As mentioned above, an HMM is characterized by the state transition probabilities, the observation model, and the initial state probabilities. We assume that the observation model only depends on the sensor. Therefore it can be specified beforehand and is the same for each HMM. We estimate the remaining parameters using observations that are assumed to originate to the environment that is to be represented.

One of the most popular approaches for estimating the parameters of an HMM

is an instance of the expectation-maximization (EM) algorithm (see Chapter 2). The basic idea is to iteratively estimate the model parameters using the observations and the parameters estimated in the previous iteration until the values converge. Let $\hat{\theta}^n$ represent the parameters estimated at the n-th iteration. The EM algorithm results in the following re-estimation formula for the transition model of cell c

$$\hat{p}(c_t = i \mid c_{t-1} = j)^{n+1} = \frac{\sum_{\tau=1}^{T} p(c_{\tau-1} = i, c_\tau = j \mid z_{1:T}, \hat{\theta}^n)}{\sum_{\tau=1}^{T} p(c_{\tau-1} = i \mid z_{1:T}, \hat{\theta}^n)}\,, \tag{4.6}$$

where $i, j \in \{\textit{free}, \textit{occ}\}$ and T is the length of the observation sequence used for estimating the parameters. Note that the probabilities on the right-hand side are conditioned on the observation sequence $z_{1:T}$ and the previous parameter estimates $\hat{\theta}^n$. These probabilities can be efficiently computed using the *forward-backward* procedure (see [Rabiner, 1989]).

This, however, is an offline approach that requires storing the complete observation sequence for each cell. An online version of the algorithm was derived by Mongillo and Deneve [Mongillo and Deneve, 2008]. To calculate the transition probabilities in Equation (4.6), this algorithm only needs to store the sufficient statistics

$$\phi_{ijh}(t; \hat{\theta}) = \frac{1}{t} \sum_{\tau=1}^{t} \delta(z_\tau, h)\, p(c_{\tau-1} = i, c_\tau = j \mid z_{1:t}, \hat{\theta})\,, \tag{4.7}$$

where $i, j \in \{\textit{free}, \textit{occ}\}$, $h \in \{\textit{free}, \textit{occ}, \textit{no-observation}\}$, and $\delta(z_\tau, h) = 1$ if $z_\tau = h$ and 0 otherwise. Dropping the dependence on $\hat{\theta}$, only 16 values have to be stored for each cell. The algorithm uses ϕ_{ijh} instead of the probabilities computed with the forward-backward procedure to estimate the transition model. Therefore it implements only a partial expectation step, while the maximization step remains exact.

Besides being an online approach with small storage requirements, the prefactor $1/t$ in Equation (4.7) allows the algorithm to handle non-stationary environment's dynamics. Additionally, the algorithm updates with each observation the occupancy state according to Equation (4.1). These properties make the online version of the EM algorithm an attractive alternative for systems operating over extended periods of time.

Figure 4.4: Google Map image of the parking lot of the Department of Computer Science at the University of Freiburg.

4.2 Experimental Evaluation

We implemented our proposed model and tested it in simulation and using data obtained with a real robot. The goal of the experiments was to evaluate the quality and usefulness of our proposed representation.

4.2.1 Accuracy of the Representation

In a first experiment we evaluated the accuracy of our proposed representation of the environment. We steered a MobileRobots Powerbot equipped with a SICK LMS laser range finder through the parking lot of the Department of Computer Science at the University of Freiburg (see Figure 4.4). We performed a run every full hour from 7am until 6pm during one day. The range data obtained from the twelve runs (data sets d_1 through d_{12}) corresponded to twelve different configurations of the parked cars, including an almost empty parking lot (data set d_1) and a relatively occupied one (data set d_{10}). We used a SLAM approach [Grisetti *et al.*, 2005] to correct the odometry of the robot and obtain a good estimate of its pose. Range measurements were sampled at about 1 hertz, and the trajectory and velocity of the robot during each run were approximately the same to try to avoid a bias in the complete data set.

Figure 4.5 shows a qualitative comparison between dynamic and standard occupancy grids for the parking lot data set. The online EM approach was used to build the dynamic occupancy grid. We assumed that the parking lot did not change considerably during a single run and used the occupancy grids obtained from every data set with the above-mentioned SLAM approach as ground truth. In the figure, the maps on the left

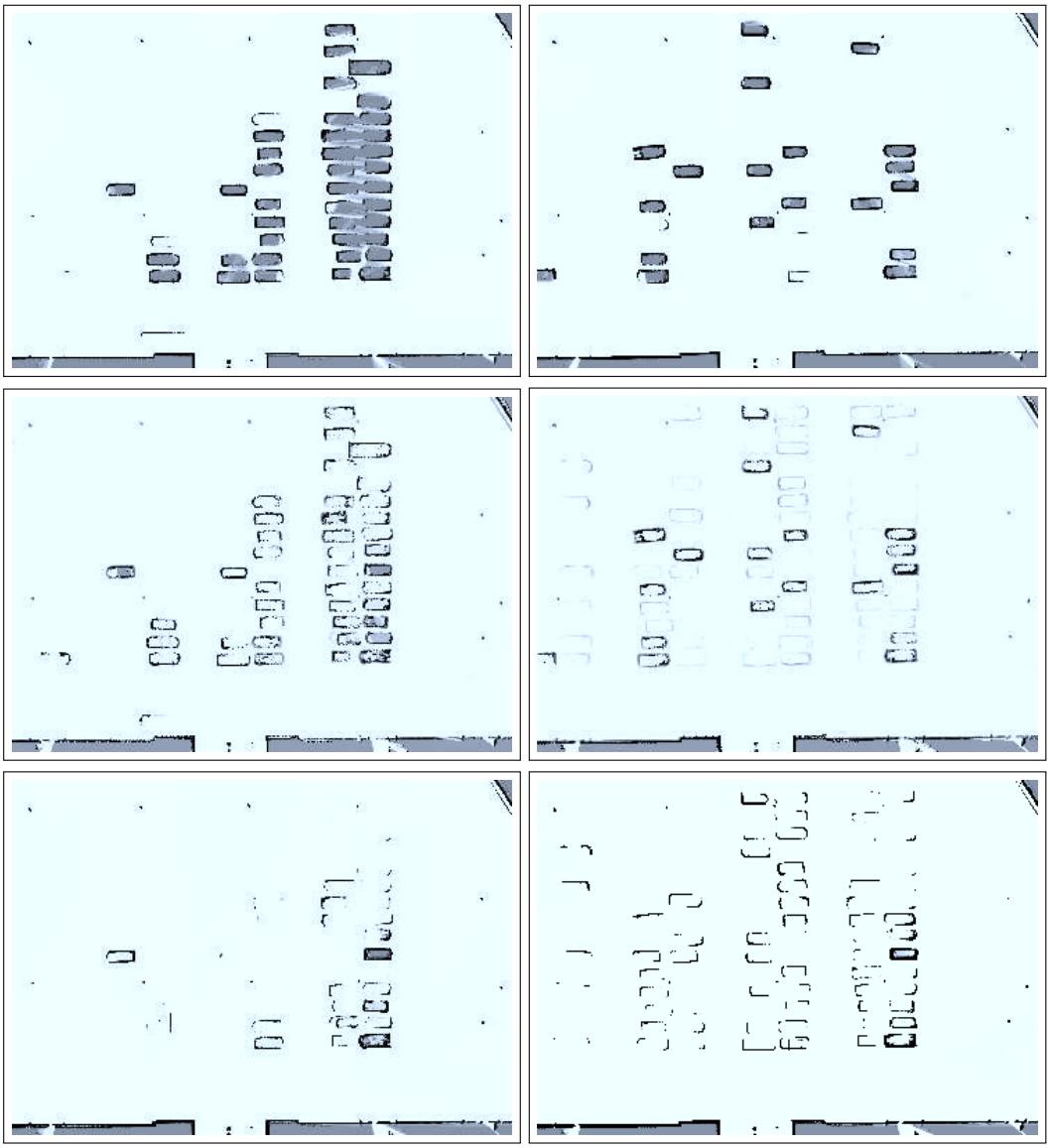


Figure 4.5: Comparison between dynamic and standard occupancy grids. Shown are the ground truth (top), dynamic occupancy grid (middle), and standard occupancy grid (bottom) maps at two different points in time.

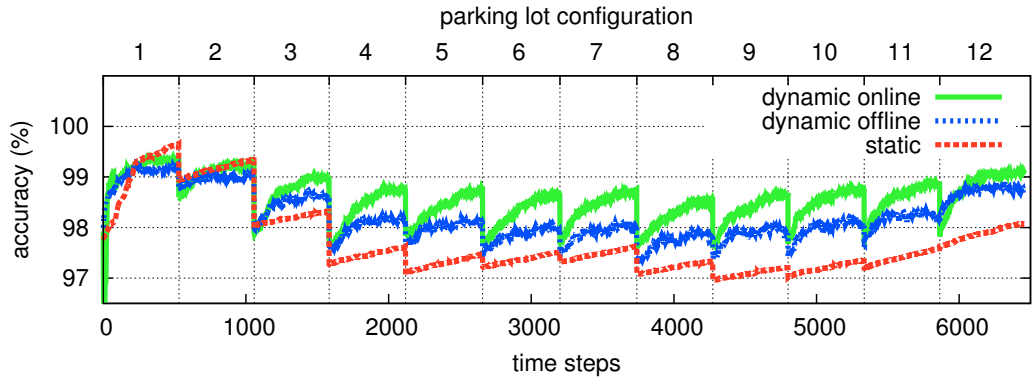


Figure 4.6: Accuracy over time of standard and dynamic occupancy grids for the parking lot data. Both online and offline parameter estimation approaches were evaluated.

column show the grids after the third run, that is, after integrating data sets d_1 through d_3. The maps on the right show the grids at the end of the last run, after integrating data sets d_1 through d_{12}. As can be seen, the dynamic occupancy grid readily adapts to the changes in the parking lot. Thus it constitutes a better representation of the environment at any point in time. Additionally, dynamic occupancy grids provide information about the probable occupancy of areas that have not been recently observed. This appears in the grids in the figure (specially the right column) as light gray areas in the places where the cars most frequently park.

To quantitatively evaluate the accuracy of our representation we computed its accuracy with respect to the ground truth maps. In this context, accuracy is defined as the number of cells correctly classified divided by the total number of classified cells. A cell c was classified as *occupied* if $p(c) > 0.5$ and as *free* if $p(c) < 0.5$. The remaining cells were not taken into account. Figure 4.6 compares the accuracy of a standard occupancy grid (*static*) against that of the dynamic occupancy grid whose parameters where estimated online (*dynamic online*). We additionally consider the case when the parameters where estimated offline (*dynamic offline*) — depicted in Figure 4.2. Figure 4.6 plots the accuracy of the grids over time for the parking lot data. After each configuration change, the accuracy of the dynamic occupancy grids quickly starts to increase as the map adapts to the new configuration. Standard occupancy grids adapt relatively quickly at first, but their adaptability decreases with the number of observations already integrated into the map.

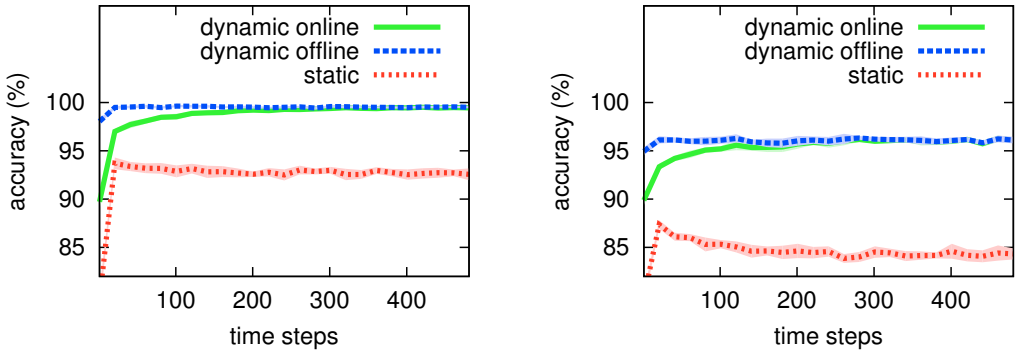


Figure 4.7: Accuracy of standard and dynamic occupancy grids for different configurations of dynamic cells in a simulated grid map. Left: $5\,\%$ dynamic cells and $5\,\%$ state change probability. Right: $25\,\%$ dynamic cells and $25\,\%$ state change probability

4.2.2 Effects of the Environment's Dynamics

The accuracy of dynamic occupancy grids and their advantage over standard occupancy grids strongly depends on the environment's dynamics. In the parking lot environment of the previous experiment, only a small number of cells were dynamic ($\sim 3\,\%$) and only few changes took place. This explains why the standard occupancy grids in Figure 4.5 corresponding respectively to configurations 3 and 12 in Figure 4.6 have high accuracy values even though they are evidently inaccurate.

The goal of this experiment was to evaluate the accuracy of our proposed representation for different environment dynamics. In the context of occupancy grids, the dynamics of the environment are characterized by the number of dynamic cells and their state change probabilities. We used a 50×50 grid map and changed the fraction of dynamic cells and state change probabilities to generate artificial data for different environment dynamics. We estimated the state transition probabilities of the dynamic occupancy grid using both the online and offline approaches and compared the resulting model accuracy against that of a standard occupancy grid for different data sets. Figure 4.7 shows the results for two different settings: one relatively static and another more dynamic. The curves correspond to the mean and standard deviation for 10 repetitions of the experiment. As can be seen in the figure, dynamic occupancy grids represent the environment more accurately than standard occupancy grids. Nevertheless, the more static the environment, the smaller the difference between the accuracies. This experiment shows that even for moderately dynamic environments dynamic occupancy grids outperform standard occupancy grids.

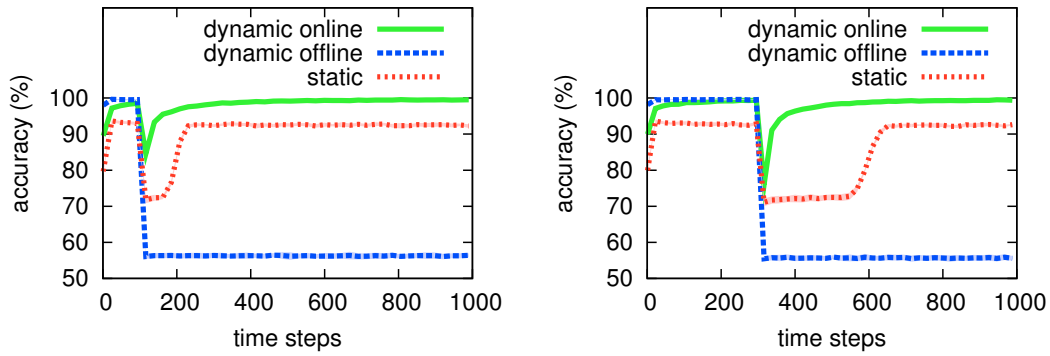


Figure 4.8: Accuracy of the parameters obtained with the online and offline approaches when the environment's dynamics change. The left and right plots correspond to changes after 100 and 300 time steps respectively.

4.2.3 Parameter Estimation

As can be seen in Figure 4.7, the offline approach produces more accurate results at first, but the difference between the results of the offline and online approaches decreases over time. This suggests that, regarding the accuracy of the representation over time, both parameter estimation techniques are comparable.

Although the offline approach produces good results from the beginning, it requires storing all observations for each cell in the grid for an a priori training phase. This is a considerable disadvantage since it limits the amount of data that can be used for training, the resolution of the grid, or the size of the environment that can be represented. Furthermore, being an offline approach, once the parameters have been estimated, they remain fixed. This makes the offline approach inappropriate for environments where the assumption that the environment dynamics are stationary does not hold. In contrast, the online approach continually adapts its parameters as new observations become available. Figure 4.8 illustrates the effects on the accuracy of the model parameters obtained with the two approaches for $5\,\%$ dynamic cells and $5\,\%$ state change probability (left plot in Figure 4.7) in the case that the environment dynamics change. For the experiment, the change consisted in selecting a new set of dynamic and static cells. The number of dynamic cells and their state change probabilities remained the same, but we obtained similar results when these parameters where changed as well. As can be seen in Figure 4.8, using the online approach, the accuracy of the model quickly returns to its value before the change. This is the result of the model parameters adapting to the new environment dynamics. The accuracy of the model whose parameters

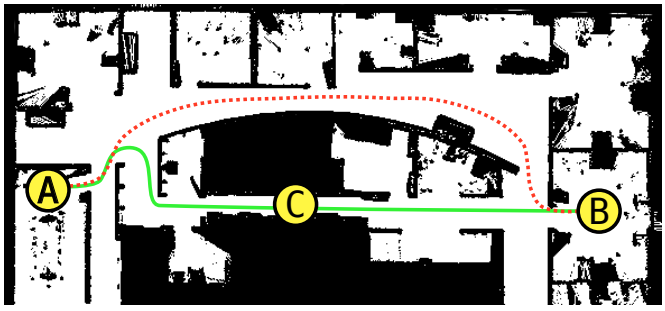


Figure 4.9: Experimental setup for the path planning experiment. The task consists in navigating between A and B. At C we added a virtual door that changed its state over time.

where estimated offline drops when the dynamics change, and remains low. We also evaluated the behavior of standard occupancy grids. As expected, the number of observations needed by a standard occupancy grid to correctly represent the occupancy of the new static cells is approximately the same as the number of previous observations. Note that the dynamic cells remain inaccurately represented.

4.2.4 Path Planning Using Dynamic Occupancy Grids

In the previous experiments we showed that dynamic occupancy grids readily adapt to changes in the environment. The goal of this experiment was to show that this adaptability can be used to improve the path planning performance of a robot. The experiment was performed in simulation within the environment shown in Figure 4.9 corresponding to part of the Intel Research Lab in Seattle (Radish) [Howard and Roy, 2003]. The task of the robot consisted in navigating between rooms A and B. We added a virtual door at position C along the shortest path between A and B. The state of the door changed each time step with a probability of 0.001. To generate the a priori map needed for path planning and obtain data for estimating the parameters of the dynamic occupancy grid, we steered the robot through the relevant parts of the environment in an offline phase.

We then performed 20 repetitions of the experiment. In every repetition, the robot executed 20 runs from one room to the other. The A^* algorithm was used for path planning and re-planning was performed at every time step. The cost of a path was computed as the sum of the traversal costs for each cell in the path. The traversal cost was set to 1 for *free* cells and infinity for *occupied* cells. The cells in the grid were classified as described in the first experiment.

values in %	*static*	*dynamic online*	*dynamic offline*
short	11.25 (±16.35)	25.50 (±22.24)	40.75 (±20.15)
long	35.75 (±14.80)	27.00 (±18.38)	15.00 (±9.03)
indirect long	4.75 (±1.12)	19.50 (±12.24)	30.50 (±8.72)
unnecessary indirect long	0.25 (±1.12)	1.50 (±2.86)	2.75 (±3.02)
unnecessary direct long	48.00 (±16.89)	26.50 (±21.34)	11.00 (±5.28)
accuracy	47.00 (±16.89)	52.50 (±16.10)	55.75 (±13.21)

Table 4.1: Occurrences of the different trajectory types for different occupancy grids and parameter estimation approaches during path planning.

values in %	*random*	*optimal*
short	24.25 (±10.17)	40.50 (±19.99)
long	24.25 (±13.70)	17.25 (±9.39)
indirect long	23.50 (±9.33)	27.75 (±9.24)
unnecessary indirect long	1.50 (±2.86)	2.25 (±3.02)
unnecessary direct long	26.50 (±13.19)	12.25 (±6.38)
accuracy	48.50 (±10.14)	57.75 (±14.09)

Table 4.2: Occurrences of the different trajectory types for two path planning policies: random and optimal.

Since numeric performance measures, like traveled distance or execution time, largely depend on the particular environment used for the experiment, we opted for a more qualitative evaluation and classified the trajectories followed by the robot into five types:

- ***short***: the robot followed the shortest path.
- ***long***: the robot followed the longer path. This was the optimal choice since the door would not have been open for the robot to pass.
- ***indirect long***: the robot tried to follow the shortest path first, found the door closed, turned around, and finally followed the longer path. In this case, following the longer path from the beginning would have been the optimal choice.
- ***unnecessary indirect long***: as in the previous case, the robot tried to follow the shortest path first, found the door closed, turned around, and finally followed the longer path. However, continuing to follow the shortest path would have been the optimal choice since the door would have opened for the robot to pass.
- ***unnecessary direct long***: the robot followed the longer path. No attempt was made to follow the shortest path. In this case, following the shortest path from the beginning would have been the optimal choice.

The values in Table 4.1 correspond to the occurrences (average and standard deviation) of the different trajectory types during the 20 repetitions of the experiment. We compared the path planning performance when using a standard occupancy grid (*static*), a dynamic occupancy grid whose parameters where estimated online (*dynamic online*), and a dynamic occupancy grid whose parameters where estimated offline (*dynamic offline*). The number of occurrences of *short* and *long* trajectories in the table indicate that the information about the state change probability of the door, represented in the dynamic occupancy grid, leads to better path planning performances. Once the door is represented as closed in a standard occupancy grid, the robot never attempts to follow the shortest path again, which, in turn, prevents the robot from updating the cells corresponding to the door. This can be seen in the table by the small number of *short*, *indirect long*, and *unnecessary indirect long* trajectories and explains the large number of *long* trajectories. Using dynamic occupancy grids, on the other hand, the state of the (unobserved) door in the map changes over time according to the learned state transition probabilities. Whenever the cells corresponding to the door are classified as free, the robot attempts to follow the shortest path.

We additionally implemented two baseline path planning policies for comparison (see Table 4.2). In the first (*random*) the robot, when in A or B, randomly chooses between the two possible paths and never replans while on the way. In the second (*optimal*) the robot has perfect knowledge about the state change probability of the door and based on its internal belief about the state of the door chooses the expected optimal path. The percentage of runs in which the optimal path was followed (*accuracy*) by this robot is an empirical upper bound for the accuracy achievable in our experimental setup. One sided t-tests show that the accuracy for *dynamic offline* was significantly higher than those for *static* and *random* on a 5% level. Between the accuracies for *dynamic offline* and *optimal* we could not find a significant difference.

4.3 Related Work

Previous work on mapping dynamic environments can be divided into two groups: approaches that filter out sensor measurements caused by dynamic elements and approaches that explicitly model aspects of the environment dynamics. Filtering out sensor measurements is based on probabilistic sensor models that identify the measurements which are inconsistent with a reference model of the environment. Fox *et al.* [Fox *et al.*, 1999], for example, use an entropy gain filter. Burgard *et al.* [Burgard *et al.*, 2000] propose a distance filter based on the expected distance of a measurement. Hähnel *et al.* [Hähnel *et al.*, 2003b] combine the EM algorithm and a sensor model that considers dynamic objects to obtain accurate maps. In contrast to these approaches, our work explicitly represents the dynamics of the environment in the environment's model itself instead of relying on sensor models to represent them.

To model the dynamics of the environment, some authors have proposed augmented representations of the environment which explicitly represent dynamic objects. The approach of Anguelov *et al.* [Anguelov *et al.*, 2002], for example, computes shape models of non-stationary objects. They create maps at different points in time and compare those maps using an EM-based algorithm to identify the parts of the environment that change over time. Petrovskaya and Ng [Petrovskaya and Ng, 2007] extend occupancy grid maps with parameterized models of dynamic objects and apply a Rao-Blackwellized particle filter to estimate the pose of the robot and the state of the dynamic objects. The above-mentioned approaches are based on the identification and modeling of dynamic objects in the environment. Our approach, in contrast, does not depend on high level object models and considers only the occupancy of the space at a lower level of abstraction. It is similar to the spatial affordance maps proposed by Lu-

ber *et al.* [Luber *et al.*, 2009]. They represent space-dependent occurrences of relevant people activity events in the context of people tracking using Poisson processes. The fundamental difference to our model is that we consider the occupancy of the space and not people tracking events. For this reason our approach relies on HMMs that are better suited than Poisson processes to describe the behavior of the occupancy in a cell.

The problem of modeling the space occupancy in dynamic environments at a low level of abstraction has already been addressed in the past. Wolf and Sukhatme [Wolf and Sukhatme, 2005], for example, propose a model that maintains two separate occupancy grids, one for the static and the other for the dynamic parts of the environment. Brechtel *et al.* [Brechtel *et al.*, 2010] describe a grid-based representation that in addition to the occupancy state of the cells, also stores a velocity vector interpreted as the velocity of the object that occupies the cell. These approaches, however, are rather focused on the problem of tracking dynamic objects than on representing the environment's dynamics. Biber and Duckett [Biber and Duckett, 2005] propose a model that represents the environment on multiple timescales simultaneously. For each timescale a separate sample-based representation is maintained and updated according to an associated timescale parameter. Konolige and Bowman [Konolige and Bowman, 2009] describe a vision-based system where camera views are grouped into clusters that represent different persistent configurations of the environment. Changes in the environment are handled by deleting views based on a least-recently-used principle. The fundamental difference between these previous approaches and ours is that, besides being able to continuously adapt to changes over time, our model provides an explicit characterization of the dynamics of the environment.

4.4 Conclusions

In this chapter we introduced a novel approach to occupancy grid mapping that explicitly represents how the occupancy of individual cells changes over time. Our model is a generalization of standard occupancy grids. It applies HMMs to update the belief about the occupancy state of each cell according to the dynamics of the environment. We described how our maps can be updated as new observations become available. We furthermore introduced an offline and an online technique to estimate the parameters of the model from observed data. We evaluated our approach in simulation and using real-world data. The results demonstrate that our model can represent dynamic environments more accurately than standard occupancy grids. We also demonstrated that using our model can improve the path planning performance of a robot.

Chapter 5

Improving Robot Localization Using Artificial Landmarks

For reliable navigation a mobile robot needs to be able to determine its pose in the environment and accurately track it over time. Although there exist many approaches that have been successfully applied to the localization task, dynamic objects can cause perception ambiguities that make these approaches more likely to fail and, in the worst case, prevent the pose of the robot from being uniquely determinable at all. In the context of localization, environments are considered ambiguous if they prevent different robot poses from being distinguished based on the sensor data. Figure 5.1 illustrates such a problem. It shows a typical sensor measurement obtained using a laser range scanner together with the corresponding observation likelihood. Dark colored areas correspond to high likelihood poses. As can be seen in the figure, several poses, in addition to the pose from which the scan was taken, have a high observation likelihood.

Navigation in dynamic environments, and environments that are structurally symmetrical or have only few recognizable features, can be improved by attaching landmarks to static parts in the environment. In this chapter, we focus on the problem of utilizing artificial landmarks to reduce the ambiguity in the environment. We address the problem of finding a configuration of indistinguishable landmarks that, when placed in the environment, increase the robustness in the localization of the robot. The basic idea behind our approach is that by reducing the overall ambiguity in the environment, the localization performance of the robot can be improved. We first introduce a measure for how distinguishable or unique a pose is. We then present an approach for selecting a configuration of landmarks that incrementally selects landmark locations, greedily maximizing the average uniqueness in the environment.

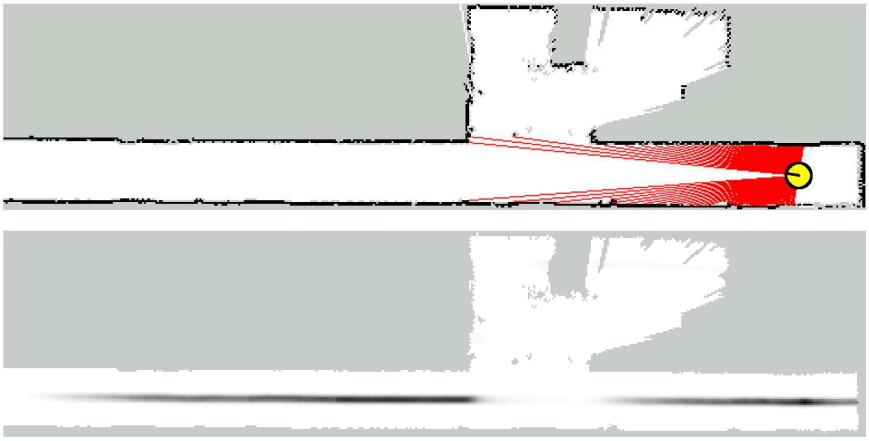


Figure 5.1: Ambiguous environments make localization approaches more likely to fail. The image shows a laser range scan (top) and the x-y-projection (maximizing over the orientation) of the corresponding observation likelihood over the pose space (bottom). The darker the color, the larger the observation likelihood at the corresponding pose. As can be seen there are many poses with an associated high observation likelihood.

The contribution of our proposed approach is two-fold: First, we present a practical approach to landmark placement that aims at improving the localization performance of the robot. Our approach provides us with both the number and location of landmarks to be placed in the environment. We consider indistinguishable landmarks which makes our approach attractive from a practical point of view since no landmark coding and complex identification system is required. As a second contribution we introduce a measure for the uniqueness of a robot pose based on sensor data. We seek to improve the localization performance by maximizing the average uniqueness in the environment. Furthermore, we describe a concrete instantiation of the landmark placement problem for grid-based representations of the environment and show experimentally that our approach improves the localization performance of the robot and outperforms other landmark selection approaches.

5.1 Pose Uniqueness

The pose x of a robot restricted to planar environments is defined by its two location coordinates r^x and r^y in the plane and its orientation r^θ in some global coordinate system. Since the robot cannot directly measure its pose but has to infer it from the observations obtained with its sensors, the uniqueness of a pose is based on these

observations. Intuitively, the uniqueness of a pose indicates how distinguishable the pose is — as observed by the the robot — from all other poses in the state space. Let us assume that the robot is equipped with a perfect sensor that makes, for a pose x and map m, a deterministic observation $z(x, m)$. Then we can define the *uniqueness* of a pose x given a map m as

$$\mathcal{U}_{\text{perfect}}(x, m) = \frac{1}{\int_{\tilde{x} \in \mathcal{X}} \delta_{\tilde{x},m}(x) \, \mathrm{d}\tilde{x}} \,, \tag{5.1}$$

where $\mathcal{X}$ represents the state space, and $\delta_{\tilde{x},m}(x)$ is 1 if $z(\tilde{x}, m) = z(x, m)$ and 0 otherwise. The denominator in Equation (5.1) simply counts the number of poses in the state space where the robot makes the same observation as in pose x. Clearly, the larger the count, the less unique the pose is. For a maximally unique pose x it holds that $\delta_{\tilde{x},m}(x) = 0$ for all $\tilde{x} \in \mathcal{X} \setminus \{x\}$. A minimally unique pose x, on the other hand, is one for which $\delta_{\tilde{x},m}(x) = 1$ for all $\tilde{x} \in \mathcal{X}$.

Since our sensor is noisy, we have to replace the deterministic function $\delta_{\tilde{x},m}(x)$ in Equation (5.1) by the likelihood of observing, at pose $\tilde{x}$, the observation z^x made at x, i.e., $p(z^x \mid \tilde{x}, m)$. Furthermore, as we don't know which measurement z^x we will obtain at pose x, we have to integrate over all potential measurements

$$\mathcal{U}_{\text{exp}}(x, m) = \int_z \frac{1}{\int_{\tilde{x} \in \mathcal{X}} p(z \mid \tilde{x}, m) \, \mathrm{d}\tilde{x}} \, p(z \mid x, m) \, \mathrm{d}z \,. \tag{5.2}$$

Since integrating over all measurements is not feasible in practice, we approximate the outer integral in $\mathcal{U}_{\text{exp}}(x, m)$ by the maximum likelihood value of the distribution $p(z \mid x, m)$ and obtain

$$\mathcal{U}(x, m) = \frac{1}{\int_{\tilde{x} \in \mathcal{X}} p(z^{x*} \mid \tilde{x}, m) \, \mathrm{d}\tilde{x}} \,, \tag{5.3}$$

where $z^{x*} = \operatorname{argmax}_z p(z \mid x, m)$ corresponds to the most likely observation at pose x given the map m. The accuracy of this approximation depends on the distribution $p(z \mid x, m)$. If $p(z \mid x, m)$ is the Dirac density, the approximation is exact. In general, the accuracy depends on how much z^{x*} dominates the outer integral.

A highly unique pose x is typically associated to a peaked distribution $p(z \mid \tilde{x}, m)$ that is at its maximum when $\tilde{x} = x$. On the other hand, a flat distribution typically corresponds to an ambiguous environment where, for all the poses $\tilde{x}$ in the state space, $p(z \mid \tilde{x}, m)$ has almost the same value.

An alternative derivation for Equation (5.3) is obtained by considering the expected value of the observation probability at x with respect to the observation probability at $\tilde{x}$

$$E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big] = \int_z p(z \mid x,m)\, p(z \mid \tilde{x},m)\, \mathrm{d}z\,. \tag{5.4}$$

This expectation is the average of the observation probability at x weighted by the observation probability at $\tilde{x}$. It can be interpreted as a measure of the "match" of the distribution $p(z \mid x,m)$ to $p(z \mid \tilde{x},m)$. Integrating over all poses in the state space, we can define the uniqueness $\mathcal{U}(x,m)$ of a pose x as

$$\begin{aligned}\mathcal{U}(x,m) &= \frac{1}{\int_{\tilde{x}\in\mathcal{X}} E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big]\,\mathrm{d}\tilde{x}} \\ &= \frac{1}{\int_{\tilde{x}\in\mathcal{X}}\int_z p(z \mid x,m)\, p(z \mid \tilde{x},m)\,\mathrm{d}z\,\mathrm{d}\tilde{x}}\,.\end{aligned} \tag{5.5}$$

For a maximally unique pose x it holds that

$$E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big] = 0$$

for all $\tilde{x} \in \mathcal{X} \setminus \{x\}$. A minimally unique pose x, on the other hand, is one for which

$$E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big] = E_{p(z|\tilde{x}',m)}\big[p(z \mid x,m)\big]$$

for all $\tilde{x}, \tilde{x}' \in \mathcal{X}$.

Since integrating over all measurements in Equation (5.5) is not feasible in practice, we approximate $E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big]$ by the maximum likelihood value of $p(z \mid x,m)$

$$E_{p(z|\tilde{x},m)}\big[p(z \mid x,m)\big] \approx p(z^{x*} \mid \tilde{x},m)\,, \tag{5.6}$$

and obtain Equation (5.3).

5.2 Landmark Placement

Given a set $\mathcal{V}$ of N candidate landmarks, the general landmark placement problem consists of finding a configuration $m \subseteq \mathcal{V}$ of landmarks that maximizes a given target function. There exist many possible aspects to consider when specifying the target function, like the number of selected landmarks and area covered, for example. The

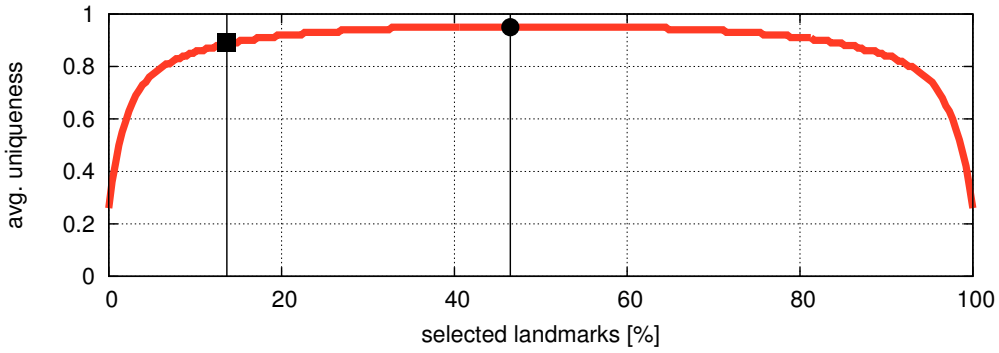


Figure 5.2: Normalized average uniqueness as a function of the number of selected landmarks. The black dot (center) marks the point where the average uniqueness reaches its maximum. The black square (left) indicates the average uniqueness for the number of landmarks selected by our approach.

target function we consider is the average uniqueness value in the environment. Concretely, we look for the configuration m^* so that

$$m^* = \underset{m \subseteq \mathcal{V}}{\operatorname{argmax}} \left(\frac{1}{||\mathcal{X}||} \int_{x \in \mathcal{X}} \mathcal{U}(x, m) \, \mathrm{d}x \right), \tag{5.7}$$

where $||\mathcal{X}||$ represents the size or volume of the state space. By maximizing the average uniqueness in the environment we seek to improve the localization performance of the robot.

The combinatorial nature of the problem makes the enumeration of all possible solutions for finding the optimal one intractable. However, an approximate solution to Equation (5.7) can be efficiently computed in an incremental fashion by successively selecting the landmark that maximizes the average uniqueness until no further improvement is possible. The main disadvantage of this approach is that it selects an unnecessarily large number of landmarks. In practical experiments we found that approximately 50% of the candidate landmarks are selected before no further improvement is possible. Figure 5.2 shows the typical behavior of the average uniqueness as a function of the number of selected landmarks. As can be seen, the average uniqueness reaches its maximum when approximately half of the candidate landmarks are selected. Adding further landmarks provides no additional improvement and, as a matter of fact, the average uniqueness starts to decrease as further landmarks are selected. In order to determine the number of landmarks to select, we use a heuristic approach

Algorithm 5.1 Incremental landmark placement

Input: Set $\mathcal{V}$ of N candidate landmarks

1: $m^* = \emptyset$
2: **while** $\mathcal{V} \neq \emptyset$ **do**
3: $\quad l' = \text{argmax}_l \overline{\mathcal{U}}(x, \{l\} \cup m^*)$
4: $\quad \bar{u}' = \overline{\mathcal{U}}(x, \{l'\} \cup m^*)$
5: $\quad$ **if** $\nabla \bar{u}' > \epsilon$ **then**
6: $\quad\quad m^* = \{l'\} \cup m^*$
7: $\quad\quad \mathcal{V} = \mathcal{V} \setminus \{l'\}$
8: $\quad$ **else**
9: $\quad\quad$ **return** m^*
10: $\quad$ **end if**
11: **end while**

based on the gradient of the average uniqueness. Our landmark placement algorithm terminates whenever the gradient drops bellow some specified threshold. The larger the value for the threshold, the smaller the number of selected landmarks. The average uniqueness is normalized using an instance specific upper bound in order to use the same threshold for different instances of the problem. Assuming a finite and discrete state space, the upper bound for the average uniqueness is given by

$$\overline{\mathcal{U}}(m) = \frac{1}{\min_x p(z^{x*} \mid x, m = \emptyset)} \tag{5.8}$$

and can be determined before selecting the first landmark, i.e. when $m = \emptyset$.

The approach proposed in this work is specified in Algorithm 5.1. Line 3 computes the landmark l' that maximizes the average uniqueness $\overline{\mathcal{U}}(x, \{l'\} \cup m^*)$. Line 5 computes the gradient of the average uniqueness if l' would be selected. If the value of the gradient is smaller than the threshold ϵ, the algorithm terminates and the final configuration m^* is returned. Otherwise, landmark l' is added to the configuration m^*, removed from the set of candidate landmarks, and the algorithm continues and tries to select another landmark. Assuming a finite and discrete state space, the complexity of the algorithm is $O(KM^2)$ where $K < N$ is the number of selected landmarks and M is the size of the state space. The factor M^2 is a consequence of the computation of the average uniqueness that requires the computation of the uniqueness — $O(M)$ — for each state in the state space.

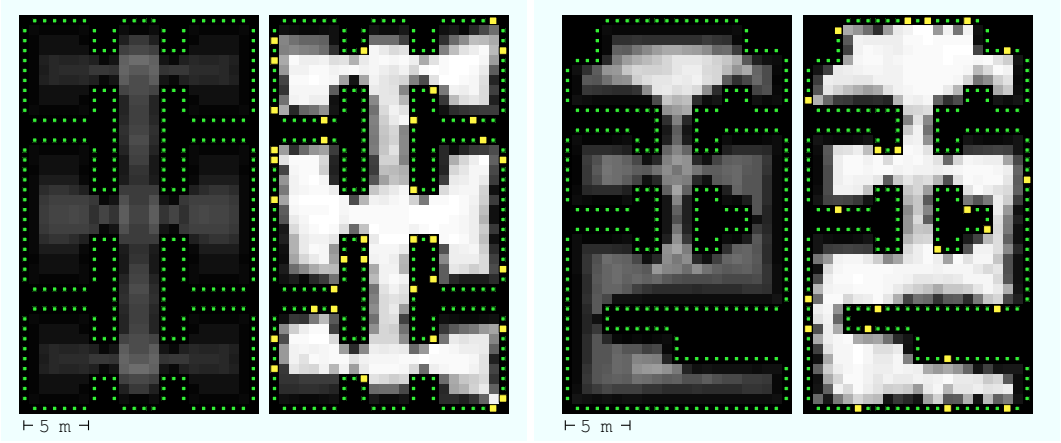


Figure 5.3: Improving the uniqueness in the environment by placing landmarks. The figure shows the uniqueness before and after placing the landmarks for two different environments. The uniqueness at each pose is projected onto the grid map by minimizing over the orientation. The lighter the color, the higher the uniqueness. Also shown are the landmark configurations obtained using our approach. The highlighted locations correspond to the landmarks.

5.3 Experimental Evaluation

To evaluate the improvement in the localization performance obtained when landmarks were placed according to our proposed algorithm, we carried out a set of experiments in simulation and on a real robot. A 2-dimensional occupancy grid with a resolution of 0.5 meters was used to represent the environment. The set of candidate landmarks consisted of all occupied cells in the grid. The sensor used for our experiments was a laser range scanner that in addition to the range and bearing, also returned the reflectivity of the measured objects. As landmarks we considered stripes of retro-reflective tape (see Figure 5.6). Based on the reflectivity we classified individual measurements into pure range measurements and measurements that correspond to landmarks.

As observation model $p(z \mid x, m)$ we used a variant of the likelihood field model described in Chapter 2. In this model, the individual range measurements are assumed to be independent of each other. The likelihood of each pure range measurement is computed according to the distribution $p_r(z^i) \sim \mathcal{N}(d_r, \sigma_r^2)$ based on the distance d_r between the endpoint of the measurement z^i and its closest obstacle in the map m. Here, $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and standard deviation σ. In the case of landmark measurements the distribution $p_l(z^i) \sim \mathcal{N}(d_l, \sigma_l^2)$ is used,

where d_l denotes the distance to the closest landmark in the map. The likelihood of an observation $z = \{z^1, \ldots, z^K\}$ is then computed as

$$p(z \mid x, m) = \prod_{i=1}^{K} p_l(z^i)^{\delta(z^i)} \cdot p_r(z^i)^{(1-\delta(z^i))}\,, \tag{5.9}$$

where

$$\delta(z^i) = \begin{cases} 1 & \text{if } z^i \text{ corresponds to a landmark} \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

This simple general model does not take visibility constraints into account and assumes a perfect landmark detection. However, it can be efficiently evaluated and is sufficient for the purpose of our experiments.

To compute the uniqueness as specified in Equation (5.3), the state space (x-y coordinates and orientation θ) was divided into cells of 0.5 meters and a resolution of 90 degrees was used for the orientation. Ray-tracing was used to simulate the most likely observations needed to compute the uniqueness value at every pose. Figure 5.3 shows the landmark configurations obtained with our approach for two artificial maps. Also shown in the maps is the uniqueness in x-y space, minimized over the orientation θ, before and after placing the landmarks. As threshold ϵ in our landmark placement algorithm we set the minimum value of the gradient to 1. For our specific sensor model Equation (5.9), a theoretical upper bound for the average uniqueness is given by $1/\sqrt[K]{2\pi\sigma^2}$, where $\sigma = \max(\sigma_r, \sigma_l)$. Note that our approach is not restricted to grid-based representations, it only requires a way to compute the uniqueness for the specific representation. Additionally, the computation of the upper bound for the average uniqueness is not strictly necessary. Alternatively, the empirical maximal value can be used instead. The disadvantage of that strategy is that the algorithm can't stop until the maximal average uniqueness value has been reached.

5.3.1 Global Localization Using Artificial Landmarks

To evaluate the landmark configurations obtained using our approach in the task of global localization, we generated 50 different random trajectories for each of the environments shown in Figure 5.3. In addition to noise in the range simulations, we also simulated false positives and false negatives in the landmark detections. The localization algorithm was executed 5 times for each trajectory using $10\,000$ particles initially uniformly distributed in the state space. We compared, for the same num-

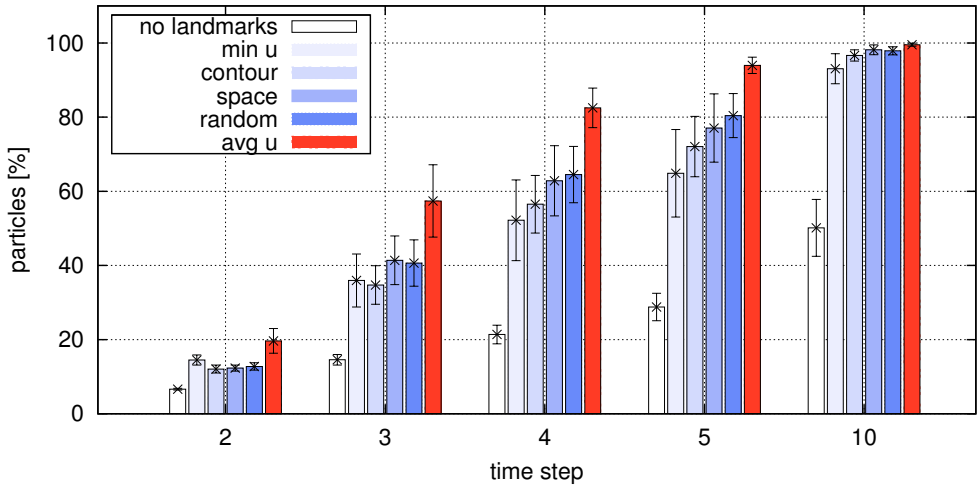


Figure 5.4: Evaluation of the landmark configuration obtained using our approach vs. alternative approaches in the task of global localization. The plots show the fraction of particles within a 1.5 meters radius around the true pose as a function of time. The values correspond to the mean and standard deviation for different repetitions of the global localization.

ber of landmarks, the configurations obtained with our approach (*avg u*) against the configurations obtained with four alternative approaches:

- ***uniform contour sampling*** (*contour*): Distributes the landmarks approximately uniformly throughout the contours of the environment. The first landmark is randomly selected from the set of candidate landmarks. Additional landmarks are selected by choosing the candidate landmark closest to the previously selected one until no more landmarks can be selected. Every time a landmark is selected, all landmarks within a specified radius are removed from the set of candidates. This radius is chosen so that the sampling approximately covers the whole map.

- ***uniform space sampling*** (*space*): Distributes the landmarks roughly uniformly throughout the environment. This approach divides the environment into squared regions of equal size and selects the candidate landmark closest to the center of each non-empty region.

- ***random sampling*** (*random*): Distributes the landmarks randomly throughout the contours of the environment. Landmarks are randomly selected from the set

of candidate landmarks. Every time a landmark is selected, all landmarks within a specified radius are removed from the set of candidates.

- ***maximize the minimal uniqueness*** (*min u*): The approach described in Algorithm 5.1 was modified so that it would maximize the minimum uniqueness in the environment $m^* = \text{argmax}_{m \subseteq \mathcal{V}} \left(\min \mathcal{U}(x, m) \right)$, instead of maximizing the average uniqueness,

The results of the experiment for one of the environments (left one in Figure 5.3) are shown in Figure 5.4. As performance metric for the global localization task we considered the fraction of particles within a 1.5 meters radius around the true pose after 2, 3, 4, 5 and 10 integrations of measurements (time steps). The values correspond to the mean and standard deviation for the different trajectories and runs of the localization. As can be seen in the figure, the configuration obtained by our method improves the global localization performance best since particles are more quickly converging towards the true pose of the robot. A t-test showed that the improvement was significant on the 5% level for all the evaluated environments, time steps and alternative approaches. Clearly, the amount of improvement obtainable depends on the inherent uniqueness of the environment. A larger improvement can be obtained for inherently ambiguous environments (left one in Figure 5.3) than for inherently unique environments (right one in Figure 5.3). The first 3 alternative approaches, *contour*, *space*, and *random* are simple and fast, but do not take into account the ambiguities that can originate when selecting landmarks, and the resulting landmark configurations are therefore not as good for improving the localization performance as the ones obtained with our approach. The fourth approach, *min u*, has the property that a lower bound for the uniqueness in the environment is guaranteed. However, this does not provide a significant improvement in the localization performance.

5.3.2 Choosing the Number of Landmarks

The goal of this experiment was to evaluate the performance of our gradient-based heuristic when determining automatically the number of landmarks to be placed. Figure 5.5 shows the fraction of particles within a 1.5 meters radius around the true pose of the robot after 2, 3, and 10 integrations of measurements as a function of the number of selected landmarks. Also indicated in the figure are the values corresponding to the number of landmarks selected by our gradient-based heuristic ($\sim 10\%$) when using a threshold of 1. This corresponds to a 45 degrees positive gradient. The moti-

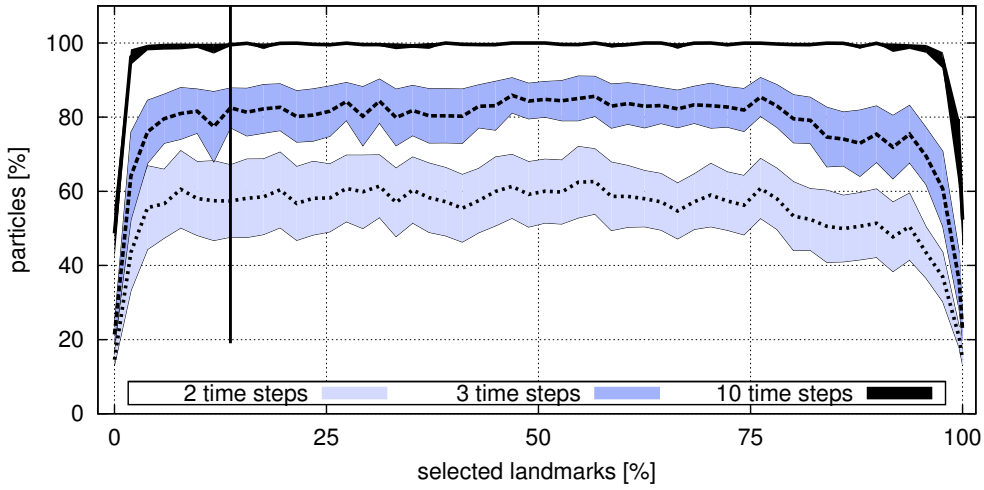


Figure 5.5: Global localization performance as a function of the number of selected landmarks. The plots show the fraction of particles within a 1.5 meters radius around the true pose of the robot after 2, 3, and 10 time steps. The values correspond to the mean and standard deviation for different repetitions of the experiment. The number of landmarks selected by our approach ($\sim 10\%$) is indicated by the vertical line.

vation for choosing this value is that increasing the percentage of selected landmarks by 1% provides less than a 1% increment in the normalized average uniqueness. Using a different value for the threshold, or weighting differently the parameters (average uniqueness vs. fraction of selected landmarks) the number of selected landmarks can be controlled. As can be seen in Figure 5.5 selecting more landmarks does not provide an improvement in the localization performance. Furthermore, the number of landmarks selected is well beyond the point where fewer landmarks would cause the localization performance to decrease drastically.

An additional result of this experiment is that, as can be seen in the figure, the behavior of the localization performance as a function of the number of selected landmarks is similar to the behavior of the average uniqueness (see Figure 5.2). This experimental result suggests a direct connection between the average uniqueness in the environment and localization performance.

5.3.3 Experiments with Real Data

We also evaluated our approach using data gathered with a MobileRobots Pioneer P3-DX robot equipped with a SICK LMS 291 laser range finder (see Figure 5.6). We

Figure 5.6: The building 103 of Department of Computer Science at the University of Freiburg consisted on a long, featureless corridor of approximately 80×3 meters size. Data was gathered using a MobileRobots Pioneer P3-DX robot equipped with a SICK LMS 291 laser range finder. As landmarks (right image), stripes of retro-reflective material were taped to the walls in the locations indicated by our approach.

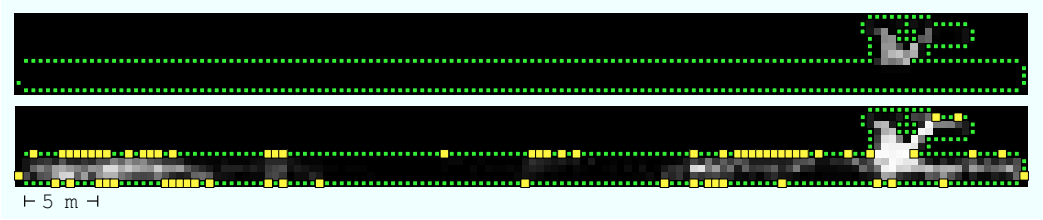


Figure 5.7: Landmark configuration and uniqueness before and after placing the landmarks for the *building 103* data set gathered with a real robot using a laser range scanner.

steered the robot through building 103 of the Department of Computer Science at the University of Freiburg and created an occupancy map of the environment using a standard SLAM technique [Grisetti *et al.*, 2005]. The environment consisted of a long, featureless corridor of approximately 80×3 meters size. Figure 5.7 shows the landmark configuration obtained using our approach. In order to make the environment more ambiguous, range measurements larger than 10 meters where disregarded. For building the map, however, the full 80 meters depth range of the laser scanner was used. The map resolution and state space discretization are described in Section 5.3. As landmarks, stripes of retro-reflective material were taped to the walls in the locations indicated by our approach. We used a threshold on the reflectivity value to classify the laser measurements caused by the landmarks.

After placing the landmarks, we steered the robot again through the environment and used the above mentioned SLAM technique to obtain an approximated ground truth for comparison. For the statistical analysis we divided the data into 5 parts and evaluated the landmark configuration in the task of global localization as described in Section 5.3.1. Figure 5.8 shows the results of the experiment. As expected, a substantial improvement in the localization performance was obtained when using the landmarks. The performance is, however, lower than the one obtained in simulation. This is mostly due to the simplistic sensor model considered for the experiments. The model is sufficient for the purpose of our evaluation, but we expect that using a better model, for example one that considers the distance and angle of incidence of the beams at the moment of detecting a landmark, would produce better results for real data.

We also evaluated the configuration obtained using our approach in the task of position tracking. Figure 5.9 shows the Euclidean distance between the average pose of the particle set and the true pose of the robot as a function of time. As can be seen in the figure, using landmarks can also improve the accuracy of the localization in the ambiguous areas of the environment. With and without landmarks the error grows as the robot moves along the corridor and decreases when the ends of the corridor are visible ($t \sim 150, 300, 450, 600$). With landmarks, however, the error is substantially smaller when moving along the corridor.

5.4 Related Work

Many localization techniques rely on natural features of the environment [Leonard and Durrant-Whyte, 1991; Burgard *et al.*, 1996] to estimate the robot's pose. These approaches are particularly attractive as they do not require the environment to be

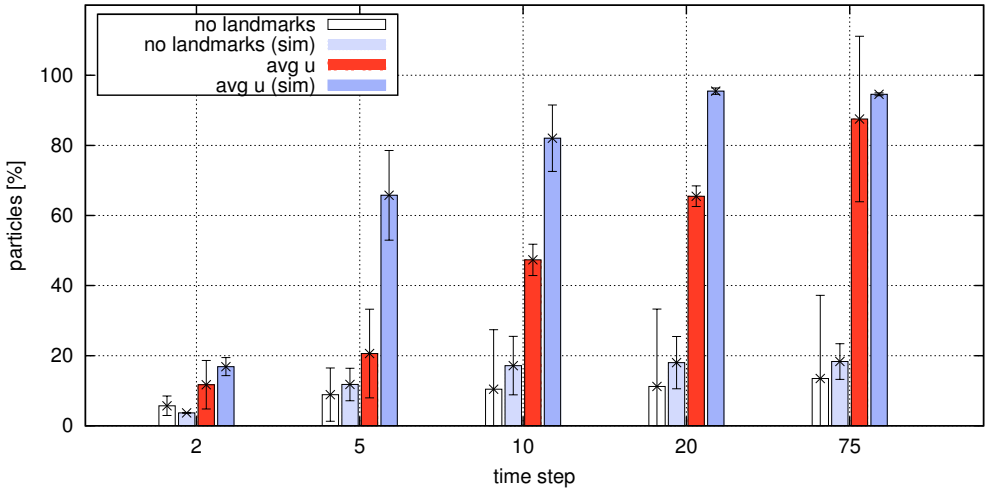


Figure 5.8: Global localization performance for the *building 103* data set. The plots show the fraction of particles within a 1.5 meters radius of the true pose of the robot as a function of time. Also shown are the results obtained in simulation (*sim*).

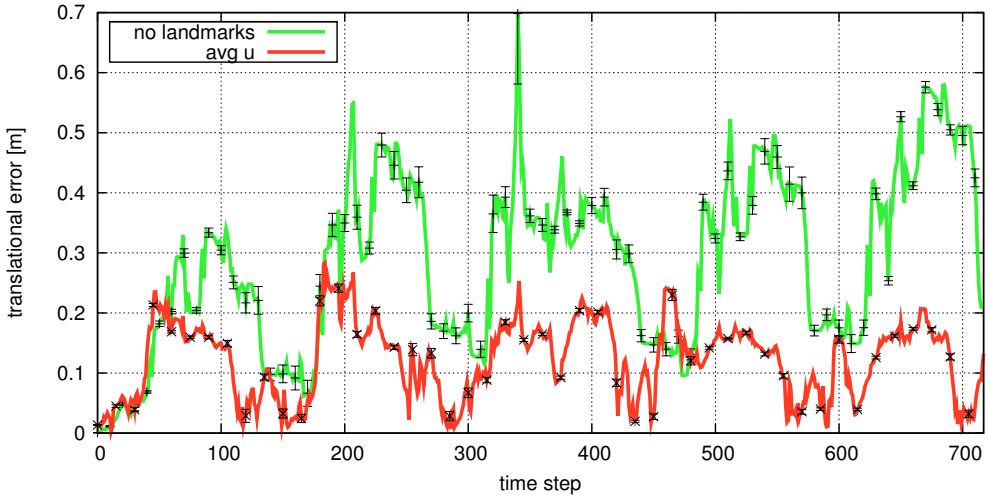


Figure 5.9: Average localization error during position tracking with and without utilizing landmarks for the *building 103* data set.

modified. However, inherently ambiguous environments make these approaches more likely to fail. Artificial landmarks offer the possibility of improving the reliability of the localization. Some approaches [Howard *et al.*, 2001; Rafflin and Fournier, 1996] consider artificial landmarks that can be uniquely identified. Such approaches greatly simplify the localization problem, but they require a landmark coding and non-trivial identification system. For this reason, we consider only indistinguishable landmarks.

The landmark placement problem as addressed in our work can be formulated as the problem of selecting a subset of landmarks out of a finite set of candidate landmarks. Sutherland and Thompson [Sutherland and Thompson, 1993] where one of the first to address this problem. They demonstrate that the localization error depends on the configuration of the selected landmarks. Salas and Gordillo [Salas and Gordillo, 1998] propose a simulated annealing technique to find the landmark configuration that maximizes the size of the region from where a landmark can be seen. Sinriech and Shoval [Sinriech and Shoval, 2000] specify a set of constraints about the number of landmarks and their distance to critical locations in the environment, and formulate landmark placement as a nonlinear optimization problem. Sala *et al*. [Sala *et al.*, 2004] decompose the environment into regions from which a minimum number of landmarks can be observed. They use a graph-theoretical formulation to find the decomposition with the minimum number of regions. All of the above mentioned approaches rely on pure geometrical reasoning for estimating the pose of the robot. In contrast to that, our approach is tightly coupled with a robust, probabilistic localization framework.

Other researchers have also focused on the localization performance at the moment of selecting landmarks. Thrun [Thrun, 1998], for example, uses a neural network to extract features from the sensor data and selects the subset of those features that minimizes the average posterior localization error. Lerner *et al*. [Lerner *et al.*, 2006] formulate the problem as a semi-definite programming problem and specify a cost function to weight different localization parameters according to the specific task at hand. Strasdat *et al*. [Strasdat *et al.*, 2009] use reinforcement learning to obtain an online landmark selection policy. The approach of Zhang *et al*. [Zhang *et al.*, 2005] selects, at every time step, the set of landmarks that minimizes the entropy of the resulting posterior distribution. All of these methods operate online and are concerned with the salient features observed at every time step during localization. In contrast, our approach works in an offline fashion, and instead of observed features, we rely physical objects as landmarks. The main difference with previous approaches is that, using a measure for the uniqueness of a pose, we explicitly consider the symmetries and ambiguities that can originate when placing landmarks.

5.5 Conclusions

In this chapter we presented a landmark placement approach that seeks to reduce the overall ambiguity in the environment to improve the localization performance of a mobile robot. To this extend we proposed a measure for the uniqueness of a robot pose based on the appearance of the environment as observed by the robot. Due to the combinatorial nature of the landmark placement problem, we introduced an approximative approach that incrementally selects landmark locations from a set of candidate locations and thereby maximizes the average uniqueness in the environment. Furthermore, we described a concrete application in the context of localization with laser range scanners given a grid-based representation of the environment. We evaluated our approach for different environments in simulation and using real data. The results demonstrate that our approach yields substantial improvements in the localization performance.

Part II

Modeling Temporal Dynamics

Chapter 6

Hidden Markov Models for Situation Recognition in Traffic Scenarios

In the previous chapters of this thesis, we represented and reasoned about the environment dynamics at a low level of abstraction. So far, the only features that we have considered are the changes in the occupancy of the space. In the remaining two chapters, we address the problem of reasoning in dynamic environments at a higher level of abstraction.

A fundamental requirement for an autonomous system to be able to act intelligently is the continuous monitoring and understanding of the current situation it is involved in. Knowing what is going on is relevant for predicting what will happen, which in turn can be used to make informed decisions, avoid risks, and, in general, improve the performance of the system. Situation recognition, however, is not an easy task even if the state of the system or its environment can be estimated accurately. To robustly recognize the current situation at any given time, the temporal context needs to be taken into account. Additionally, the system must be able to deal with ambiguities, since there may be more than one possible interpretation, and some interpretations might be contradictory. Furthermore, the system needs to deal with uncertainty in the environment, sensor noise, and inaccuracies in its models. On top of this all, the system must recognize situations as they are evolving, that is, in an online fashion.

In this chapter, we present a framework for modeling and online-recognition of situations. Although the framework is generic, we focus on a driver assistant application in traffic scenarios and consider situations that typically occur in highway-like driving settings. The situations detected by our current system include *passing*, *following*, and *aborted passing* situations. Within our framework, the process of change is viewed as

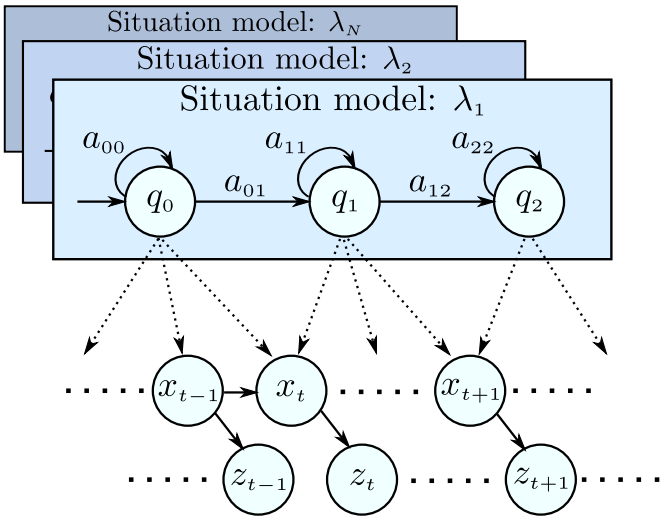


Figure 6.1: Schematic representation of the proposed framework for situation modeling and recognition. Each situation type is described by an individual HMM λ_i. A situation instance exists as long as the corresponding HMM recognizes the state sequence being generated.

a series of snapshots, each describing the state of the system at a particular time. Based on this characterization, we speak of *situation types* and *situation instances*, where a situation instance is defined as a state sequence that has some meaningful interpretation. A situation type, on the other hand, is the set of all situation instances that can be grouped under the same interpretation. We take a model-based approach in which hidden Markov models (HMMs), as described in Chapter 2, are used for characterizing and recognizing situations. Each situation type is described by an individual HMM, which specifies the admissible state sequences that correspond to an occurrence of the given situation. A graphical representation of the proposed framework is shown in Figure 6.1. The state space model in the lower part of the figure corresponds to a dynamic Bayesian network that characterizes the system. The upper part of the figure corresponds to a layer of different situation-HMMs, which are evaluated against the estimated state of the system x_t at each point in time t.

In this chapter, we present a practical approach for modeling and recognizing situations in an online setting. As mentioned above, we show how our framework can be used for characterizing typical situations in a vehicular traffic scenario, and how situation instances can be tracked while they are developing. Experimental results using real and simulated data show that our system can recognize and track multiple situa-

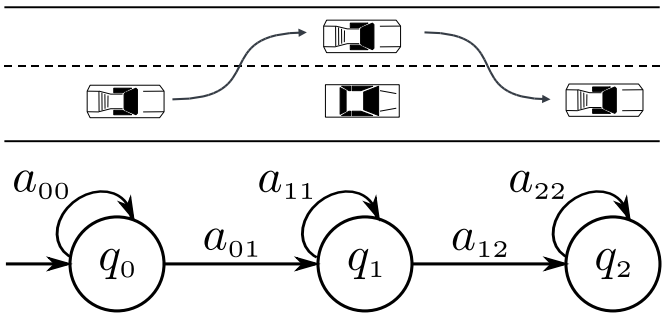


Figure 6.2: A *passing* maneuver, in which the reference car (square car in the middle) is being passed on its left hand side by another one. In this case, we divided the maneuver into three stages and, thus, used a 3-state HMM over abstract world states as a model (bottom).

tion instances in parallel, and make sensible decisions between competing hypotheses. Additionally, we show that our models can be used for making rough predictions about the position of the tracked vehicles.

6.1 Modeling Situations using HMMs

Our approach to modeling the dynamics of realistic systems, such as vehicular traffic, is to assume two layers of abstraction: first, on a higher abstraction level, the so-called *situation models* describe how the system evolves over longer periods of time at a lower spatial resolution (e.g., "car A passes car B on the left"). Secondly, on a more detailed level, a *state-space model* describes the concrete dynamics of the environment involving the relationship between the state x_t of the system and the observations z_t.

Fully interweaving both abstraction layers would lead to an intractable model in all but the simplest cases. Therefore, the two layers are loosely coupled as visualized in Figure 6.1, that is, the posterior state estimates in the state-space models are treated as fixed "observations" by the situation models. Concretely, as state-space model we assume a dynamic Bayesian network, in which the state x_t and observation z_t at time t are characterized by a set of random variables (see Figure 6.1). The state x_t of the system at time t is estimated from the sequence of previously obtained observations $z_{1:t}$ using the recursive state estimation scheme presented in Chapter 2.

On the more abstract level, a *situation instance* is defined as a sequence of states that has some meaningful interpretation. A *situation type* corresponds to the set of all situation instances that are grouped under the same interpretation. To characterize a

situation type s, we use a hidden Markov model λ_s, that describes the stereotypical state sequence corresponding to the situation type over the wide range of variations inherent to the different situation instances. A situation HMM consists of a tuple $\lambda = (Q, A, B, \pi)$, where

- $Q = \{q_0, \ldots, q_N\}$ represents a finite set of N states.
- $A = \{a_{ij}\}$ is the state transition matrix in which each entry a_{ij} represents the probability of a transition from state q_i to state q_j.
- $B = \{b_i(x)\}$ is the observation model, where $b_i(x)$ represents the probability of observing the state x of the system while being in state q_i in the situation model.
- $\pi = \{\pi_i\}$ is the initial state distribution, where π_i represents the probability of state q_i being the initial state.

Although the states Q in a HMM are hidden, a concrete meaning can often be associated with them. In our case, we choose Q as a set of N salient states x from the state-space model. The transition probabilities a_{ij} specify the admissible state sequences that correspond to an instance of that situation. Consider, for example, the passing maneuver illustrated in Figure 6.2, in which a car is passed on the left side by another car. We can describe this type of situation using an HMM with three states q_0, q_1, and q_2 where the first state, q_0, corresponds to the passing car being behind of the reference car, q_1 corresponds to the passing car being on the left, and q_2 corresponds to the passing car being in front of the reference one.

One possible way to characterize the observation model B is, for example, to discretize the state space using relational logic and define the individual observations based on the discretized state space (see [Meyer-Delius *et al.*, 2007]). In relational logic, an *atom* $\mathtt{r}(\mathtt{t_1}, \ldots, \mathtt{t_n})$ is an n-tuple of terms $\mathtt{t_i}$ with a relation symbol $\mathtt{r}$. A *term* can be either a variable $\mathtt{R}$ or a constant $\mathtt{c}$. Relations can be defined over the state variables or over features that can be directly extracted from them. Table 6.1 illustrates possible relations defined over the *distance* and *bearing* state variables in our traffic scenario. An *abstract state* is then defined as a conjunction of logical atoms (see [Cocora *et al.*, 2006]). For example the abstract state $q_i \equiv (\mathtt{far}(\mathtt{R}, \mathtt{R}'), \mathtt{behind}(\mathtt{R}, \mathtt{R}'))$, which represents all states where a car is *far* and *behind* another car. Using relational logic to discretize the space, the states of a situation HMM would then correspond to abstract states.

The observation model assigns to each relation the probability mass associated to the interval of the continuous state space that it represents. The resulting distribution

Relation	Distances
$\texttt{equal}(\mathtt{R}, \mathtt{R}')$	$[0\,\text{m}, 1\,\text{m})$
$\texttt{close}(\mathtt{R}, \mathtt{R}')$	$[1\,\text{m}, 5\,\text{m})$
$\texttt{medium}(\mathtt{R}, \mathtt{R}')$	$[5\,\text{m}, 15\,\text{m})$
$\texttt{far}(\mathtt{R}, \mathtt{R}')$	$[15\,\text{m}, \infty)$

Relation	Bearing angles
$\texttt{in\_front\_of}(\mathtt{R}, \mathtt{R}')$	$[315°, 45°)$
$\texttt{right}(\mathtt{R}, \mathtt{R}')$	$[45°, 135°)$
$\texttt{behind}(\mathtt{R}, \mathtt{R}')$	$[135°, 225°)$
$\texttt{left}(\mathtt{R}, \mathtt{R}')$	$[225°, 315°)$

Table 6.1: State space discretization using relational logic: example distance and bearing relations for a traffic scenario.

is thus a histogram that assigns to each relation a single cumulative probability. Such a histogram can be thought of as a piecewise constant approximation of the continuous density. The probability $b_i(x)$ of observing state x while being in state q_i is then computed as the product of the probabilities associated to each of the relations that constitute q_i. This representation of the continuous state space, is basically a discretization of the state space into a finite number of discrete dimensions with a finite number of values. The concrete discretization used depends on the particular problem at hand. A coarse discretization, for example, enables powerful generalizations and can be expressed relatively economically. However, if the partitions are too large, they will not allow differentiation between similar situations. If they are too small, the advantage of abstracting the continuous space will be reduced, and the description of situations will become increasingly complicated.

The characterization of the observation model used for the framework described in this chapter is given by a finite mixture of K Gaussian distributions

$$b_i(x) \;=\; \sum_{k=1}^{K} c_{ik}\,\mathcal{N}(\mu_{ik}, \Sigma_{ik})\,, \tag{6.1}$$

where x is the observed system state, c_{ik} is the mixture coefficient for the k-th mixture in situation state q_i, and $\mathcal{N}(\mu_{ik}, \Sigma_{ik})$ is the multivariate Gaussian distribution with mean μ_{ik} and covariance matrix Σ_{ik} associated to the k-th mixture in q_i. Note that the observed state x, corresponding to the estimated state of the system at the state-space level, is treated as an observation at the situation level.

In order to estimate the parameters of a situation HMM λ, we use an approach based on the expectation-maximization (EM) algorithm. As described in Chapter 2, this is an approximative iterative optimization technique for maximizing the likelihood of the data. The algorithm takes an initial estimate of the parameters and greedily improves it by following the likelihood gradient. We assume that the number of states N

and mixtures K in the observation model are fixed and given. Therefore, the initial state distribution π, the transition matrix A, and the observation model B are the free parameters to be learned. The training data for each situation type s consists of a set of observation sequences $X_s = \{x^1, \ldots, x^c\}$, where each $x^i = \{x_1^i, \ldots, x_{t_i}^i\}$ is a sequence of states of the state space model that corresponds to an instance of situation s.

A key issue when using approximative iterative optimization techniques is how to choose the initial parameter estimates. A common approach is to set π and A randomly or uniformly. However, for B, good initial estimates are essential. In this work the initial estimates for π, A, and B were all specified by manually dividing the observation sequences into multiple segments and averaging the observations within segments.

6.2 Recognizing Situation Instances

Given a set of M trained situation models $\lambda_1, \ldots, \lambda_M$, and a sequence $x_{1:t}$ of states of the state space model, our approach to situation recognition is based on evaluating the likelihood $p(x_{1:t} \mid \lambda_i)$ of the sequence for each model λ_i. This likelihood is computed incrementally using the *forward procedure* (see [Rabiner, 1989]) given as

$$p(x_{1:t} \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i), \tag{6.2}$$

where

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(x_{t+1}), \tag{6.3}$$

and

$$\alpha_1(j) = \pi_j b_j(x_1). \tag{6.4}$$

At each point in time t, the framework incrementally updates the likelihoods computed at time $t-1$ independently for the different models.

In certain scenarios it is reasonable to consider two or more situation types as being mutually exclusive or competing. In this kind of problems, we would like to be able to select, among the competing models, the most likely one. Assuming that the stereotypical sequences of the competing situation types can be differentiated and that the learned models accurately characterize them, we can use the likelihood $p(x_{1:t} \mid \lambda_s)$

of the observation sequence given the different models λ_s to select the one that provides the better explanation for the sequence [Rabiner, 1989].

For deciding between two competing situation models, we compute the posterior odds, which provides a way of evaluating evidence in favor of a probabilistic model relative to an alternative one. The posterior odds $p_{\lambda_2}^{\lambda_1}$ for two competing models λ_1 and λ_2 given an observation sequence $x_{1:t}$ is computed as

$$p_{\lambda_2}^{\lambda_1} = \frac{p(x_{1:t} \mid \lambda_1)\, p(\lambda_1)}{p(x_{1:t} \mid \lambda_2)\, p(\lambda_2)}, \tag{6.5}$$

that is, the ratio of the likelihoods of the models being compared given the data multiplied by the model priors. The likelihood $p(x_{1:t} \mid \lambda)$ of an observation sequence $x_{1:t}$ given a model λ can be computed efficiently using the forward procedure as described above. The prior probabilities $p(\lambda)$ allow us to include information about how likely a given model is, prior to any evidence. We learn these from the training data using simple counting.

6.3 Experimental Evaluation

Our framework was implemented and tested in a vehicular traffic scenario using a simulated driving environment as well as with real data. The goal was to show that our framework could be used to model and recognize different situations instances in a dynamic multi-vehicle environment. We considered three different situations that typically occur on a highway, namely *passing*, *aborted passing*, and *following*.

- ***passing***: A passing car approaches the reference car from behind, it passes on the left, and finally ends up in front of it.
- ***aborted passing***: as in the *passing* situation, a passing car approaches from behind, but instead of actually passing the reference car, it slows down before being abeam and ends up behind it again.
- ***follow***: the reference car is followed from behind by another car at a short distance and at approximately the same velocity.

For the experiments, the state x_t of the system at time t consisted of the relative distance r_t^i, relative bearing ψ_t^i, and relative speed v_t^i of each car i around the reference car. These features were sufficient to characterize the modeled situations, being

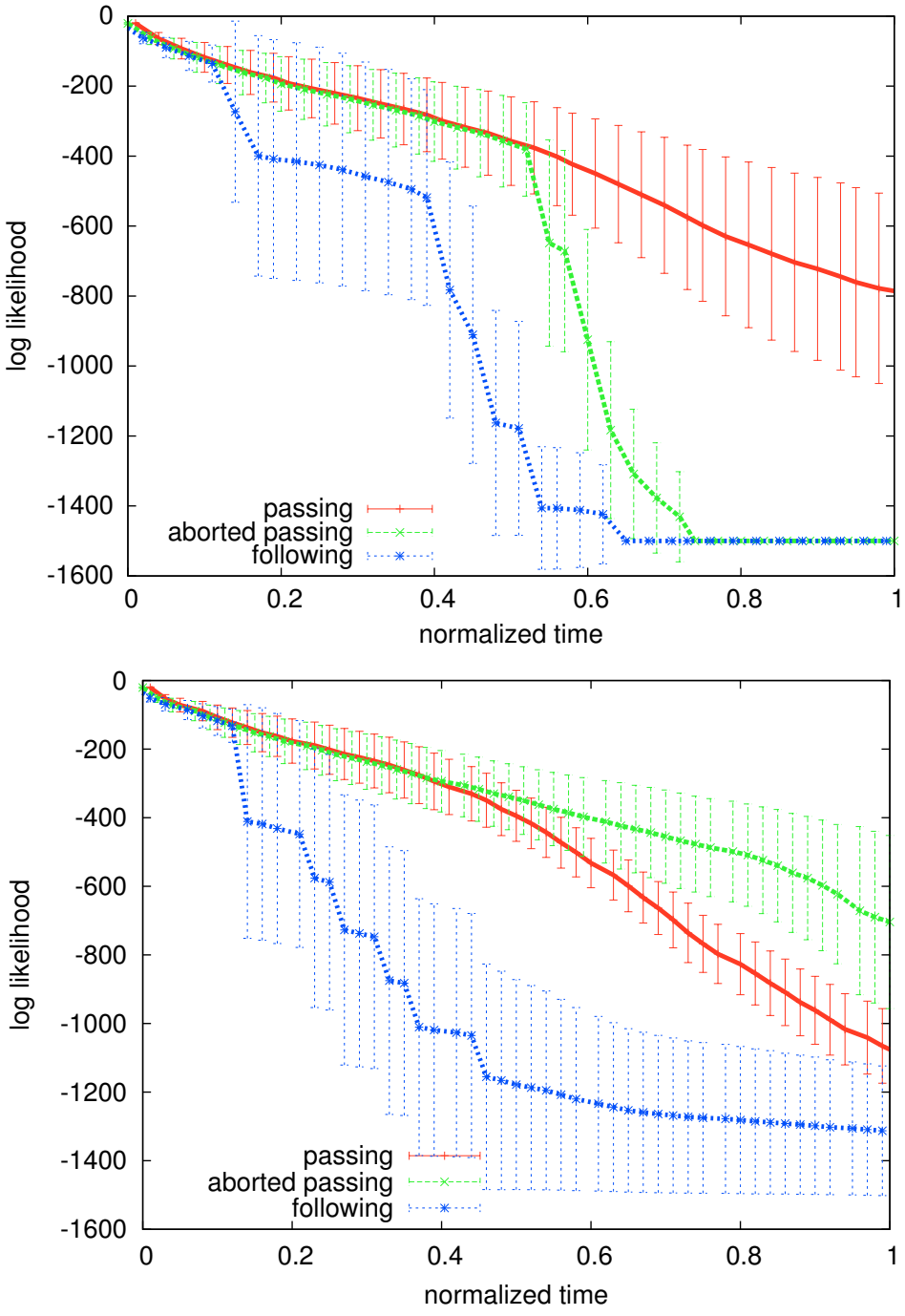


Figure 6.3: Average likelihood of 10 observation sequences corresponding to a passing (left) and an aborted passing (right) maneuver according to the three different situation models: *passing*, *aborted passing*, and *following*.

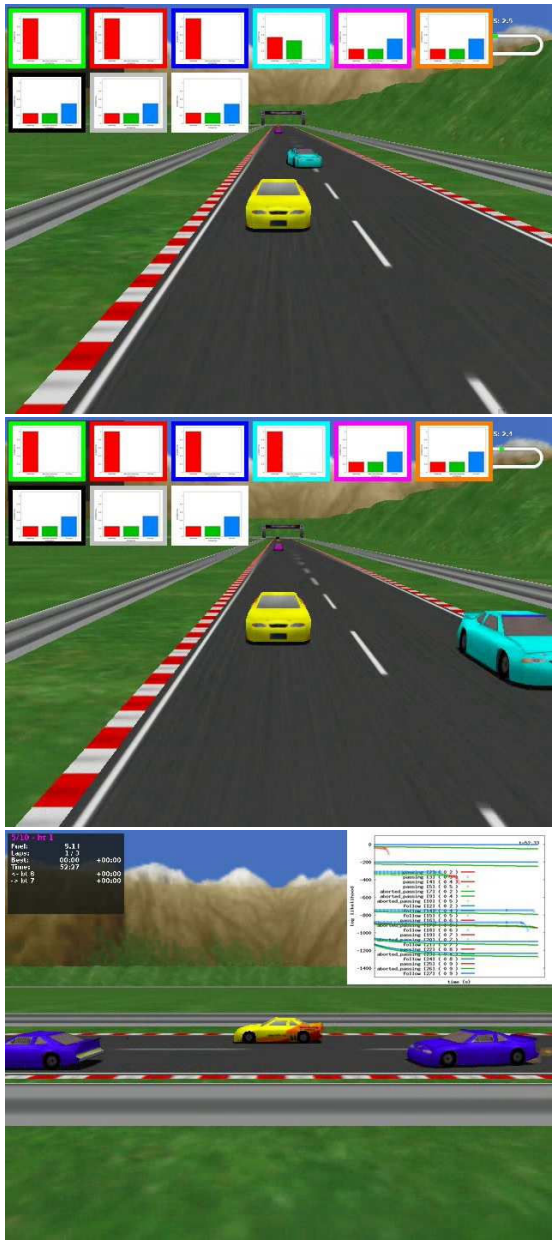


Figure 6.4: The situation models were learned from a large set of simulated traffic scenarios involving multiple users using a simulated driving environment. Top: it is not yet clear whether the cyan car will actually pass the yellow one (see the forth bar chart on top). Middle: at this point, the passing maneuver was clearly identified. Bottom: Inspecting the evolution of situation likelihoods involving many agents.

also robust against variations in the different situation instances. In our experiments, each situation HMM consisted of 5 states with one three-dimensional Gaussian as observation model. Some HMMs could not be trained when using more states due to insufficient data, and using less states made the models too general and reduced their discriminative capacity. The initial estimates of the parameters π, A, and B of the models where manually set by segmenting the situation stereotypical sequence into meaningful states. The EM algorithm was then used to optimize the parameters so as to maximize the likelihood of the data.

6.3.1 Recognizing Individual Situation Instances

The goal of this experiment was to demonstrate that our approach can be used to successfully characterize and track different situation types. We first trained the different situation models using sequences generated in a driving simulator (TORCS) [Espié and Guionneau, 1997]. The training data, consisting in 20 instances of each situation type, was generated using randomly selected speeds for the cars in different circuits. As test sequences, 10 passing and 10 aborted passing maneuvers were generated. Figure 6.3 plots the average log likelihood and standard deviation of the test sequences according to the different situation models. Since different executions of a maneuver produce sequences of different length, they were first normalized using interpolation in order to compare them.

After an approaching car was detected, that is, when an approaching car was within a 50 meter radius of the reference car, we started computing the likelihood of the state sequence for the different situation models as described in Section 6.2. In the figures, it can be observed how the likelihood given a model measures how well the model explains the current state sequence. For the passing maneuvers (top plot) the *passing* situation model provides the best explanation compared with the other models. For the aborted passing maneuvers (bottom plot), however, the model does not perform as well. For example, observe how at approximately 20% of the maneuver, as the passing car starts changing to the left lane, the likelihood according to the *following* model starts to decrease. This occurs since the model expects the following car to remain behind the reference car and therefore ceases to provide an explanation for the observations. Similarly, at approximately 50% of the maneuver, when the passing car is abeam, the likelihood according to the *aborted passing* model starts to decrease too.

The error bars in the figure capture the variance in the different executions of the maneuvers. However, this variance is greatly inflated by the normalization over the

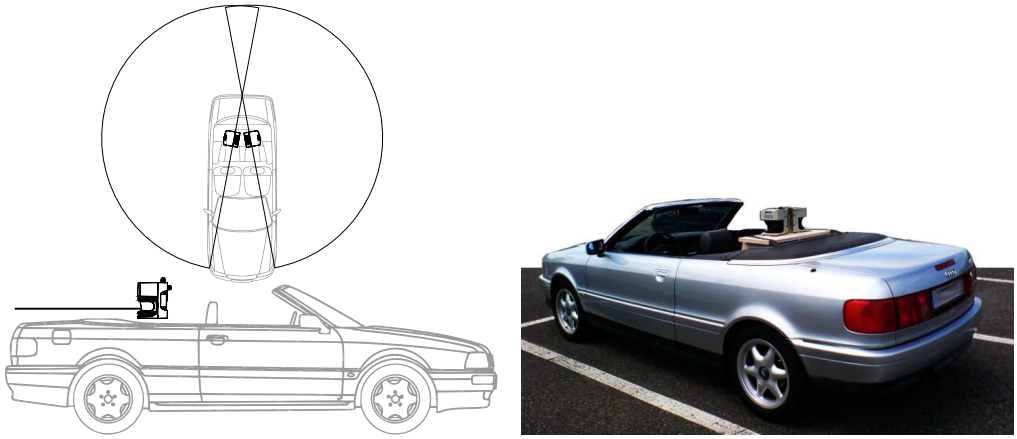


Figure 6.5: Arrangement of two SICK laser range finders on a convertible used for gathering real data. Each laser has a field of view of 180 degrees and can detect objects as far as 80 meters with an angular resolution of 1 degree at 75 Hz.

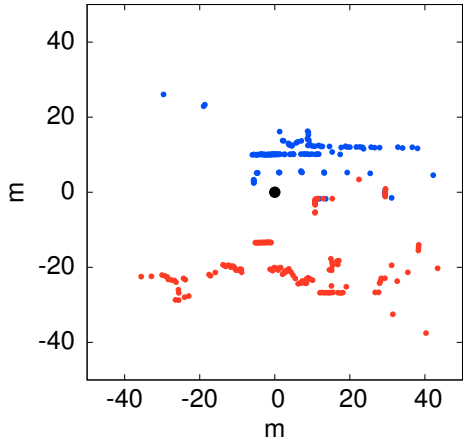


Figure 6.6: The arrangement of the two lasers (see Figure 6.5) provided a 340 degrees field of view.

length of the sequences. This is why the bars appear so large. Specially after the likelihood of a sequence given a model falls below a certain threshold. In this experiment, we set the minimum allowed log likelihood value to -1500 for better visualization of the results.

6.3.2 Recognizing Multiple Situation Instances

A similar experiment was carried out with additional cars driving simultaneously to evaluate the performance and robustness of the framework. The scenario consisted of nine different cars passing the reference car. Every time one of the cars came within a 50 meter radius of the reference car, the three situation models where instantiated for the approaching car, and evaluated as described in Section 6.2. Once a car was outside the tracking range, the associated models where discarded. The results (see Figure 6.4) showed that our approach can also be used in scenarios where multiple cars are being simultaneously tracked, instantiating and eliminating multiple different instances of the situation models as cars appear and disappear from the state space, over extended periods of time. Since each situation instance is evaluated independently from the others, the complexity of the situation tracking algorithm increases only linearly in the number of situation types and cars in the state space.

Recognizing Situation Instances in Real Data

The framework was also evaluated using real data. Two SICK laser range scanners were mounted on a convertible as illustrated in Figure 6.5. Each laser has a field of view of 180 degrees and can detect objects as far as 80 meters with an angular resolution of 1 degree at 75 hertz. The arrangement of the two lasers provided a 340 degrees field of view as illustrated in the figure. Due to the blind spot in the field of view of the laser arrangement, states in which cars were in front of the reference car could not be considered. Data was gathered by driving over more than 50 kilometers on highways and state roads at velocities of up to 110 km/h. Note that in this work, we concentrate only on the recognition of situation instances and do not deal with the tracking of the cars. The trajectories of the cars were extracted using a manually initialized Kalman filter to track the cars.

From the gathered data, only 14 and 8 complete tracks corresponding respectively to *passing* and *following* situations could be successfully extracted. Due to the technical limitations of the sensors together with their arrangement, many situation instances could not be captured, or were captured only partially in the data. Figure 6.7 plots the average log likelihood of 5 observation sequences corresponding to a passing maneuver according to the *passing* and *following* models trained with the real data.

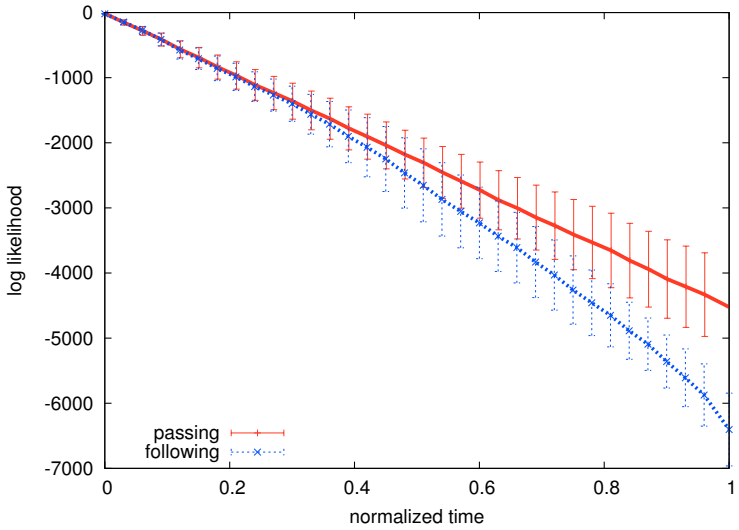


Figure 6.7: Likelihood of 5 observation sequences corresponding to passing maneuvers extracted from real data according to the *passing* and *following* situation models.

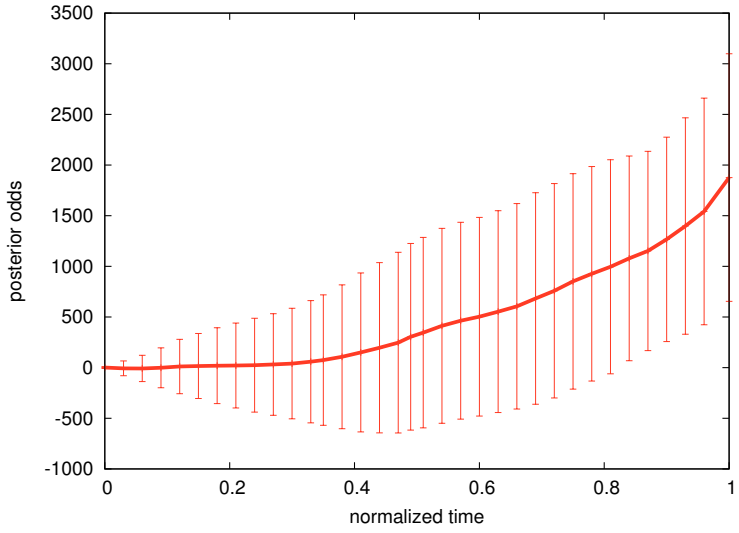


Figure 6.8: Posterior odds in favor of the *passing* situation model compared against the *follow* situation model according to the observations for the passing maneuver.

6.3.3 Competing Situation Models

In this experiment we illustrated how the posterior odds can be used for choosing between different situation models. Figure 6.8 plots the posterior odds $p_{\lambda_f}^{\lambda_p}$ in favor of the *passing* situation model λ_p compared against the *following* situation model λ_f for the real data used in the previous experiment. The model priors $p(\lambda_p)$ and $p(\lambda_f)$ needed to compute the posteriors were obtained form the training data by counting the number of instances of the *passing* and *following* model.

A positive $p_{\lambda_f}^{\lambda_p}$ can be interpreted as evidence provided by the data in favor of the *passing* situation model. The motivation behind using the posterior odds as criterion for model selection can be observed in the results of the previous experiment, in which the likelihood of the sequence according to the model actually corresponding to the executed maneuver is generally higher than the likelihood according to the other models. Figure 6.8 also illustrates how the posterior odds can be used to make decisions between competing situation models as discussed in Section 6.2.

6.3.4 State Prediction

In this experiment we demonstrated how our learned models can be used for predicting the state of the system. State prediction within the HMM framework consists of computing the belief state of the HMM as in Equation (6.3) but without correcting for new evidence. The state $\bar{x}_{t+\tau}$ in the state space model, τ time steps in the future, can then be computed as

$$\bar{x}_{t+\tau} = \frac{\sum_{i=1}^{N} \alpha_{t+\tau}(i)\, \mu_i}{\sum_{i=1}^{N} \alpha_{t+\tau}(i)}, \tag{6.6}$$

where μ_i is the mean of the multivariate Gaussian distribution of the observation model for situation state q_i (see Section 6.1). Figure 6.9 plots the predicted state of a car for one of the sequences gathered from real data. The sequence corresponds to a passing maneuver and the learned *passing* model was used to obtain a 1-second prediction. Our HMM-based situation models allow us not only to describe but also predict situations with complex dynamics. We compare the prediction results of our models against a base-line constant-velocity model. This model can be quite accurate for predictions when the situation is highly linear (as are the situations in our presented experiments) but is extremely sensitive to errors in the state estimation. Our HMM-based models, on the other hand, are robust against these errors since the predictions are based on

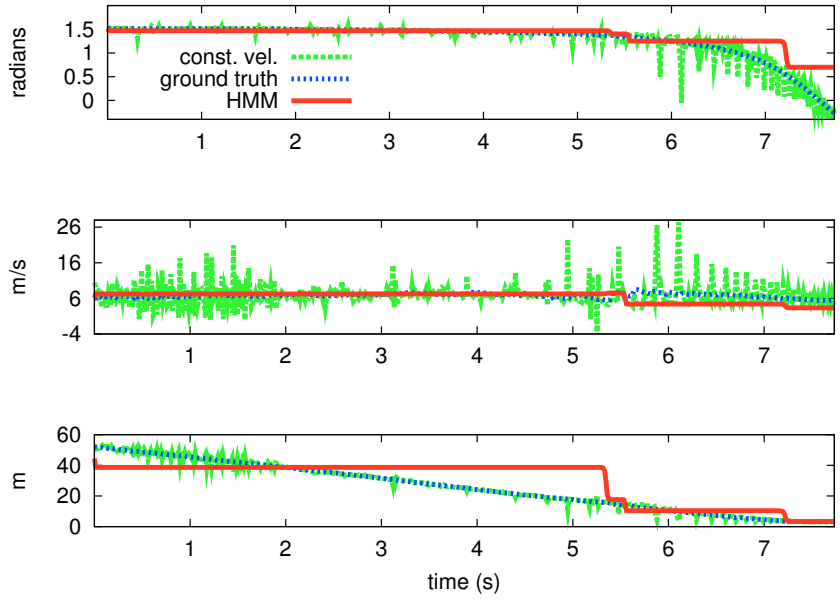


Figure 6.9: 1-second prediction for a sequence corresponding to a passing car according to a constant velocity model (*const. vel.*) and our learned HMM-based *passing* situation model (HMM).

the learned model and not only on the estimated state of the system. It must be noted, however, that the accuracy of the prediction depends strongly on the parameters of the model. For example, using a coarse discretization of the state space may lead to a model which can produce inaccurate predictions (see bottom plot).

6.4 Related Work

In the field of intelligent agents, *space* and *time* modeling has been approached using qualitative knowledge representation and reasoning, like, for example, using the *situation calculus* [McCarthy, 1963] or the *event calculus* [Kowalski and Sergot, 1986]. Despite the existence of these formalisms that simultaneously represent space and time, most modern approaches combine spatial and temporal calculi. Muller's [Muller, 1998] spatio-temporal theory, for example, is basically a first-order axiomatization of spatio-temporal entities based on RCC [Randell *et al.*, 1992]. Wolter and Zakharyaschev [Wolter and Zakharyaschev, 2000] combine RCC and propositional time logic [Manna and Pnueli, 1992]. Brandon *et al.* [Bennett *et al.*, 2002] also use propo-

sitional time logic but combine it with modal logic to produce a logic system that describes topological relationships that change over time. Gerevini and Nebel [Gerevini and Nebel, 2002] use Allen's interval calculus [Allen, 1983] to temporalize RCC.

All these approaches are mostly focused on the representation of and reasoning about spatio-temporal facts. Several authors have investigated the problem of extracting such facts from quantitative data to recognize relevant temporal configurations of those facts in an online fashion. Ghallab [Ghallab, 1996], for example, introduced the concept of a *chronicle* as a set of events and a set of temporal constraints between those events, where the events are symbolic representations obtained from sensors. In an online fashion, the recognition system processes these events and if they match the event model of a chronicle, then an instance of this chronicle is said to occur. Nagel [Nagel, 2001] describes a complete system capable of transforming sequences of video images into a natural text description of spatio-temporal developments. Nagel introduces the notion of a *generically describable situation* as a combination of a state-scheme and an action-scheme, where a state-scheme is a characterization of the state and an action-scheme specifies an action that could be performed based on the state-scheme.

Similar to the approaches of Ghallab and Nagel, our framework describes each relevant spatio-temporal configuration using an individual model that is evaluated as the state of the system changes. However, the approaches of Ghallab and Nagel do not explicitly deal with the inherent uncertainty in the observations and actions of a system. The hidden Markov model (see Rabiner [Rabiner, 1989]) is one of the most popular probabilistic models for representing sequences of states that have structure in time. Brand *et al.* [Brand *et al.*, 1997], for example, represent and classify sequences corresponding to T'ai Chi Ch'uan gestures using *coupled hidden Markov models*. Ghahramani *et al.* [Ghahramani and Jordan, 1997] use *factorial hidden Markov models* to model a collection of musical pieces of J.S. Bach. Landwehr [Landwehr, 2008] extracts different activities executed in parallel during the preparation of breakfast at home using *interleaved hidden Markov models*.

Like in the approaches mentioned above, we use hidden Markov models to describe distributions over meaningful state sequences. However, we additionally present a complete framework for real-time recognition of sequences that are consistent with the models. This approach is similar in spirit to the one presented by Bennewitz *et al.* [Bennewitz *et al.*, 2008] where a complete framework for recognizing gestures is presented. In contrast to the left-to-right models used by Bennewitz *et al.* we do not put restrictions on the state transitions of the model and we also discuss how recognized situations can be used to predict future developments in the scene.

6.5 Conclusions

In this chapter, we presented a general framework for modeling and recognizing situations. We take a model-based approach in which each situation type is described by an individual HMM that specifies the admissible state sequences corresponding to an instance of the given situation. For deciding between two competing situation models, we use the posterior odds, which provides a way of evaluating evidence in favor of a probabilistic model relative to an alternative one. We evaluated the approach experimentally using simulated and real data in the context of a driver assistant application with situations that typically occur in highway-like driving scenarios. The results demonstrate that our system is able to recognize and track multiple situation instances in parallel and to make sensible decisions between competing hypotheses. Additionally, we show that our models can be used for roughly predicting the position of the tracked cars.

Chapter 7

Regression-Based Situation Recognition for Traffic Scenarios

In Chapter 6 we presented a situation recognition framework where each situation type was described using an individual hidden Markov model (HMM). One of the drawbacks of this framework is that there is no straightforward approach to determine the states of the models. Furthermore, using a finite and usually small number of states to represent sequences of continuous state variables leads to discretizations of the sequences that may be inappropriate. In this chapter we present an alternative approach that overcomes this limitation by modeling the stereotypical state sequence using a regression function. In contrast to the approach presented in the previous chapter, using a regression function requires no artificial segmentation of the data. Furthermore, the regression-based approach is less susceptible to the effects of insufficient training data.

Like in the previous chapter, we consider a driver assistant application. The model for a given situation is learned from multiple state sequences or trajectories of the corresponding situation. The insight behind this approach is that, although instances of a given situation are in general different from each other, there is an inherent similitude that characterizes the situation. Figure 7.1 plots multiple trajectories of a *passing* situation together with the learned model. In our concrete example, a trajectory is described by the bearing, distance, and speed of the passing car relative to the car being passed. We can clearly see that the different trajectories are similar; they all present a distinctive form that characterizes the *passing* situation. The learned model for the *passing* situation is visualized in the figure by the thick lines and the filled curves, corresponding respectively to the mean and variance of the features.

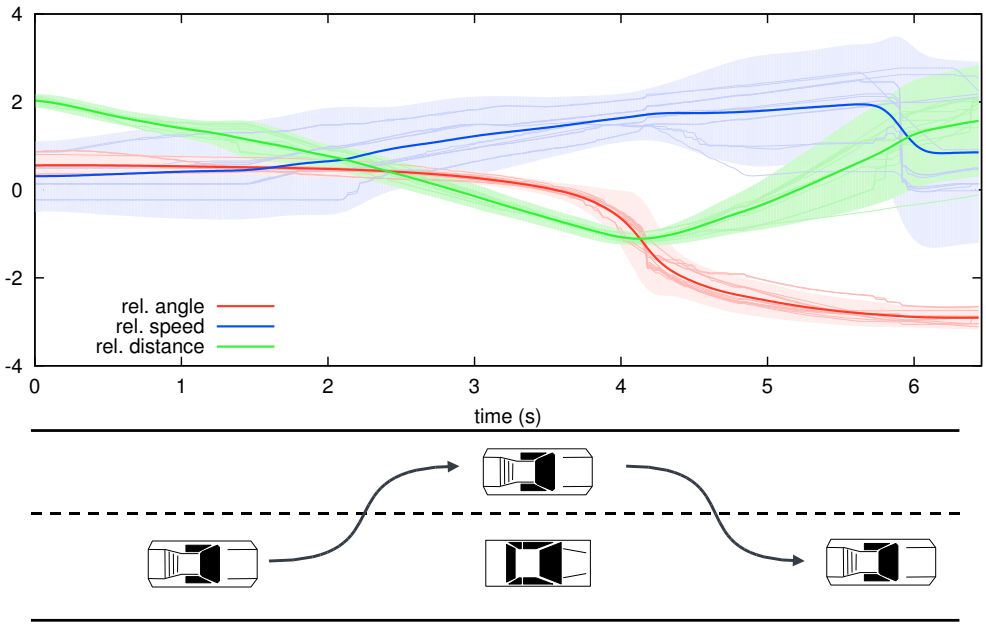


Figure 7.1: Regression model learned from trajectories corresponding to a *passing* situation. The training trajectories are visualized as thin lines and each color represents a different dimension (bearing, distance, and speed of the passing car relative to the car being passed) in state space. From these trajectories, a generalized model for the *passing* situation is learned using kernel regression, visualized by the thick line and the filled curve, corresponding to the mean and variance respectively.

We formulate the situation recognition problem using a dynamic Bayesian network (DBN) that represents, in a factorized way, the relevant aspects of the state of the system using random variables and conditional probability distributions between these variables. A high-level latent state variable that represents the current situation determines the dynamics of the lower-level state variables. The dynamics are described by a function that approximates the state of the system in time. This function is learned from a set of labeled training trajectories using kernel regression. Thus, each situation is modeled by an individual regression function that describes the characteristic dynamics of the situation. Trajectories can then be classified by evaluating their likelihood for the different models. This can be done even for trajectories that correspond only to a partial instance of a situation, allowing us to recognize situations online.

In this chapter we propose a practical approach for modeling and recognizing situations. We show how our framework can be used for learning models of typical situations in a vehicular traffic scenario. We also describe how situation instances can

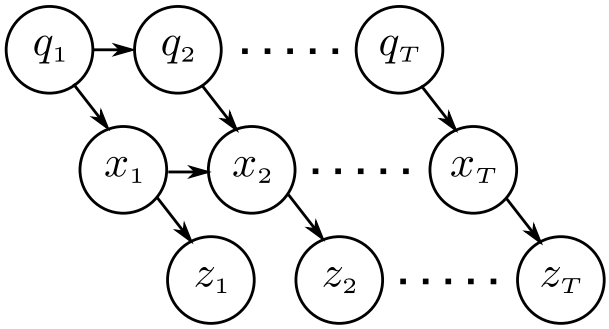


Figure 7.2: Structure of the dynamic Bayesian network used in our framework for learning the dynamics of the system in a given situation. For time t, x_t denotes the state of the system, z_t the observation of the state, and g_t is a high-level state variable that represents the current situation and determines the dynamics of the system.

be recognized while they are developing. Experimental results using real and simulated data show that our system can robustly recognize different traffic situations even for partially observed instances.

7.1 Modeling Situations using Regression

Each type of situation is associated with a particular system dynamics. The goal of our approach consists in learning the system dynamics corresponding to a situation from multiple training trajectories. Then, using the learned system dynamics, we can infer the most likely situation type for new trajectories.

We model the system using a DBN (see Figure 7.2). At each time step t, the random variable x_t represents the state of the system, z_t the observation of the state, and q_t is a high-level latent variable that describes the internal state of the current situation s. The observation z_t depends on the current state of the system x_t, which in turn depends on the previous state x_{t-1} and the current situation state q_t. The DBN encodes the following conditional probability distributions: the observation model $p(z_t \mid x_t)$ that represents the probability of observing z_t from state x_t, the state transition model $p(x_t \mid x_{t-1}, q_t)$ that corresponds to the state transition probability of the system and describes the dynamics of the system at time t for the given situation, and the situation-level state transition model $p(q_t \mid q_{t-1})$ that characterizes how the internal state of the situation evolves in time.

For a given situation s of length T, we learn a model $f_s = p(x_{1:T} \mid s)$ that describes the dynamics of the state of the system during the complete development of the situation. This model also describes, implcitly, how the internal state of the situation evolves in time. Using a set of M training trajectories $x^1_{1:T^1}, \ldots, x^M_{1:T^M}$, where each trajectory $x_{1:T^n} = \{x_1, \ldots, x_{T^n}\}$ consists of a sequence of states corresponding to the same situation, we apply kernel smoothing to obtain the model that best approximates all trajectories. As the training trajectories are in general of different lengths, we apply Dynamic Time Warping to standardize them before learning the situation model. Finally, we can also learn the prior over situations $p(s)$ by counting how many training trajectories belong to each different situation type.

Given an observation sequence $z_{1:t}$, the trajectory $x_{1:t}$ can be estimated using the Bayesian recursive state estimation scheme described in Chapter 2. We can then select the model that best fits the trajectory by computing the likelihood of each model given the data

$$p(s \mid x_{1:t}) \quad \propto \quad p(x_{1:t} \mid s)\, p(s)\,, \tag{7.1}$$

and selecting the model with the largest likelihood.

7.1.1 Kernel Smoothing

We treat the problem of learning a model for a situation s as a non-linear regression problem. The goal is to estimate the function f^s that approximates the state of the system x as a function of time. Consequently, f^s will represent the characteristic dynamics of the state for situation s. Given a set of training trajectories we learn the function that best approximates all of them.

Assuming that at every time step t the state x_t of the system is normally distributed over all training instances, we formulate $f^s(t)$ as

$$f^s(t) \quad = \quad \mathcal{N}(\mu^s(t),\, \Sigma^s(t))\,, \tag{7.2}$$

where $\mu^s(t)$ is a k-dimensional mean vector with the same dimensionality as x_t, and $\Sigma^s s(t)$ is the corresponding covariance matrix. Accordingly, the problem of estimating f^s can be stated as the problem of learning the mean $\mu^s(t)$ and covariance $\Sigma^s(t)$ of the normal distributions at each point in time t.

To estimate these parameters, we use kernel smoothing [Nadaraya, 1964] which is a non-parametric technique for approximating the density function of a random vari-

able from a set of sample instances. The idea is to approximate the value of the parameters as the weighted average of the neighboring sample points. These weights are given by a kernel function parameterized by a distance measure in the domain of the function. In our case, the weight of the samples depends on the temporal distance of the samples. Given a set of D trajectories corresponding to the same situation, each having a length of T, and assuming that the state dimensions are independent from each other, kernel smoothing estimates the mean $\mu_i(t')$ for dimension i at time t' as

$$\mu_i(t') = \frac{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})\, x_{d,t}^{i}}{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})}, \tag{7.3}$$

where $x_{d,t}^{i}$ is the value for the i-th dimension of the state of the system in trajectory d at time index t, and $K(u)$ is a Gaussian kernel with bandwidth h. This bandwidth determines how the influence of the neighboring samples decreases with the distance in time. The variance $\sigma_i^2(t')$ for dimension i at time t' is estimated as

$$\sigma_i^2(t') = \frac{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})\, (x_{d,t}^{i} - \mu_i(t'))^2}{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})}. \tag{7.4}$$

The result of applying kernel smoothing to the training data is a function $f(t)$ that describes, for each point in time t, the characteristic state of the system by the mean $\mu(t)$ and covariance $\Sigma(t)$.

7.1.2 Aligning Situation Instances

To be able to use the kernel density estimator method, the training trajectories must be of the same length. To handle temporal variations we apply Multi-Dimensional Dynamic Time Warping (MD-DTW) [ten Holt *et al.*, 2007], a variation of Dynamic Time Warping (DTW) [Sakoe and Chiba, 1978], to make all trajectories equally long. We select a reference trajectory from the training set, and all other trajectories are aligned to it. Kernel density estimation can then be directly applied on the aligned trajectories.

Dynamic Time Warping is a technique for aligning the time axis of two time-indexed sequences. The algorithm computes the minimum-cost alignment or warp between two series $x_{1:l_x}$ and $y_{1:l_y}$. Usually, the Euclidean distance is used as the distance measure $d(x_i, y_j)$ between the points in the sequences. The minimum-cost warp is then efficiently found using dynamic programming to compute the cumulative cost

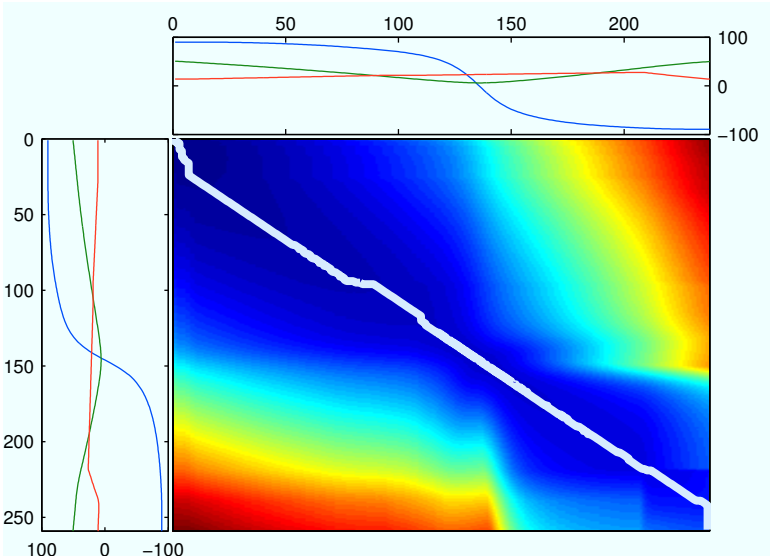


Figure 7.3: Dynamic Time Warping between two different trajectories. The figure visualizes the cost matrix induced by the recursive computation of the cumulative warping cost. The bold white line corresponds to the minimum-cost warping path.

$\gamma(i,j)$ corresponding to the minimum-cost warp of the partial sequences $x_{1:i}$ and $y_{1:j}$. The value of $\gamma(i,j)$ is recursively computed as

$$\gamma(i,j) = d(x_i, y_j) + min\{\gamma(i-1,j), \gamma(i,j-1), \gamma(i-1,j-1)\}\,. \tag{7.5}$$

The cost C_W of the minimum-cost warp is then given by the value of $\gamma(l_x, l_y)$ and the warp is constructed by tracing back from $\gamma(l_x, l_y)$ to $\gamma(1,1)$. Figure 7.3 illustrates the cost matrix induced by the recursive computation of Equation (7.5). Each entry i, j in the matrix corresponds to the value of the cumulative cost $\gamma(i,j)$ and the bold white line corresponds to the minimum-cost warp.

Multi-Dimensional Dynamic Time Warping is a generalization of DTW for multi-dimensional sequences where the distance between the points in the sequences corresponds to the n-dimensional Euclidean distance. To meaningfully compare different dimensions, each point x_i in the sequences is standardized as $x'_i = (x_i - \mu)\sigma^{-1}$, where μ and σ are the sample mean and standard deviation, before computing the distances. To make the alignments more robust, we compute, for each point in the sequence, an approximation of the derivative in each dimension as in [Keogh and Pazzani, 2001]. These are added to the state space and included in the distance between the points, effectively incorporating information about the shape of the trajectories being aligned.

7.2 Recognizing Situation Instances

Having trained a set of situation models, we want to select the one that best describes any given trajectory $x_{1:t}$. In other words, we want to select the model f^{s^*} for the situation s^* such that

$$s^* = \underset{s}{\operatorname{argmax}}\, p(f^s \mid x_{1:t})\,. \tag{7.6}$$

This requires the computation of the likelihood $p(x_{1:t} \mid f^s)$ of the trajectory $x_{1:t}$ for each situation model f^s. Given a regression model f^s corresponding to a situation s and a trajectory $x_{1:t}$, the likelihood of the trajectory given the model is computed as

$$p(x_{1:t} \mid s) = \prod_{t=1}^{t} p(x_t \mid f^s(t))\,, \tag{7.7}$$

where

$$p(x_t \mid f(t)) = \frac{1}{(2\pi)^{k/2}|\Sigma(t)|^{1/2}} e^{-\frac{1}{2}\left((x_t-\mu(t))^T\Sigma(t)(x_t-\mu(t))\right)}\,. \tag{7.8}$$

Equation (7.7) assumes that the states of the system are independent from each other. This is in general not the case, however, this approximation works well for our purposes and is easily computed.

Using the likelihood as a measure of the quality of a model, we can now select from a set of competing models, the one that produces the highest likelihood for a given trajectory. In this way, we can use our trained models for classifying trajectories. Note, however, that before computing the likelihood, the trajectory must have the same length as the model. This is achieved as explained in Section 7.1.2 by aligning the trajectory against the reference trajectory for the corresponding model.

7.2.1 Recognizing Partial Instances

The previous discussion about recognizing situations implicitly assumes that the start and the end of the trajectories that are being evaluated correspond, respectively, to the start and the end of the situation. For evaluating a trajectory that corresponds only to an incomplete instance of a situation, where the end of the instance does not correspond to the end of the situation, the MD-DTW approach as described in Section 7.1.2 can not be directly applied.

The MD-DTW algorithm finds the best global alignment between two sequences. However, for aligning an incomplete trajectory we need to find the best local alignment beginning at the common starting point. Given an incomplete trajectory $x_{1:t}$ and a reference trajectory $y_{1:T}$, the minimum-cost local warp is constructed by tracing back from $\gamma(t, j^*)$ to $\gamma(1, 1)$ where $j^* = \operatorname{argmin} \gamma(t, j)$ for $j = 1 \dots T$.

The warping algorithm can also be modified to include a scaling cost that penalizes the stretching or contraction of a sequence [Keogh and Pazzani, 2001]. In this way, we can alleviate the problem of over- or under-scaling a sequence. Once the incomplete trajectory is aligned, its likelihood can be computed directly using Equation (7.7). In this way, we are able to recognize situations as they are developing.

7.3 Experimental Evaluation

Our framework was tested in a vehicular traffic scenario using a simulated driving environment as well as real data. We considered three typical maneuvers as situations: *passing*, *aborted passing* and *following* as described in Chapter 6. The state of the system x_t was described by the bearing ψ, distance d, and speed v of the neighboring cars relative to the reference car. These features were sufficient to characterize the maneuvers, being also robust against variations in the different instances. We also evaluated our framework in the context of social robot navigation.

7.3.1 Recognizing Situation Instances

We first trained a model for each situation type using 30 trajectories generated in the simulation environment. Each trajectory started as soon as the neighboring car was closer than 50 meters and ended when the car was more than 50 meters away. As reference trajectory we selected one with average length. We then used MD-DTW to align the training trajectories and generated the corresponding model applying kernel smoothing. Figure 7.1 depicts some of the training trajectories and learned model for the *passing* situation. The likelihood for a set of 15 validation trajectories not used for training was computed to evaluate the classification performance of our models. For a better interpretation of the data, the average Mahalanobis distance over the length of the trajectory is used as fit error of a model. Table 7.3.1 presents the average fit error of the validation set for the learned models.

As can be seen in the table, the smallest error is obtained when the trajectories and the model correspond to the same situation type, which is the expected result. This

	Θ_{ap}			Θ_{f}			Θ_{p}		
	ψ	d	v	ψ	d	v	ψ	d	v
ap	**0.94** ± 0.42	**0.87** ± 0.37	**0.95** ± 0.34	6.49 ± 2.17	2.58 ± 1.31	4.22 ± 0.74	6.74 ± 0.51	2.53 ± 0.34	2.64 ± 0.43
f	2.09 ± 0.20	3.74 ± 3.17	3.04 ± 0.76	**0.80** ± 0.30	**2.43** ± 1.44	**2.48** ± 1.32	8.28 ± 0.22	9.08 ± 5.91	4.26 ± 0.91
p	17.27 ± 0.83	5.66 ± 0.82	18.08 ± 6.45	38.55 ± 3.28	11.46 ± 0.82	20.12 ± 7.94	**0.79** ± 0.43	**1.02** ± 0.15	**1.24** ± 0.35

Table 7.1: Average and standard deviation of the fit error for the models learned on artificial data.

experiment shows that our framework allows us to construct models that represent the characteristic dynamics of the maneuvers and could be used for recognition.

7.3.2 Recognizing Partial Situation Instances

To evaluate how our approach performs at recognizing trajectories that correspond to incomplete instances of a situation, we repeated the previous experiment using only the initial segment of the trajectories. Figure 7.4 plots the the average accuracy and standard deviation for each model as a function of the length of the trajectory segments. The classification results were evaluated using k-fold cross-validation with $k = 6$.

As can be seen in the figure, 80% of the length of the trajectory was enough to correctly classify all the segments. Even 50% of the length was enough to obtain reasonable classification results for all the models. Models corresponding to complex situations require more evidence to correctly classify a trajectory. This can be observed in the figure by the poor performance for proportionally short trajectories of the *passing* and *aborted passing* models. The *following* model, on the other hand, describes a relatively simple situation. As can be seen in the figure, the *following* model had a good classification accuracy even for very short trajectories.

7.3.3 Application in a Traffic Scenario

To evaluate the complete situation recognition approach in a real-time setting, we integrated our framework into a driving simulator (TORCS) [Espié and Guionneau, 1997] which features a simple 3D physics model and provides us with the absolute position and velocity of the vehicles at 50 hertz. At every time step, that is, every 0.02 seconds,

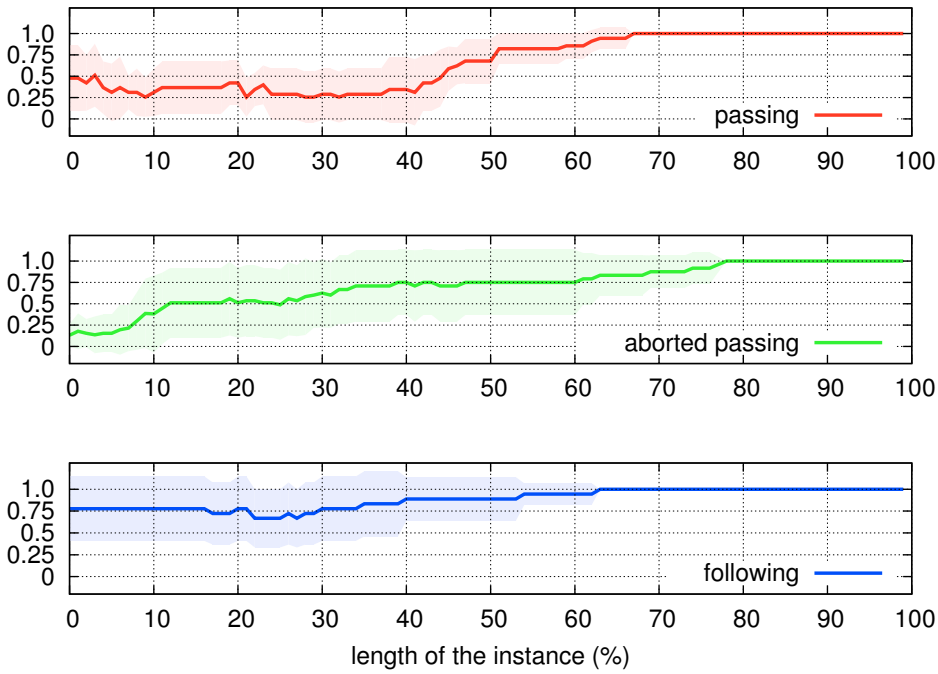


Figure 7.4: Classification accuracy of the models for partial trajectories. The figure shows the average accuracy and standard deviation for each model as a function of the length of the partial trajectories.

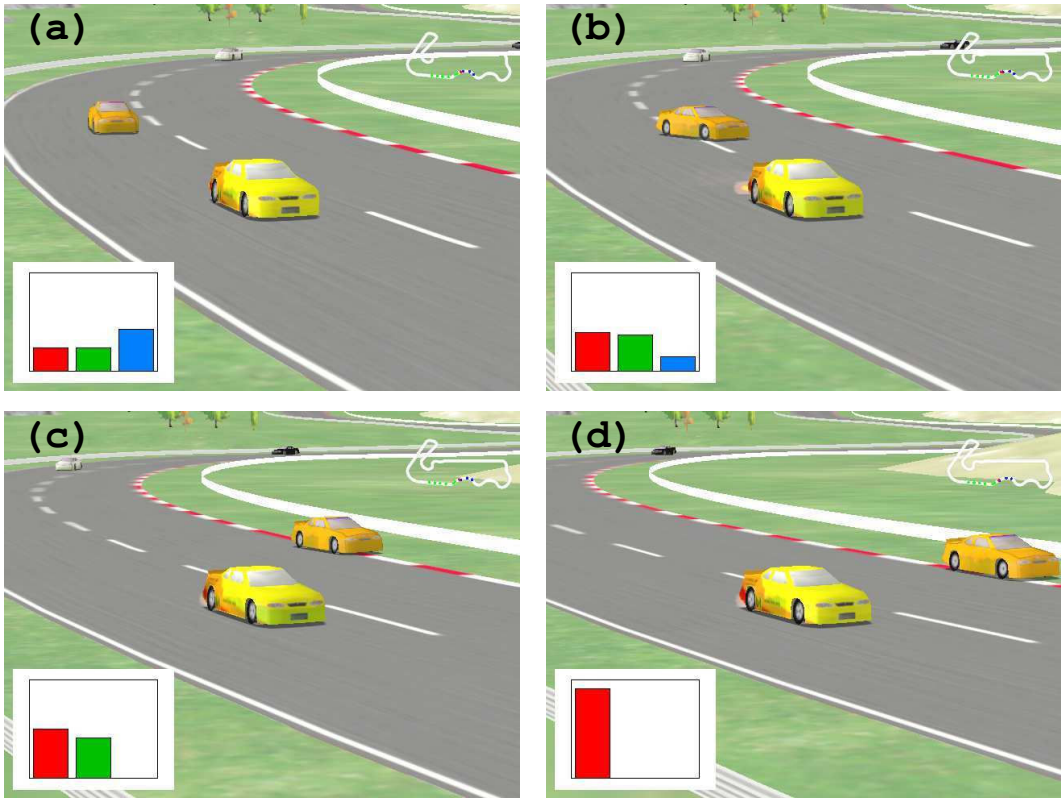


Figure 7.5: Screen shots of the simulation environment showing a car (orange) passing another one (yellow). The bars in the plots correspond to the likelihood of the trajectory of the passing car for the *passing* (red), *aborted passing* (green) , and *following* (blue) model. At first (**a**), the *following* situation is the most likely, but as the maneuver develops (**b**) - (**d**), the *passing* situations becomes more likely.

the state of the neighboring vehicles is computed. This means that, for each vehicle, the relative bearing, distance, and speed is estimated, and added to the corresponding trajectory. Then, after standardizing the state's values and computing the derivatives, the trajectories are aligned and the likelihood computed for every situation model.

Figure 7.5 shows a series of screen shots of the simulation environment in a two vehicle scenario together with the results of our situation recognition framework. The images show a car (orange) passing another one (yellow). The bars in the plots correspond to the normalized likelihood of the trajectory of the passing car for the *passing* (red), *aborted passing* (green), and *following* model (blue). As can be seen, at first (**a**), the *following* situation is the most likely, but as the maneuver develops (**b**) - (**d**), the *passing* situations becomes more likely. The framework was also tested in scenar-

	Θ_f			Θ_p		
	ψ	d	v	ψ	d	v
f	**0.84** $\pm$ 0.33	**0.87** $\pm$ 0.39	**0.84** $\pm$ 0.45	2.61 $\pm$ 0.49	7.03 $\pm$ 3.12	1.47 $\pm$ 0.51
p	8.23 $\pm$ 2.53	1.4 $\pm$ 0.19	2.59 $\pm$ 1.71	**0.73** $\pm$ 0.57	**0.84** $\pm$ 0.31	**0.81** $\pm$ 0.6

Table 7.2: Average and standard deviation of the model's fit error trained on real data.

ios with up to 9 vehicles (plus the reference one). The experiments showed that our approach can be used in such scenarios, where multiple vehicles are being simultaneously tracked, over extended periods of time.

The most time demanding step in the whole process is aligning the trajectories to the situation models. If the length of the reference trajectory of a model is N and the length of the trajectory being aligned is M, the time complexity for the alignment is $O(NM)$, that is, $O(N^2)$. However, since the cumulative cost $\gamma(i, j)$ can be incrementally computed as new states are added to the trajectories, the important factors in the performance of the approach are the number of situation models, the length of their reference trajectories, and number of neighboring vehicles.

The framework was also evaluated using real data. As described in Chapter 6, two SICK laser range scanners were mounted on a convertible as illustrated in Figure 6.5. Each laser has a field of view of 180 degrees and can detect objects as far as 80 meters with an angular resolution of 1 degree at 75 hertz. The data was gathered by driving over more than 50 kilometers on highways and state roads at velocities of up to 110 km/h. Note that in this work, we do not deal with the recognition and tracking of vehicles, and the trajectories were manually extracted from the data.

Due to the technical limitations of the sensors together with their arrangement, many situation instances could not be captured, or were captured only partially. From the gathered data, only 19 (10 *passing* and 9 *following*) useful trajectories could be extracted. Table 7.3.3 presents the fit error of the training set for the two trained models. As can be seen, the smallest error lies on the diagonal of the table where the sequence and the model correspond to the same maneuver type.

We also evaluated the classification accuracy of the models trained on real data against the trajectories generated in the simulation environment and vice versa. It must be noted that the trajectories from the real data where sampled at 75 hertz while the ones obtained from the simulator were sampled at 50 hertz. Also, because of the

	real training	artificial training	
f	-	0.77	real testing
	1.0	-	artificial testing
p	-	0.80	real testing
	0.57	-	artificial testing

Table 7.3: Classification accuracy of models trained and evaluated with real and artificial data.

configuration of the lasers, the trajectories corresponding to passing maneuvers where truncated when the passing vehicle was left of the reference one. Thus, the corresponding models only describe the maneuver until that point. When evaluating the trajectories generated in the simulation against the models trained with real data, we only considered the first half of the trajectories. This affected the classification accuracy, as shown in the previous experiment. Despite all these we obtained reasonable classification results as can be seen in Table 7.3.3

7.3.4 Recognizing Human Motion Behavior

We also evaluated our framework using real data in a social robot navigation context. We considered two different behaviors, *left* and *right passing* that typically occur when a persons encounters a robot in narrow space. We first trained a model for each behavior type using multiple trajectories acquired during an experiment where different human subjects where asked to walk along a corridor while a moving robot would meet them half the way, moving to their original starting point. Figure 7.6 depicts some of the training trajectories and learned model for the *right passing* behavior. Figure 7.7 plots the the average prediction accuracy and standard deviation for each model as a function of time using cross-validation. As can be seen in the figure an almost perfect prediction could be made with 1 second in advance. The poor prediction performance beyond 1.30 seconds is due to the nature of the modeled behaviors which are identical up to 75% of their duration.

7.4 Related Work

The approach presented in this chapter is similar, in essence, to the problem of programming a robot by human demonstration [Calinon and Billard, 2004], where a hu-

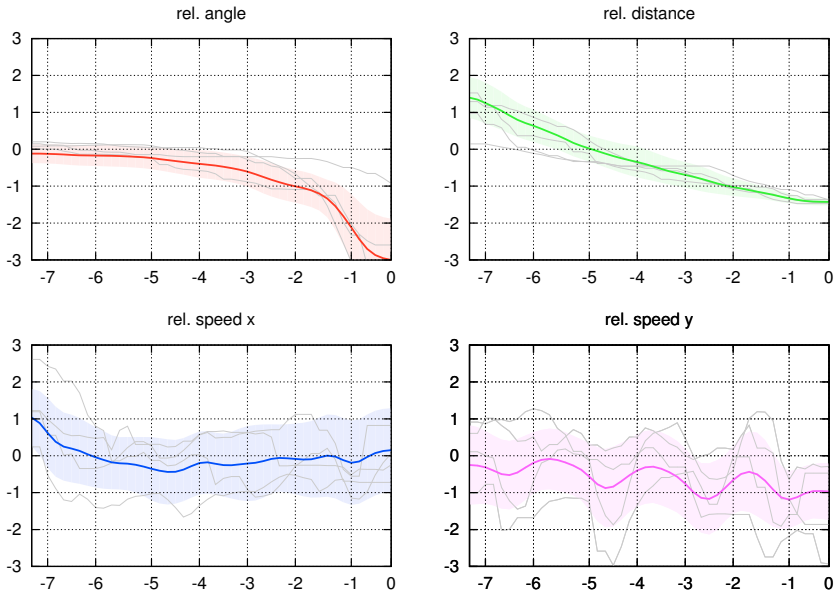


Figure 7.6: Regression model for the *right passing* behavior. Training trajectories are visualized as thin lines. Each plot represents a different dimension of the state space.

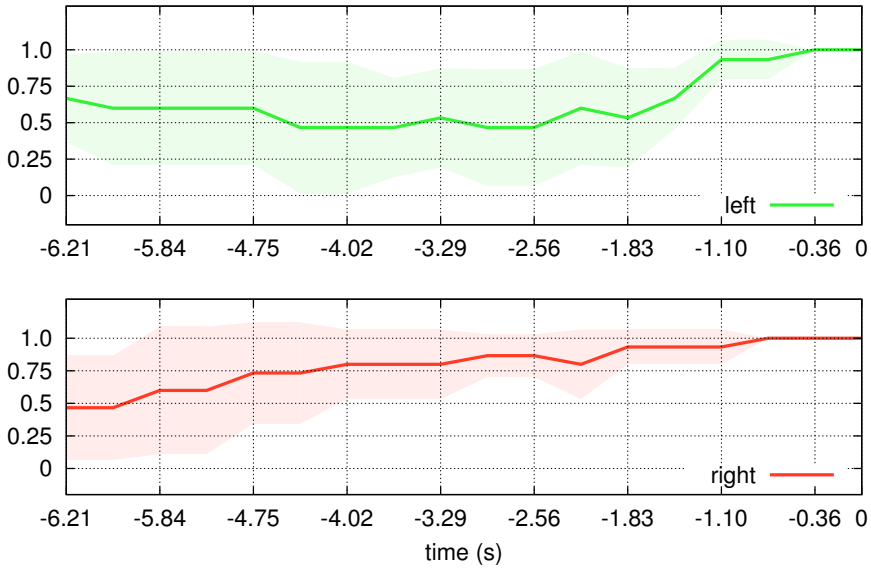


Figure 7.7: The figure shows the average accuracy and standard deviation for each model as a function of time, where $0\ s$ corresponds to the moment of passing.

man performs an action or task multiple times, and the robot must infer a generalized representation of the task. We build upon the work of Eppner *et al.* [Eppner *et al.*, 2009], where a framework for learning and reproducing tasks with a robotic manipulator is presented. The fundamental difference between our work and these approaches is that they focus on the learning and reproduction of the task, whereas our approach concentrates on the classification and online recognition of the tasks (situations in our case) using the learned models.

Our dynamic Bayesian network formulation of the system, where a high-level variable determines the dynamics of the lower-level states, is similar to the one made by the Switching Linear Dynamic Systems (SLDS) models. By switching between multiple linear dynamic models, an approximate description of the continuous non-linear dynamics of the system is obtained. Pavlović and Rehg [Pavlovic and Rehg, 2000] apply SLDS models to the problem of classifying human motion. Oh *et al.* [Oh *et al.*, 2006] present an extension of the SLDS framework that allows a parameterization of the duration model of standard SLDS models. They apply their approach for decoding the honeybee dance. In contrast to these SLDS-based techniques, our approach learns the non-linear dynamics of the system using individual regression models.

As described in Chapter 6, the HMM framework can be used to model and recognize state sequences that have structure in time [Brand *et al.*, 1997; Bennewitz *et al.*, 2008]. However, as already mentioned, using a finite (and usually small) number of states imposes a sometimes unnatural segmentation of the sequences to be modeled that can lead to poor recognition accuracies. The main advantage of our approach over the HMM-based approaches is that no artificial partitioning of the sequences is required.

7.5 Conclusions

In this chapter, we presented a general framework for modeling and recognizing situations. We take a model-based approach in which each situation type is described by an individual regression model that describes the characteristic dynamics of the system over time. We formalize the problem using a DBN and learn the characteristic dynamics of a situation from training instances. We then use the likelihood of the data as criterion for model selection and describe how to classify trajectories online. The approach was evaluated experimentally using real and simulated data in the context of a driver assistant application in traffic scenarios. The results show that our approach can robustly recognize different traffic situations even from partially observed instances.

Chapter 8

Conclusions

Mobile robots must, ultimately, be able to operate reliably in dynamic environments. This challenging task requires them to cope with the uncertainty inherent to the environment and the robotic system itself. With this work we contribute several novel solutions to a number of relevant problems associated with the long-term operation of mobile robots in dynamic environments.

To accomplish their tasks, mobile robots move in the environment, navigating from one place to another. Most existing robotic systems depend on a map of the environment provided beforehand, which is used as reference disregarding potential changes in the environment. Another popular approach is to rely on simultaneous localization and mapping techniques to build the map while navigating the environment. Both of these approaches are strongly based on the restrictive assumption that the environment is static and does not change. State-of-the-art systems are able to deal with certain changes in the environment, like moving people, for example. This, however, is usually achieved by implicitly modeling the dynamics of the environment as sensor noise and not as an inherent property of the environment.

In this thesis we addressed the problem of representing the dynamics of the environment in the model of the environment itself. In Chapter 3 we proposed an extended representation of the static environment that allows to model semi-static objects. These, we defined to be objects that change their location with relatively low frequency and therefore provide important information for estimating the pose of the robot. We described a localization approach that uses the measurements caused by semi-static objects to build local maps which temporarily extend the reference map of the environment. Our approach is a generalization of the well-known Monte-Carlo localization algorithm for static environments. Similarly to many existing approaches,

we assume that a reference map of the static parts of the environment is given beforehand. When the observations of the robot are consistent with this map, our approach corresponds exactly to the standard Monte-Carlo localization approach. However, we rely on the temporary local maps to keep track of the inconsistent observations caused by semi-static objects. Whenever the robot enters an area for which a temporary map already exists we try to use this map as reference for localization. Taking advantage of the measurements caused by semi-static objects is particularly important in large open spaces like parking lots or warehouses, where the static parts of the environment are few and usually occluded, and semi-static objects can provide valuable localization information. We showed experimentally that our model of the environment and localization framework can be used to robustly and accurately estimate the pose of the robot where standard state-of-the-art approaches fail.

Improving the localization performance of the robot is one of the benefits of explicitly representing the changes in the environment. Another advantage is a better understanding of the environment. This allows the robot to make more informed decisions, for example, when planning a trajectory to a goal location. In Chapter 4 we considered the problem of modeling a mobile robot's environment taking the dynamics of the environment explicitly into account. We presented a probabilistic model that represents the occupancy of the space and characterizes how this occupancy changes over time. Concretely, we described the environment as a spatial grid and used hidden Markov models to represent the belief about the occupancy state and state transition probabilities of each grid cell. We described how our representation can be updated as new observations become available, and presented techniques to estimate, both offline and online, the parameters of the model from observed data. The experimental results showed that this model can represent dynamic environments better than standard occupancy grids. We also demonstrated that our model can be used to improve the path planning performance of the robot.

An alternative approach to improve robot navigation in dynamic environments is to attach landmarks to static parts in the surroundings. This strategy can also help navigation in environments that are structurally symmetrical or have only few recognizable features so that the pose of the robot cannot be uniquely determined. In Chapter 5 we presented an approach to compute a configuration of indistinguishable landmarks that maximizes the *uniqueness* in the environment by decreasing the overall ambiguity. Clearly, the location of the landmarks has to be chosen carefully as not to introduce additional symmetries and ambiguities. We introduced a measure for the *uniqueness* of a robot pose based on sensor data that explicitly considers a probabilistic observation

model. Based on this measure, we described a landmark selection algorithm that incrementally selects landmark locations and greedily maximizes the average uniqueness in the environment. Furthermore, we described a concrete instantiation of the landmark placement problem for a grid-based representation of the environment. We evaluated the proposed approach in various simulated and real world environments. The results demonstrate that our landmark selection technique improves the localization performance of the robot and clearly outperforms other landmark selection approaches.

In addition to estimating the current state of the environment, a robotic system must also be able to interpret this information to act intelligently. This kind of reasoning involves the prediction of future states to make informed decisions, avoid risks, and, in general, improve the robot's performance when accomplishing its task. In the final two technical chapters of this thesis we addressed the problem of modeling and recognition of situations. We focused on an autonomous car application and considered situations that typically occur in highway-like driving settings, like, for example, passing, following, and aborted passing situations. We reason about *situation types* and *situation instances*. Concretely, we define a situation instance as a sequence of states that has some meaningful interpretation or, alternatively, as a meaningful spatio-temporal configuration of the system. A situation type is then defined as the set of all situation instances that are grouped together under the same interpretation. A situation type can be thought of as an equivalence class of state sequences.

In Chapter 6 we presented a framework for modeling and online-recognition of situations instances. We described each situation type using an individual hidden Markov model (HMM). A situation HMM specifies the admissible state sequences which correspond to an instance of the given situation type. We computed the parameters for each situation HMM individually from training instances using the Baum-Welch algorithm. To recognize occurring situations, we evaluated the likelihood of the current state sequence with respect to each situation model and computed the likelihood ratio for deciding between two competing situation models. This ratio was used to evaluate evidence in favor of a probabilistic model relative to an alternative one. Using real and simulated data we showed experimentally that our system can recognize and track multiple situation instances in parallel and make sensible decisions between competing situation types. Additionally, we demonstrated that our models can be used for making rough predictions about future states of the system.

One of the drawbacks of using HMMs to describe the stereotypical state sequence corresponding to a situation type is that there is no straightforward approach to determine the states in the model. Furthermore, representing trajectories using a finite and

usually small number of states leads to a discretization of the state sequence that may be inappropriate. In Chapter 7 we described an alternative approach where we modeled the stereotypical state sequence using a regression function that approximates the state of the system at each point in time. We used kernel regression to learn these functions from sets of labeled training sequences. To align sequences of different lengths, we applied Dynamic Time Warping before learning the underlying situation model. Each situation type was then characterized by an individual regression function that describes its characteristic dynamics. Similar to the approach described in Chapter 6, we evaluated the likelihood of the current state sequence with respect to each situation model to recognize occurring situations. We implemented and evaluated our situation recognition framework using simulated and real data. The results showed that our approach can robustly keep track of multiple situation type hypotheses even if the situation instance has only been partially observed.

All the approaches presented in this thesis were implemented and tested is simulation as well as with different mobile robotic platforms. In particular we used a MobileRobots Pioneer P3-DX and Powerbot both equipped with SICK LMS laser range finders. Furthermore, we gathered real vehicular traffic data for the situation modeling and recognition chapters using two SICK laser range finders mounted on a convertible. For the simulations we relied on two simulation environments: the Carnegie Mellon Robot Navigation Toolkit (CARMEN) and the TORCS driving simulator. Our experiments showed that explicitly considering the dynamics of the environment can considerably improve the performance of a robotic system.

As mentioned above, this thesis contributes to the field of robotics by providing several novel solutions to a number of relevant problems associated to the operation of mobile robots in dynamic environments. The issues and challenges addressed are fundamental for the ultimate goal of long-term autonomous operation of a mobile robot. We developed novel probabilistic models that explicitly represent the dynamics of the environment and presented efficient methods for probabilistic inference that enable a robotic system to reason about these dynamics. Concretely, the main contributions of this thesis are:

- A robot localization framework for semi-static environments that extends the reference model of the environment to explicitly represent semi-static objects.
- A probabilistic model for dynamic environments that explicitly characterizes the dynamic of the occupancy in the environment.
- An approach to artificial landmark placement to improve robot localization in

dynamic and inherently ambiguous environments.

- A framework to model and recognize situations in vehicular traffic scenarios.

Long-term operation of mobile robots outside controlled environments has been gaining increasing interest both in research as well as in the industry. This indicates that the fundamental functionality and technology is mature enough to face this big challenge. We believe that the approaches presented in this thesis constitute a relevant contribution to the ultimate goal of long-term autonomous operation of mobile robots.

8.1 Future Work

In this thesis we argued that mobile robots can profit from an explicit characterization of the dynamic in the environment. In spite of the promising results, a number of relevant aspects remain to be addressed or improved. The rest of this section points out several potential directions for future research.

In Chapter 4, we proposed dynamic occupancy grids as spatial description of the environment that explicitly describes the dynamics in a place-dependent way. In addition to the spatial dependency, temporal aspects can also have a strong influence on the dynamics of the environment. For example, the dynamics of an office environment clearly depends on the time of day. A mobile robot could benefit from a representation that jointly characterizes the spatial and temporal dependencies of the environment's dynamics. The complexity of the model, however, could be a limiting factor. Furthermore, as the number of parameters in the model increases, an appropriate treatment of the usually insufficient training data becomes critical.

In dynamic environments, a robot should be able to continually update its internal representation of the environment to accommodate for changes, that is, perform life-long SLAM. This could be implemented, for example, integrating our proposed dynamic occupancy grids with a Rao-Blackwellized particle filter framework. Each particle would update its associated map using the online approach described in Chapter 4. The weights of the particles could then be computed using an appropriate probabilistic sensor model. This model would implicitly address the general problem of robustly distinguishing between actual changes in the environment, sensor noise and localization errors.

In the context of situation modeling, our proposed approaches learned the stereotypical state sequence for a situation type at a rather low level of abstraction. We considered continuous features like relative distance, velocity and bearing. At this level

of abstraction, the type of situations that can be represented is relatively limited. For example, cyclic situations, or situations involving a varying number of features cannot be characterized in a precise way. Although modeling at a higher level of abstraction would allow to represent a larger number of situations, such symbolic approaches are not well suited for reasoning in continuous domains. Future research in the context of spatio-temporal modeling should consider representations that are accurate enough for dealing with low level features and expressive enough to describe high-level features of the situations.

Mobile robots do not only need to operate autonomously, they also need to do this over extended periods of time. We believe that reliable long-term autonomous operation will significantly expand the application domains of mobile robots. This, in turn, will boost further research in long-term autonomous operation and pave the way for novel applications. With this thesis we contribute in part to the realization of this long-term goal.

Bibliography

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Andrade-Cetto and Sanfeliu, 2002] Juan Andrade-Cetto and Alberto Sanfeliu. Concurrent map building and localization on indoor dynamic environments. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3), 2002.

[Andrieu *et al.*, 2003] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[Anguelov *et al.*, 2002] Dragomir Anguelov, Rahul Biswas, Daphne Koller, Benson Limketkai, and Sebastian Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of the Conference on Uncertainty in AI (UAI)*, 2002.

[Bennett *et al.*, 2002] Brandon Bennett, Anthony G. Cohn, Frank Wolter, and Michael Zakharyaschev. Multi-dimensional modal logic as a framework for spatio-temporal reasoning. *Applied Intelligence*, 17(3):239–251, 2002.

[Bennewitz *et al.*, 2008] Maren Bennewitz, Tobias Axenbeck, Sven Behnke, and Wolfram Burgard. Robust recognition of complex gestures for natural human-robot interaction. In *Proc. of the Workshop on Interactive Robot Learning at Robotics: Science and Systems Conference (RSS)*, 2008.

[Biber and Duckett, 2005] Peter Biber and Tom Duckett. Dynamic maps for long-term operation of mobile service robots. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[Brand *et al.*, 1997] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. In *CVPR*, pages 994–999, 1997.

[Brechtel *et al.*, 2010] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

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

[Burgard *et al.*, 2000] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.

[Burgard *et al.*, 2009] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kümmerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D. Tardós. A comparison of SLAM algorithms based on a graph of relations. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[Calinon and Billard, 2004] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. 2004.

[Cocora *et al.*, 2006] Alexandru Cocora, Kersting Kersting, Christian Plagemann, Wolfram Burgard, and Luc De Raedt. Learning relational navigation policies. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[Dellaert *et al.*, 1999] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[Eppner *et al.*, 2009] Clemens Eppner, Jrgen Sturm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Imitation learning with generalized task descriptions. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[Espié and Guionneau, 1997] Eric Espié and Christophe Guionneau. TORCS - The Open Racing Car Simulator, http://torcs.sourceforge.net/, 1997.

[Estrada *et al.*, 2005] Carlos Estrada, José Neira, and Juan D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4), 2005.

[Fox *et al.*, 1999] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[Gerevini and Nebel, 2002] Alfonso Gerevini and Bernhard Nebel. Qualitative spatio-temporal reasoning with RCC-8 and Allen's interval calculus: Computational complexity. In *ECAI*, pages 312–316, 2002.

[Ghahramani and Jordan, 1997] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2-3):245–273, 1997.

[Ghallab, 1996] Malik Ghallab. On chronicles: Representation, on-line recognition and learning. In *KR*, pages 597–606, 1996.

[Grisetti *et al.*, 2005] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[Grisetti *et al.*, 2010] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[Gutmann and Konolige, 2000] Jens-Steffen Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Proc. of the Conf. on Intelligent Robots and Applications (CIRA)*, 2000.

[Hähnel *et al.*, 2002] Dirk Hähnel, Dirk Schulz, and Wolfram Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[Hähnel *et al.*, 2003a] Dirk Hähnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[Hähnel *et al.*, 2003b] Dirk Hähnel, Rudolf Triebel, Wolfram Burgard, and Sebastian Thrun. Map building with mobile robots in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[Hähnel, 2004] Dirk Hähnel. *Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, December 2004.

[Howard and Roy, 2003] Andrew Howard and Nicholas Roy. The robotics data set repository (Radish), 2003.

[Howard *et al.*, 2001] Andrew Howard, Maja J. Mataric, and Gaurav S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[Keogh and Pazzani, 2001] Eamonn J. Keogh and Michael J. Pazzani. Derivative dynamic time warping. In *In Proceedings of First SIAM International Conference on Data Mining (SDM'2001)*, 2001.

[Konolige and Bowman, 2009] Kurt Konolige and James Bowman. Towards lifelong visual maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[Kowalski and Sergot, 1986] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

[Landwehr, 2008] Niels Landwehr. Modeling interleaved hidden processes. In *Machine Learning, Proc. of the Twenty-Fifth Int. Conf. (ICML 2008)*, pages 520–527, Helsinki, Finland, 2008.

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

[Lerner *et al.*, 2006] Ronen Lerner, Ehud Rivlin, and Ilan Shimshoni. Landmark selection for task-oriented navigation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[Levin *et al.*, 2006] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[Luber *et al.*, 2009] Matthias Luber, Gian Diego Tipaldi, and Kai O. Arras. Place-dependent people tracking. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2009.

[Manna and Pnueli, 1992] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems.* Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[McCarthy, 1963] John McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963.

[Meyer-Delius *et al.*, 2007] Daniel Meyer-Delius, Christian Plagemann, Georg von Wichert, Wendelin Feiten, Gisbert Lawitzky, and Wolfram Burgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proc. of the 31th Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications*, 2007.

[Mongillo and Deneve, 2008] Gianluigi Mongillo and Sophie Deneve. Online learning with hidden Markov models. *Neural Computation*, 20:1706–1716, 2008.

[Montemerlo *et al.*, 2002] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[Moravec and Elfes, 1985] Hans P. Moravec and Alberto E. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1985.

[Muller, 1998] Philippe Muller. A qualitative theory of motion based on spatio-temporal primitives. In *KR*, pages 131–143, 1998.

[Nadaraya, 1964] E. A. Nadaraya. On estimating regression. *Theory of Probability and Its Application*, 9:141–142, 1964.

[Nagel, 2001] Hans-Hellmut Nagel. Towards a cognitive vision system. Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik der Universität Karlsruhe (TH), 2001. Electronic publication.

[Oh *et al.*, 2006] Sang Min Oh, James M. Rehg, and Frank Dellaert. Parameterized duration mmodeling for switching linear dynamic systems. In *CVPR (2)*, pages 1694–1700, 2006.

[Olson, 2009] Edwin Olson. Real-time correlative scan matching. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[Pavlovic and Rehg, 2000] Vladimir Pavlovic and James M. Rehg. Impact of dynamic model learning on classification of human motion. In *CVPR*, pages 1788–1795, 2000.

[Petrovskaya and Ng, 2007] Anna Petrovskaya and Andrew Y. Ng. Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77 (2), pages 257–286, 1989.

[Rafflin and Fournier, 1996] Catherine Rafflin and Alain Fournier. Learning with a friendly interactive robot for service tasks in hospital environments. *Autonomous Robots*, 3:399–414, 1996.

[Randell *et al.*, 1992] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *KR*, pages 165–176, 1992.

[Sakoe and Chiba, 1978] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, Feb 1978.

[Sala *et al.*, 2004] Pablo L. Sala, Robert Sim, Ali Shokoufandeh, and Sven J. Dickinson. Landmark selection for vision-based navigation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[Salas and Gordillo, 1998] Joaquín Salas and José Luis Gordillo. Placing artificial visual landmarks in a mobile robot workspace. In *Proc. of the Ibero-American Conf. on Artificial Intelligence (IBERAMIA)*, 1998.

[Sinriech and Shoval, 2000] David Sinriech and Shraga Shoval. Landmark configuration for absolute positioning of autonomous vehicles. *IIE Transactions*, 32:613–624, 2000.

[Stachniss and Burgard, 2005] Cyrill Stachniss and Wolfram Burgard. Mobile robot mapping and localization in non-static environments. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2005.

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

[Strasdat *et al.*, 2009] Hauke Strasdat, Cyrill Stachniss, and Wolfram Burgard. Which landmark is useful? Learning selection policies for navigation in unknown environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[Sutherland and Thompson, 1993] Karen T. Sutherland and William B. Thompson. Inexact navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1993.

[ten Holt *et al.*, 2007] Gineke A. ten Holt, Marcel J. T. Reinders, and Emile A. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *In Proc. of the conference of the Advanced School for Computing and Imaging (ASCI 2007)*, 2007.

[Thrun *et al.*, 2005] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT-Press, 2005.

[Thrun, 1998] Sebastian Thrun. Finding landmarks for mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

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

[Thrun, 2002] Sebastian Thrun. Robotic mapping: A survey. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[Wang and Thorpe, 2002] Chieh-Chih Wang and Charles Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[Williams *et al.*, 2002] Stefan B. Williams, Gamini Dissanayake, and Hugh F. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[Wolf and Sukhatme, 2005] Denis F. Wolf and Gaurav S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005.

[Wolter and Zakharyaschev, 2000] Frank Wolter and Michael Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR*, pages 3–14, 2000.

[Zhang *et al.*, 2005] Sen Zhang, Lihua Xie, and M.D. Adams. Entropy based feature selection scheme for real time simultaneous localization and map building. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.