Landmark Placement for Mobile Robot Navigation

Maximilian Beinhofer

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Landmark Placement for Mobile Robot Navigation

Maximilian Beinhofer

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

Dekan	Prof. Dr. Yiannos Manoli
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Andreas Krause
	Eidgenössische Technische Hochschule Zürich
Tag der Disputation	29.08.2014

Zusammenfassung

Die Fähigkeit, zuverlässig und präzise zu navigieren, ist eine der wichtigsten Grundvoraussetzungen für mobile Roboter, um Aufgaben wie Andockmanöver, Warentransport, oder systematische Reinigung erfolgreich selbstständig durchzuführen. Selbstständig navigierende Roboter weichen üblicherweise durch Ungenauigkeiten in der Bewegungsausführung von ihrem vorgesehenen Bewegungspfad ab. Um diese Abweichung zu schätzen und entsprechend gegensteuern zu können, sind Roboter in der Regel mit Sensoren ausgestattet, mit denen sie ihre Umgebung wahrnehmen. Typische Umgebungen, in denen Roboter agieren, wie zum Beispiel Industriehallen, enthalten oftmals verschiedene Bereiche, die sich stark ähneln, oder Bereiche, die sich regelmäßig verändern. Damit sich Roboter auch in solchen Bereichen zurechtfinden können, werden in der Praxis oft künstliche Landmarken in der Umgebung angebracht. Beispiele für künstliche Landmarken sind reflektierende Markierungen, Radarbaken, oder Barcodes, die von den Sensoren des Roboters wahrgenommen werden können. Die präzise Platzierung von künstlichen Landmarken ist in der Regel mit hohen Kosten verbunden, und auch die Landmarken selbst können, je nach Typ, teuer sein. Deshalb ist es vorteilhaft, so wenige Landmarken wie möglich zu platzieren. Um trotzdem die gewünschte Genauigkeit im Verhalten des Roboters zu gewährleisten, ist eine sorgfältige Auswahl der Landmarkenpositionen entscheidend.

In dieser Arbeit behandeln wir das Problem, eindeutig identifizierbare künstliche Landmarken so zu platzieren, dass sie für Roboter, die wiederholt gleichartige Navigationsaufgaben ausführen, optimal geeignet sind. In industriellen Anwendungen ist es zum Beispiel üblich, dass Roboter den gleichen vorgegebenen Bewegungspfad viele Male hintereinander abfahren. Für diese Art von Anwendung stellen wir automatische Berechnungsverfahren vor, die eine möglichst kleine Menge von Landmarkenpositionen finden, um die gewünschte Genauigkeit im Verhalten des Roboters zu gewährleisten. Um die Güte einer Landmarkenplatzierung zu beschreiben, verwenden wir zwei verschiedene Werte. Zum einen betrachten wir die Genauigkeit, mit der der Roboter seine eigene Pose in der Umgebung schätzen kann. Falls der Roboter allerdings seine Navigationsentscheidungen anhand von dieser Posenschätzung nach bekannten, vorgegebenen Regeln trifft, dann können wir auch die Navigationsgenauigkeit selbst, also die tatsächliche Nähe des Roboters zu seinem vergegebenen Bewegungspfad, verwenden. Da die konkreten Bewegungs- und Messfehler des Roboters und die daraus resultierenden Navigationsent-

scheidungen zum Zeitpunkt der Landmarkenplatzierung noch unbekannt sind, benutzen wir zur Landmarkenplatzierung die erwartete Lokalisierungsgenauigkeit und die erwartete Navigationsgenauigkeit anstatt der konkreten Werte dieser Funktionen.

Diese Erwartungswerte effizient abzuschätzen, ist eine der zwei großen Herausforderungen beim Platzieren von Landmarken. Die andere ist die kombinatorische Natur des Problems, die beste Teilmenge aus der Menge aller möglichen Landmarkenpositionen auszuwählen. In dieser Arbeit präsentieren wir Lösungen zum Umgang mit beiden Herausforderungen. Um mit dem – im Allgemeinen $\mathbf{NP}$-schwierigen – kombinatorischen Landmarkenplatzierungsproblem umzugehen, verwenden wir Methoden aus dem Gebiet der submodularen Funktionsoptimierung. Diese Methoden bestimmen auf effiziente Art approximative Lösungen für das Platzierungsproblem und garantieren dabei eine hohe Approximationsgenauigkeit. Um die erwartete Lokalisierungs- und Navigationsgenauigkeit des Roboters effizient zu schätzen, verwenden wir eine Linearisierung der Modelle des gesamten Navigationszyklus des Roboters, bestehend aus Bewegungsausführung, Observation von Landmarken, Lokalisierung und Auswahl der nächsten gewünschten Bewegung. Wir führen eine neue Methode ein, um in dem linearisierten Navigationszyklus die erwartete Navigationsgenauigkeit effizient rekursiv zu berechnen. Die aus der Effizienz dieser Berechnung resultierende Effizienz unserer Landmarkenplatzierungsansätze macht es möglich, mit unseren Ansätzen auch für sehr große Probleminstanzen, also lange Bewegungspfade, Landmarken zu platzieren. Des Weiteren führen wir auch ein Verfahren ein, um bei der Platzierung von Landmarken mit Situationen umgehen zu können, in denen eine gewisse Anzahl der platzierten Landmarken auf unvorhersehbare Weise vom Roboter aus nicht observierbar ist. Solche Situationen treten zum Beispiel für Arten von Landmarken auf, die mit der Zeit verschlissen werden, oder wenn der Roboter sich die Umgebung mit anderen Fahrzeugen teilt, die seine Sichtachse auf Landmarken blockieren können.

Für Aufgaben wie zum Beispiel Retten von Vermissten in Katastrophengebieten ist es allerdings nicht möglich, die Umgebung, in der der Roboter agieren soll, im Vorfeld mit Landmarken zu bestücken. Zudem hat der Roboter in solchen Anwendungsszenarien typischerweise keine genaue Karte von seiner Umgebung. Um zuverlässig navigieren zu können, muss der Roboter also anhand der Observationen seiner Sensoren eine Karte der Umgebung schätzen und sich gleichzeitig in dieser Karte lokalisieren. Im Zusammenhang mit dieser Aufgabe betrachten wir das fundamentale Problem der Datenassoziation. Datenassoziation bezeichnet das Problem, zu entscheiden, ob zwei verschiedene Observationen von natürlichen Landmarken dieselbe Landmarke oder zwei unterschiedliche Landmarken zeigen. Wenn der Roboter mit einem Mechanismus ausgestattet ist, mit dem er selbstständig eindeutig unterscheidbare künstliche Landmarken in seiner Umgebung ausbringen kann, dann kann er die Observationen dieser Landmarken später verwenden, um die Schätzung seiner relativen Bewegung zwischen zwei solchen Observationen und

damit auch die Schätzung der Datenassoziationen zwischen den natürlichen Landmarken zu verbessern. Wir stellen eine neuartige Methode vor, mit der eine Strategie zur Ausbringung künstlicher Landmarken generiert werden kann, die das Ziel verfolgt, die Schätzung der Datenassoziationen zu erleichtern. Unsere Methode verwendet verstärkendes Lernen mit Monte-Carlo-Verfahren um eine optimale Strategie aus einer Reihe von Beispielepisoden zu lernen.

Zusammenfassend beinhaltet diese Arbeit die folgenden Beiträge:

- Nachdem in den Kapiteln 2 und 3 eine Übersicht über verwandte Arbeiten und verwendete Standardmethoden gegeben wird, stellen wir in Kapitel 4 ein neuartiges Verfahren zur Landmarkenplatzierung vor. Dieses zielt darauf ab, die erwartete Lokalisierungsgenauigkeit eines Roboters entlang eines vorgegebenen Bewegungspfades zu maximieren. Mit Hilfe von Ergebnissen aus der submodularen Funktionsoptimierung leiten wir eine garantierte untere Schranke für die Approximationsgenauigkeit unseres Verfahrens her. Um die erwartete Lokalisierungsgenauigkeit zu schätzen, verwenden wir in diesem Ansatz Monte-Carlo-Simulationen.
- In Kapitel 5 führen wir ein Verfahren zur Schätzung der erwarteten Navigationsgenauigkeit ein, das signifikant effizienter ist als Monte-Carlo-Simulationen. Dieses Verfahren linearisiert den gesamten Navigationszyklus des Roboters und verwendet eine effiziente Methode, die erwartete Navigationsgenauigkeit anhand der linearisierten Modelle rekursiv zu berechnen.
- In Kapitel 6 stellen wir ein Verfahren zur Landmarkenplatzierung vor, das die erwartete Navigationsgenauigkeit des Roboters optimiert und in den auftretenden Berechnungen die effiziente Methode aus Kapitel 5 verwendet. Die resultierende Effizienz in der Landmarkenplatzierung erlaubt es uns, mit diesem Ansatz auch sehr große Probleminstanzen zu behandeln.
- Dieses Verfahren modifizieren und erweitern wir in Kapitel 7, um auch mit Situationen umgehen zu können, in denen dem Roboter die Sichtachse zu einer gewissen Anzahl von platzierten Landmarken auf unvorhersehbare Weise langfristig versperrt ist. Auch hier verwenden wir Verfahren aus der submodularen Funktionsoptimierung.
- In Kapitel 8 präsentieren wir einen Vergleich der drei oben angeführten Landmarkenplatzierungsmethoden.
- Zuletzt führen wir in Kapitel 9 einen Ansatz ein, mit dem eine Strategie zur Ausbringung künstlicher Landmarken generiert werden kann, die das Ziel verfolgt, dem Roboter die Schätzung von Datenassoziationen zu erleichtern. Wir verwenden dabei verstärkendes Lernen, um eine optimale Strategie aus einer Reihe von Beispielepisoden zu lernen.

Alle in dieser Arbeit vorgestellten Verfahren wurden in ausführlichen Experimenten, sowohl in Simulation als auch mit echten Robotern, getestet. Die Ergebnisse dieser

Experimente belegen, dass die von uns verwendeten Modelle echte Roboter realistisch widerspiegeln, und dass die Landmarkenmengen, die mit unseren Methoden platziert wurden, zu höheren Lokalisierungs-, Navigations- und Datenassoziationsgenauigkeiten führen als Landmarkenmengen, die mit anderen Verfahren platziert wurden. Wir sind der Ansicht, dass die in dieser Arbeit vorgestellten Methoden ein nützliches Werkzeug sind, um die Sicherheit und die Zuverlässigkeit mobiler Roboter in der Praxis, insbesondere in industriellen Anwendungen, zu gewährleisten.

Abstract

Being able to navigate accurately is one of the fundamental capabilities of a mobile robot to effectively execute a variety of tasks including docking, transportation, and manipulation. To achieve the desired navigation accuracy, mobile robots are typically equipped with on-board sensors to observe persistent features in the environment, to estimate their pose from these observations, and to adjust their motion accordingly. Since real-world environments often contain changing or ambiguous areas, existing features can be insufficient for mobile robots to establish a robust navigation behavior. A popular approach to overcome this problem and to enable accurate localization is to use artificial landmarks like reflective markers or barcodes. However, depending on the type of artificial landmark, the landmarks themselves or their precise placement can be costly. Therefore, it is desirable to place as few landmarks as possible to achieve the desired accuracy, which makes selecting beneficial landmark positions especially important.

In industrial settings, for example, mobile robots often have to perform repetitive tasks that include traveling along the same trajectory through a known environment. For such scenarios, we present approaches that aim at finding a minimum set of landmark positions in order to optimize the expected quality of the robot's task execution. We measure this quality either by the expected accuracy of the localization estimate of the robot or – if the robot bases its navigation decisions directly on its localization estimate – by the expected accuracy of the navigation behavior of the robot. In order to efficiently estimate the expected accuracy in navigation, we introduce a novel recursive calculation scheme for the expected distributions of the robot's deviation from its desired trajectory. For dealing with the generally $\mathbf{NP}$-hard landmark placement problem, we use techniques from submodular function optimization to efficiently generate near-optimal landmark configurations. The resulting efficiency of our landmark placement approaches makes it possible to apply them even to large-scale scenarios.

In contrast to the above-mentioned methods, landmark placement for robots traveling through unknown and unmapped environments requires different approaches. If the robot has a device to deploy a limited number of artificial landmarks itself, it will later be able to use them as fixed anchors to adjust the estimate of its relative motions between individual observations of the same landmark. We present a novel approach for learning an optimal landmark deployment policy for this scenario.

We evaluated all presented methods in extensive experiments both in simulation and

with real mobile robots. The experiments demonstrate that our approaches outperform baseline methods and work well on real robots. We believe that the presented landmark placement methods are a useful tool for guaranteeing a safe and reliable operation of mobile robots in practice, especially in industrial settings.

Acknowledgments

Without the support of several people, this thesis would not exist. First of all, I would like to thank my advisor Wolfram Burgard for giving me the guidance that I needed in the beginning of my graduate studies, and for giving me the freedom to pursue my own ideas later on. His support, his encouragement, and the exceptional work environment that he created in the AIS group made my research not only possible, but fun. Besides many other of his ideas that influenced this thesis, the general idea of considering landmark placement as an optimization problem was his.

I would also like to thank Andreas Krause for sharing his insights in intensive discussions that led to the landmark placement approach presented in Chapter 7 and for acting as a reviewer for this thesis.

Many thanks to my co-authors and collaborators Jörg Müller, Henrik Kretzschmar, Daniel Meyer-Delius, Jürgen Hess, Daniel Kuhner, Philipp Ruchti, and Alexander Kleiner for the great time working together. I thank Henrik Kretzschmar, Axel Rottmann, and Jörg Müller for teaching a mathematician how to program and for being great friends. Furthermore, I would like to thank Christoph Sprunk for discussions and help concerning the holonomic motion model of the KARIS robot, Rainer Kümmerle for g2o support, and Dominik Joho for sharing his experience in RFID technology with me. For proof-reading earlier drafts of this document, I thank Nichola Abdo, Felix Endres, Barbara Frank, Jürgen Hess, Henrik Kretzschmar, Markus Kuderer, Tayyab Naseer, Christoph Sprunk, Benjamin Suger, and Andreas Wachaja.

For their administrative and technical support, I thank Susanne Bourjaillat, Michael Keser, Manuela Kniss, and Dagmar Sonntag.

A special thanks goes to my parents, who lovingly raised and educated me and thereby made it possible for me to start working on this thesis in the first place and to Gisbert Lawitzky, who suggested applying at the AIS lab to me. My deepest gratitude and thanks go to my family: Kathrin for her love, her patience, and for believing in me, and Paul for the joy and happiness he has brought to my life.

This work has been partly supported by the German Research Foundation (DFG) within the Research Training Group 1103 and under contract number SFB/TR 8.

Contents

Chapter 1

Introduction

Mobile robots nowadays fulfill a broad variety of tasks. There are not only a few of them exploring foreign planets, but mobile robots are widely used for performing repetitive tasks in industry, and even chores like vacuum cleaning in our homes. These different applications of mobile robots all have in common that the quality with which the robot can autonomously fulfill its task depends strongly on the localization accuracy of the robot. For example, a badly localized robot on a foreign planet might accidentally drive onto dangerous terrain from which it cannot back out again, badly localized industrial robots might not be able to perform precise docking maneuvers, and badly localized vacuum cleaning robots might miss dirty spots on the floor or even fall down a staircase. In order to accurately localize themselves, mobile robots are typically equipped with on-board sensors, like cameras or laser range finders, to observe their environment. If the environment of the robot is known beforehand and the robot has access to a map of the observable features persistent in the environment, it can use its sensor observations to globally localize itself in the map. But even if the robot moves through an unknown and unmapped environment, it can use repeated observations of the same feature to locally adjust its estimate of its relative motions between these observations.

Especially in industrial applications, a high degree of accuracy and repeatability is usually necessary. Typical tasks for industrial mobile robots include accurately executing the same small number of pre-defined trajectories many times. However, industrial environments often contain ambiguous and dynamic areas, in which there might not exist enough persistent environment features for an accurate localization of the robot. A common way to deal with this problem is to place artificial landmarks like reflective or color-coded markers at specific positions in the environment in order to guarantee the desired accuracy in localization along the planned trajectory of the robot [25, 111]. However, depending on the type of artificial landmark, the landmarks themselves or the precise placement of the landmarks in the environment can be expensive. Also, especially active landmarks like radar or infrared beacons require a high degree of maintenance. Therefore, it is desirable to place as few landmarks as possible, which makes selecting beneficial landmark positions especially important.

In this thesis, we consider the problem of selecting the optimal positions for placing uniquely identifiable artificial landmarks in order to improve the quality of the robot's task execution. For landmark placement, we consider two different quantities that influence the quality of the task execution. The first is the accuracy of the localization estimate of the robot. The landmark observations of the robot directly influence this criterion. The second is the navigation accuracy of the robot, i.e., its ability to stay close to its desired trajectory during operation. If the robot autonomously makes its navigation decisions based on its localization estimate, then an accurate localization ultimately leads to an accurate navigation behavior. Optimizing this quality criterion leads to landmark positions observable for the robot in situations in which an accurate localization is especially important for the navigation task at hand, like, for example, sharp turns. When evaluating the quality criteria, we take into account that mobile robots in the real world are typically faced with noise and errors in motion execution and sensor observations. Therefore, we describe the evolution of the state of the robot during operation with a sequence of probability distributions and do not optimize the ideal values of the quality criteria but their expected values according to these distributions.

The two main challenges in landmark placement based on these quality criteria are estimating the expected localization and navigation accuracy and dealing with the combinatorial nature of the placement problem. Landmarks are placed before the robot starts operation, when the concrete observations and control commands are not yet known. At that time, computing the expected localization and navigation accuracy of the robot requires integrating over the high-dimensional space of all motion controls and sensor observations that the robot will possibly perform during operation. In general, this integral cannot be solved in closed form. However, being able to calculate the expected quality criteria in an efficient way is especially important in landmark placement due to the second challenge, the combinatorial problem structure of the placement problem. Landmark placement is the problem of selecting the minimum subset of the set of all possible landmark locations for placement that yields the desired value of the considered quality criterion. This combinatorial structure typically leads to $\mathbf{NP}$-hard optimization problems. Therefore, often efficient algorithms that approximate the optimal solution are applied. Still, also these algorithms usually require to evaluate a large number of candidate landmark sets, which is why an efficient way of evaluating the expected quality criteria is important.

In scenarios in which the environment of the robot contains dynamic components, e.g., other vehicles or temporarily deposited items, an additional challenge for landmark placement occurs. The dynamic components might unpredictably obstruct some of the placed landmarks from the field of view of the robot. Therefore, it is important that the placed landmarks yield a robust accuracy in task execution, even if a certain number of them is not observable.

In contrast to the landmark placement problems, landmark placement for robots traveling through unknown and unmapped environments requires different approaches. In such situations mobile robots do not only need to localize themselves in a given map in order to navigate accurately, they need to simultaneously construct a map of the environment from their sensor observations and localize themselves in this very map. In this context, we consider the fundamental problem of data association. Data association is the problem of deciding if two observations stem from the same environment feature or from different ones. If the robot itself can deploy a limited number of uniquely identifiable artificial landmarks during operation, it can use these landmarks later on as fixed anchors to ease the data association problem for the environment features. The key challenge here is to make the decisions when to deploy the artificial landmarks based only on the information that the robot has gathered with its own sensors. These decisions are crucial, as poor choices might prevent the robot from establishing correct data associations.

Summing up, we have identified the following key questions occurring in landmark placement for mobile robots:

- How to identify the optimal configuration of landmarks in the environment of a mobile robot to guarantee a certain accuracy in localization or in navigation with a minimum number of landmarks?
- How to deal with the combinatorial structure of the landmark placement problem?
- How to estimate the expected navigation and localization accuracy efficiently?
- How to deal with the problem that placed landmarks might unpredictably be obstructed from the view of the robot?
- How to decide if the robot should autonomously deploy a landmark in a given situation based only on the imperfect information available to the robot?

In the next section, we summarize our key contributions towards answering these questions.

1.1 Key Contributions

The key contributions of this thesis are approaches to placing artificial landmarks along the desired trajectory of a mobile robot and an approach to learning a policy for deciding when to autonomously deploy artificial landmarks with a mobile robot. In summary:

- We present a novel landmark placement approach for minimizing the expected localization uncertainty of a mobile robot (Chapter 4). Using the concept of submodularity, we prove that this landmark placement approach produces near-optimal

landmark sets with high confidence. For evaluating the expected localization uncertainty, we apply Monte Carlo simulation.

- For evaluating the expected navigation accuracy of a mobile robot, we introduce an efficient recursive estimation scheme (Chapter 5), which is significantly faster than the Monte Carlo simulation used in the above approach. It assumes that the robot uses a linear-quadratic regulator for navigation and applies a linearization on the whole navigation cycle.
- We present a novel landmark placement method for optimizing the expected navigation accuracy of a mobile robot (Chapter 6). Our approach applies the above-mentioned efficient estimation scheme for evaluating the expected navigation accuracy, which makes the landmark placement highly efficient. Therefore, this approach can be used even for large-scale scenarios.
- We introduce a second approach to landmark placement for mobile robot navigation (Chapter 7), which builds on and extends the above method. This approach explicitly considers the fact that a certain number of the placed landmarks might unpredictably be obstructed from the view of the robot. It applies techniques from submodular function optimization to deal with the combinatorial structure of the landmark placement problem.
- Finally, we introduce a novel method for learning when to deploy artificial landmarks autonomously with a mobile robot in order to optimize its data association performance (Chapter 9). We use Monte Carlo reinforcement learning for computing an optimal policy and apply a statistical convergence test to decide if the policy is converged and the learning process can be stopped.

1.2 Publications

Parts of this thesis have been published in the proceedings of international conferences, in an international journal, and in the proceedings of a workshop. The publications are stated in chronological order.

- M. Beinhofer, J. Müller, and W. Burgard. Near-optimal landmark selection for mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

- M. Beinhofer, J. Müller, and W. Burgard. Landmark placement for accurate mobile robot navigation. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2011.

- M. Beinhofer, H. Kretzschmar, and W. Burgard. Deploying artificial landmarks to foster data association in simultaneous localization and mapping. In *Proc. of the*

IEEE Int. Conf. on Robotics & Automation (ICRA), 2013.

- M. Beinhofer, J. Müller, A. Krause, and W. Burgard. Robust landmark selection for mobile robot navigation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

- M. Beinhofer, J. Müller, and W. Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics and Autonomous Systems (RAS)*, 61(10):1060 – 1069, 2013.

- M. Beinhofer and W. Burgard. Efficient estimation of expected distributions for mobile robot navigation. In *Proc. of the Austrian Robotics Workshop (ARW)*, 2014.

The following publications were also written during the time as research assistant. However, material from these publications is not included in this thesis.

- 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 Int. Conf. on Robotics & Automation (ICRA)*, 2011.

- D. Meyer-Delius, M. Beinhofer, and W. Burgard. Occupancy grid models for robot mapping in changing environments. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, 2012.

- J. Hess, M. Beinhofer, D. Kuhner, P. Ruchti, and W. Burgard. Poisson-driven dirt maps for efficient robot cleaning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

- J. Hess, M. Beinhofer, and W. Burgard. A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

1.3 Collaborations

Parts of this thesis are the result of collaborations with others. The landmark placement approaches presented in Chapters 4, 6, and 7 were developed in collaboration with Jörg Müller, who contributed especially to the experiments with real robots presented in these chapters. Besides other theoretical contributions, the analysis of different control modes and the way in which we leverage conditional independence in the Bayesian network in Chapter 4 were developed in intensive discussions with Jörg Müller.

The landmark deployment approach presented in Chapter 9 was developed in close co-

operation with Henrik Kretzschmar. Besides other things, Henrik Kretzschmar contributed especially to the aspects of this approach that are related to SLAM and reinforcement learning.

1.4 Notation

A note on the notation: Throughout this thesis, the distinction between random variables and their outcomes is usually clear from the context. We therefore typically use the shorthand notations $p(\mathbf{x}) := p(\mathbf{X} = \mathbf{x})$ to denote the value of the probability density function of the random variable $\mathbf{X}$ evaluated at the specific value $\mathbf{x}$ and $p(\mathbf{x} \mid \mathbf{y}) := p(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y})$ to denote the value of the probability density function of $\mathbf{X}$ evaluated at the specific value $\mathbf{x}$ conditioned on the fact that the random variable $\mathbf{Y}$ takes on the specific value $\mathbf{y}$. These shorthand notations are also widely used in the standard literature (see, e.g., Definition 4.2.1 in [19] or Section 2.2 in [102]). Only when considering information theoretic values (see, e.g., Section 3.2) we make a clear distinction between random variables and the values that they can take on. Throughout this thesis, we use the following notation:

Notation	Meaning
$x, y, \ldots$	Scalar values
$\mathbf{x}, \mathbf{y}, \ldots$	Column vectors
$\mathbf{x}_{1:t}$	Sequence of the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$
$\mathbf{X}, \mathbf{Y}, \ldots$	Vector-valued random variables
$\mathcal{A}, \mathcal{B}, \ldots$	Sets
$A, B, \ldots$	Matrices
$p(\mathbf{x})$	Probability density function evaluated at the specific value $\mathbf{x}$
$p(\mathbf{x} \mid \mathbf{y})$	Conditional probability density function at $\mathbf{x}$ conditioned on $\mathbf{y}$
$\mathbb{E}[\mathbf{X}]$	Expected value of the random variable $\mathbf{X}$
$\mathrm{Cov}(\mathbf{X})$	Covariance matrix of the random variable $\mathbf{X}$
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance Σ
$\mathbf{x}^T, A^T$	Transposes of the vector $\mathbf{x}$ and the matrix A
$\lvert A \rvert$	Determinant of the matrix A

Throughout this thesis, we use the following abbreviations:

Abbreviation	Meaning
EKF	Extended Kalman filter
HMM	Hidden Markov model
LQR	Linear-quadratic regulator
NP	Nondeterministic polynomial time
RFID	Radio-frequency identification
SLAM	Simultaneous localization and mapping
UKF	Unscented Kalman filter

Chapter 2

Related Work

In this thesis, we consider two different problems in the area of landmark placement. The first problem is to place artificial landmarks along the desired trajectory of a mobile robot before it starts operation. In this case, the goal is to optimize either the navigation performance or the localization performance of the robot during operation. The second problem is learning a policy that aids a mobile robot's SLAM system by autonomously deploying artificial landmarks during operation of the robot. In this chapter, we present a survey of the literature related to these two topics.

2.1 Landmark Placement for Localization and Navigation

The goal of our landmark placement approaches from Chapters 4, 6, and 7 is to place landmarks at a set of locations along the desired trajectory of a mobile robot to optimize either its localization performance (Chapter 4) or its navigation performance (Chapters 6 and 7). For measuring the navigation performance of the robot, we introduce a novel method for estimating the expected probability distributions of the states of the robot along its desired trajectory even before it starts operation. In the landmark placement algorithms from Chapters 4 and 7, we apply techniques from submodular function optimization.

In the following, we first give an overview of the literature related to landmark placement, and then present a survey of the state of the art in estimating expected distributions of dynamic systems. Finally, we discuss literature related to submodular function optimization.

2.1.1 Landmark Placement

In the past, the problem of finding an optimal set of landmark positions has been addressed from several points of view. Salas and Gordillo [92] consider it in terms of the art gallery problem. They use simulated annealing to find a landmark set that maximizes the area in

which a robot has a clear line of sight to at least one landmark. Erickson and LaValle [28] consider the same problem for colored landmarks. They add the constraint that from no position in the map two landmarks of the same color may be visible, and give bounds for the minimum number of colors needed to cover a polygonal region. In an extension presented in [29], they derive bounds on the maximum number of color-coded landmarks needed. Sala *et al.* [91] extend this problem to select landmark positions so that at every position in the map, at least k landmarks are observable. Rupp and Levi [90] select landmark positions on the walls of an indoor environment close to a given set of localization points. They use geometrical insights to find the landmark locations. Unlike these methods, which assume a deterministic robot behavior, our approaches explicitly model the noise of the sensors and actuators of the robot.

Ercan *et al.* [27] aim at finding the optimal configuration of bearing sensors on the boundaries of a square environment without obstacles for localizing a target inside this environment. They assume that the sensors are subject to random errors and present an approximate solution to the problem based on semidefinite programming. Tokekar and Isler [103] consider a version of this problem in which the bearing sensors can be placed anywhere inside the square environment and receive measurements with bounded errors. They analytically derive the optimal sensor configuration for this scenario. Jourdan and Roy [44] consider a fixed set of possible target positions. They place sensors on the walls of buildings to minimize the average position error bound in the sensor network. Finally, also Meyer-Delius *et al.* [72] present an approach that is independent of the trajectory taken by the robot. They increase the localization accuracy of a system already equipped with a landmark-independent sensor (e.g., a laser range finder) by placing additional landmarks in the environment. In contrast to these methods, our approaches take into account the full specification of the robot and its navigation task.

Like our approaches, Vitus and Tomlin [105] consider the full problem specification to place sensors in the environment. For measuring the navigation performance of the robot, they approximate the covariances of the expected distributions of the states of the robot with the posterior covariances of the most likely run of the robot. Similar to our approaches from Chapters 6 and 7, van den Berg *et al.* [12] evaluate sensor positions using the exact expected distributions (which they call a priori distributions) in a linearized system. Since they focus mainly on path planning, they restrict themselves to randomly sampled positions of a single sensor.

While all of the approaches above place artificial landmarks or sensors before the operation of the robot, the following approaches decide whether to utilize observed landmarks during operation. One of the first approaches in this field was introduced by Wünsche [110] in 1988. It selects the set of visible features for state estimation that produces the observation equations with the lowest condition number. Thrun [101] selects the subset of the observed landmarks for localization which minimizes the average

posterior localization error. Lerner *et al.* [65] use semi-definite programming to select landmarks that minimize the trace of the pose covariance of a moving camera. In contrast to our methods, these methods base their decisions on posterior distributions, i.e., on the information already gathered by the robot during operation.

2.1.2 Expected Distributions

There are several ways to estimate the expected distributions of the states of a dynamic system with respect to the executed control commands and sensor observations: Possibly the most generally applicable, but also computationally most demanding method is Monte-Carlo simulation. Roy *et al.* [89], for example, use Monte-Carlo simulation to estimate the expected entropy of the robot state in their coastal navigation framework. In Chapter 4, we present an approach to landmark placement that uses the same simulation as Roy *et al.* for estimating expected entropies. This technique, however, is orders of magnitude slower than our novel approach to computing expected distributions (see Section 5.1), which uses an efficient recursive calculation instead of a time-consuming simulation of sample episodes.

Another method to estimate expected distributions is to use the posterior distributions as an approximation. Vitus and Tomlin [105] use this method for sensor placement. For a given desired robot trajectory, they aim at placing sensors at a set of locations in the environment that optimizes the navigation performance of the robot. For measuring the navigation performance of the robot, they approximate the covariances of the expected distributions with the posterior covariances of the most likely run of the robot. Mastrogiovanni *et al.* [69] also use the posterior distributions to estimate the pose uncertainty of a robot before operation. They, however, use these distributions to find the optimal configuration for mounting a laser scanner on a mobile robot. Prentice and Roy [84] calculate posterior distributions to estimate the expected cost for path planning. However, posterior distributions do not encode the true uncertainty about the deviation of the robot from its desired state, but rather how well the localization framework of the robot can estimate these deviations. For measuring the navigation performance of the robot, we therefore do not apply this method.

To our knowledge, van den Berg *et al.* [11] were the first to introduce a recursive calculation scheme for expected distributions of dynamic systems, which they called a priori distributions. They used the calculated expected distributions for collision-free path planning, and later applied their approach to needle steering for surgical robots [12]. Since then, their calculation scheme has been applied to several kinds of applications. Vitus and Tomlin [106], for example, used it for chance constrained optimal control, and Patil *et al.* [82] applied it to motion planning in deformable environments.

The method that we introduce in Section 5.1 builds on the same linearization as the one by van den Berg *et al.*, and calculates expected distributions considerably faster than

their method. Therefore, it could benefit all of the above approaches.

2.1.3 Submodular Function Optimization

On the topic of submodular function optimization, there exists a large body of literature. Among the first authors to consider the submodularity property when optimizing set functions were Nemhauser, Wolsey, and Fisher. In a series of papers [32, 77, 78, 109], authored together and individually, they derived efficient approximation algorithms for the (generally NP-hard) problem of optimizing submodular set functions and proved tight approximation bounds for these algorithms.

Krause and Guestrin [53, 54] introduced these concepts in the field of machine learning. Together with others, they extended the concepts by Nemhauser *et al.* to problems like Gaussian Process optimization [58], optimization on graphs [55, 60], and robust optimization of a combination of objective functions [57], and applied them to temperature monitoring [53], path planning for multiple robots [96], and securing large water distribution networks [56].

Lately, submodular function optimization became increasingly popular in robotics. Vernaza *et al.* [104] applied submodular optimization techniques on terrain classification for outdoor robots, Schulman *et al.* [93] on selecting beneficial grasping poses for robotic manipulators, and Hollinger *et al.* [41] on model fitting for ship hull inspection with an underwater robot.

Our landmark placement approach from Chapter 7 applies the techniques introduced by Krause *et al.* [57] to select a set of landmark positions that robustly bounds the deviation of the robot from its desired trajectory, even if a certain number of landmarks are hidden from the view of the robot. In the landmark placement approach presented in Chapter 4, we build on the work of Krause and Guestrin [53] on optimizing information gain. We extend it from optimizing the information gain of bounded random variables on discrete spaces to optimizing the information gain of the states and observations of a mobile robot, which we define as unbounded random variables on continuous spaces.

2.2 Autonomous Landmark Deployment

In Chapter 9, we present an approach to learning a policy for autonomously deploying artificial landmarks with a mobile robot while it performs simultaneous localization and mapping (SLAM) [102]. The goal of our approach is to optimize the data association performance in SLAM without interfering with the SLAM system during operation.

In the past, several approaches to tackle the data association problem in SLAM have been developed. One popular method that does not rely on artificial landmarks is the joint compatibility branch and bound method by Neira and Tardós [76]. It explicitly

considers the correlations between landmarks by searching an interpretation tree for the hypothesis that covers the largest number of jointly compatible pairings. Olson [79] looks for local matches in the environment and aims to reject those matches that are not globally consistent using single cluster graph partitioning, which relies on a pair-wise consistency graph. In contrast to our approach, methods without the aid of artificial landmarks have to solely rely on the landmarks present in the environment. Therefore, with increasing ambiguity in the environment it becomes more challenging for such methods to robustly find the correct data associations.

The majority of approaches for SLAM with the aid of deployable landmarks address graph-like worlds and deterministically observable markers. For example, the approach of Dudek *et al.* [24] localizes a robot that travels along the edges of a graph and that can deploy and identify markers at the vertices. Bender *et al.* [10] present approaches for mapping a directed graph using deterministically observable undirected markers. Wang *et al.* [107] prove that the SLAM problem in an undirected graph can be solved deterministically if the robot can drop a deterministically observable directional marker. In contrast to these approaches, we apply a probabilistic model to deal with noisy motion and measurements.

Batalin and Sukhatme [3] devised a coverage strategy for a robot with no knowledge about its position. In their case, the robot can deploy active markers and use them later to move into the direction suggested by them. In the work by Kleiner *et al.* [50], the robot applies a manually designed heuristic, which takes into account the obstacle density and the estimated tag density, to deploy RFID markers to aid a SLAM system. In contrast to such approaches, our method learns a landmark deployment policy from simulated runs.

The problem of selecting informative environment features in SLAM is closely related to the problem considered in our approach. For example, Strasdat *et al.* [99] use reinforcement learning to determine a policy for feature selection that minimizes the distance between the final position of the robot and its goal. In contrast to our approach, they consider obstacle-free worlds and therefore do not need to incorporate information about the spatial structure of the environment into the learning method.

Chapter 3

Background

In this chapter, we give an overview of the established techniques in robotics and related fields that we apply in this thesis. In the first section of this chapter, we briefly introduce definitions and results from probability theory, which we apply for modeling robotic systems. In Section 3.2, we present basic concepts in information theory, which we use for defining the influence of landmark observations on the localization quality of a mobile robot. In Section 3.3, we describe Bayesian state estimation and its application to mobile robot localization. Section 3.4 covers submodular function optimization, which we use for finding a near-optimal configuration of landmarks in the environment of a mobile robot. Finally, in Sections 3.5 and 3.6, we briefly cover simultaneous localization and mapping (SLAM) and Monte Carlo reinforcement learning, two techniques that we apply to learning a policy for autonomously deploying artificial landmarks with a mobile robot.

3.1 Probability Theory

In this thesis, we use a probabilistic formulation for modeling the behavior of mobile robots. In this section, we briefly introduce definitions and results from probability theory, which build the foundation of the models introduced in the following chapters. For a comprehensive introduction to probability theory, see Kallenberg [47].

For describing randomness, we use random variables. A random variable X on a discrete space $\mathcal{X}$ is a variable that can take on different values $x \in \mathcal{X}$, each with a respective probability $P(X = x)$. In contrast to that, the space $\mathcal{X}$ considered in most of our applications is continuous. A random variable X on a continuous space can be fully described by its probability density function p. For each value $x \in \mathcal{X}$ that X can take on, the value $p(X = x)$ of the probability density function describes the marginal increase of the probability $P(X \in M)$ resulting from adding x to the set M. Therefore, calculating the probability $P(X \in M)$ that the value of X is in the set M can be achieved by integrating over the density function:

$$P(X \in M) = \int_{x \in M} p(X = x) \, \mathrm{d}x \, . \tag{3.1}$$

Two important quantities in the context of random variables are their expected values and their variances. The expected value (also called expectation or mean) of a random variable X on a continuous space $\mathcal{X}$ is defined as

$$\mathbb{E}[X] = \int_{x \in \mathcal{X}} x \, p(X = x) \, \mathrm{d}x \,. \tag{3.2}$$

Similarly, the variance of X is defined as

$$\mathrm{Var}(X) = \int_{x \in \mathcal{X}} (x - \mathbb{E}[X])^2 \; p(X = x) \, \mathrm{d}x \,. \tag{3.3}$$

Furthermore, for a vector-valued random variable $\mathbf{X}$ on a continuous vector space $\mathcal{X}$, its covariance matrix is defined as

$$\mathrm{Cov}(X) = \int_{\mathbf{x} \in \mathcal{X}} (\mathbf{x} - \mathbb{E}[\mathbf{X}])(\mathbf{x} - \mathbb{E}[\mathbf{X}])^T \, p(\mathbf{X} = \mathbf{x}) \, \mathrm{d}\mathbf{x} \,. \tag{3.4}$$

As can be seen from the definition, covariance matrices are always positive semidefinite.

A special type of random variables are Gaussian-distributed random variables, which are also called normal-distributed random variables. A vector-valued Gaussian-distributed random variable $\mathbf{X}$ on the space $\mathbb{R}^n$ is fully specified by its expectation and its covariance, which are typically denoted as $\mathbb{E}[\mathbf{X}] = \boldsymbol{\mu}$ and $\mathrm{Cov}(\mathbf{X}) = \Sigma$. Therefore, we use the notation

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{3.5}$$

to denote that a random variable $\mathbf{X}$ is Gaussian distributed with expectation $\boldsymbol{\mu}$ and covariance Σ. The probability density function of such a Gaussian is specified as

$$p(\mathbf{X} = \mathbf{x}) = ((2\pi)^n |\Sigma|)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right), \tag{3.6}$$

where n is the dimensionality of the random variable and $|\Sigma|$ is the determinant of its covariance matrix. An interesting property of Gaussians is that affine transformations of Gaussians always yield Gaussians:

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \quad \Rightarrow \quad A\mathbf{X} + \mathbf{b} \sim \mathcal{N}(A\boldsymbol{\mu} + \mathbf{b}, A\Sigma A^T) \,, \tag{3.7}$$

where $A \in \mathbb{R}^{m \times n}$ is a constant matrix and $\mathbf{b} \in \mathbb{R}^m$ is a constant vector.

For two random variables X and Y, their joint probability distribution can be defined by their joint probability density function

$$p(X = x, Y = y) \,. \tag{3.8}$$

X and Y are called independent, if

$$p(X = x, Y = y) = p(X = x) \, p(Y = y) \,. \tag{3.9}$$

If X and Y are not independent, then knowing the outcome y of Y changes the probabilities for the outcomes of X. Conditional probability density functions account for this fact. Given that $p(Y = y) > 0$, the probability density function of X conditioned on the outcome y of Y is defined by

$$p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)}\,. \tag{3.10}$$

The following two theorems are very useful when dealing with conditional probability density functions:

Theorem 3.1 (Law of total probability)**.**

$$p(X = x) = \int p(X = x \mid Y = y)\, p(Y = y)\, \mathrm{d}y\,. \tag{3.11}$$

Theorem 3.2 (Bayes rule)**.** *If $p(Y = y) > 0$, then it holds that*

$$p(X = x \mid Y = y) = \frac{p(Y = y \mid X = x)p(X = x)}{p(Y = y)}\,. \tag{3.12}$$

A fundamental theorem for comparing expected values is Jensen's inequality:

Theorem 3.3 (Jensen's inequality)**.** *Given a random variable X, a convex function f, and a concave function g, it holds that*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{3.13}$$

and

$$g(\mathbb{E}[X]) \geq \mathbb{E}[g(X)]\,. \tag{3.14}$$

Throughout the rest of this thesis, the distinction between random variables and their outcomes is usually clear from the context. We therefore typically use the shorthand notations

$$p(x) := p(X = x) \tag{3.15}$$

and

$$p(x \mid y) := p(X = x \mid Y = y)\,, \tag{3.16}$$

which are in accordance with the standard literature (see, e.g., Definition 4.2.1 in [19] or Section 2.2 in [102]). Only when considering information theoretic values, as in the next section, we make a clear distinction between random variables and the values that they can take on.

3.2 Information Theory

In this section, we give an overview of the basic concepts in information theory that we apply in Chapter 4 for measuring the localization quality of a mobile robot. For a detailed introduction to information theory, see Cover and Thomas [20].

Information theory considers the question of how to quantify information, and was initially introduced in 1948 by Shannon [94] to analyze data compression and signal processing techniques. Shannon defined the content of a message as a random variable and introduced the *entropy* of a random variable as the expected value of information contained in the corresponding message. Since then, the entropy of a random variable has also often been considered as a measure for the uncertainty about the outcome of the variable [14, 53, 98].

Concretely, the entropy H of a discrete random variable X on a set $\{x_1, \ldots, x_n\}$ is defined as

$$H(X) := -\sum_{i=1}^{n} P(X = x_i) \, \log_b(P(X = x_i)) \,, \tag{3.17}$$

where b the base of the logarithm, which, in this formula, is typically considered to be 2, e, or 10.

Even though originally intended only for discrete random variables by Shannon, the definition of the entropy can be extended to random variables on continuous spaces. This so-called differential entropy h of a random variable X on a continuous space can be defined as

$$h(X) := -\int p(X = x) \, \log p(X = x) \, \mathrm{d}x \,, \tag{3.18}$$

where $\log$ is the natural logarithm to the base e. Equivalently, the differential entropy of a set of random variables $X_1, \ldots, X_n$ is defined as

$$h(X_1, \ldots, X_n) := \\ -\int p(X_1 = x_1, \ldots, X_n = x_n) \, \log p(X_1 = x_1, \ldots, X_n = x_n) \, \mathrm{d}(x_1, \ldots, x_n) \,. \tag{3.19}$$

Based on these definitions, one can prove that the joint entropy $h(X_1, \ldots, X_n)$ of a set of mutually independent random variables $X_1, \ldots, X_n$ is the sum of the individual entropies [20], i.e.,

$$h(X_1, \ldots, X_n) = \sum_{i=1}^{n} h(X_i) \,. \tag{3.20}$$

The entropy of a random variable X conditioned on the specific outcome y of another random variable Y is defined as

$$h(X \mid Y = y) := -\int p(X = x \mid Y = y) \, \log p(X = x \mid Y = y) \, \mathrm{d}x \,. \tag{3.21}$$

In contrast to that, the entropy of the random variable X conditioned on the random variable Y itself (see Section 2.2 in [20]) is defined as

$$h(X \mid Y) := h(X, Y) - h(Y) \tag{3.22}$$

$$= -\int\int p(X = x \mid Y = y)\, \log p(X = x \mid Y = y)\, \mathrm{d}x\, p(Y = y) \mathrm{d}y \tag{3.23}$$

$$= \int h(X \mid Y = y)\, p(Y = y)\, \mathrm{d}y \tag{3.24}$$

where Equation (3.23) follows from applying the definition of the conditional probability density (Equation (3.10) in the previous section) multiple times. Equation (3.24) yields that $h(X \mid Y)$ is the expected value of $h(X \mid Y = y)$ with respect to the probability distribution of Y. A result of the above definition of conditional entropy is the "information never hurts"-principle, which states that

$$h(X \mid Y) \leq h(X)\,. \tag{3.25}$$

It can be proven by applying Jensen's inequality (Theorem 3.3 in the previous section) on Equation (3.24).

Finally, the mutual information I (also called information gain) of two random variables X and Y on continuous spaces is defined as the reduction of the differential entropy of one of the variables that results from observing the other variable:

$$I(X; Y) = h(X) - h(X \mid Y) = h(Y) - h(Y \mid X)\,. \tag{3.26}$$

In Chapter 4, we use the mutual information between the states of a robot and its landmark observations to measure the amount of information that the observations of the landmarks contain about the states of the robot.

3.3 Recursive Bayesian State Estimation

In this section, we give an overview of the general recursive Bayesian state estimation method [102] and its concrete implementations. Furthermore, we describe how recursive Bayesian state estimation can be used in landmark-based mobile robot localization.

Recursive Bayesian state estimation, also known as the Bayes filter, is a probabilistic method to recursively estimate the state of a dynamic system. In recursive Bayesian state estimation, the evolution of the state $\mathbf{x}$ of the system over time is modeled by a time-discrete sequence of states $\mathbf{x}_{0:T}$. At every time step t, the system performs two actions:

- *State transition:* The system executes a control command $\mathbf{u}_t$, changing the state $\mathbf{x}_{t-1}$ to the next state $\mathbf{x}_t$.

- *Observation:* The system generates an observation $\mathbf{z}_t$, from which the state $\mathbf{x}_t$ can be inferred.

The Bayes filter explicitly takes into account the uncertainties occurring in the execution of these actions by modeling them not with deterministic functions, but with probability distributions. Concretely, it models the state transition step with the probability distribution

$$p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \tag{3.27}$$

of the current state $\mathbf{x}_t$ conditioned on the current control command $\mathbf{u}_t$ and the previous state $\mathbf{x}_{t-1}$. It describes the observation step by the distribution

$$p(\mathbf{z}_t \mid \mathbf{x}_t) \tag{3.28}$$

that encodes the likelihood of making observation $\mathbf{z}_t$ when in state $\mathbf{x}_t$. In accordance with this probabilistic formulation, the Bayes filter also represents the state $\mathbf{x}_t$ of the system at time t not with a deterministic value, but with a complete probability distribution $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$, which is conditioned on all current and previous control commands and observations. This distribution is called the *belief* about the state of the system. The Bayes filter calculates the belief at time t recursively as

$$p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \, p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) \, \mathrm{d}\mathbf{x}_{t-1} \tag{3.29}$$

from the previous belief using the observation and transition distributions defined in Equations (3.27) and (3.28). In Equation (3.29), η_t is a normalizing constant that ensures that $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ integrates up to 1 over all $\mathbf{x}_t$.

The recursive update scheme defined in Equation (3.29) can be derived mathematically, if the dynamic system described above is framed as a hidden Markov model (HMM) [16]. Figure 3.1 shows the dynamic Bayesian network describing this HMM. The arrows in the figure describe the stochastic dependencies of the variables. As can be seen from the figure, the *complete state* assumption (also called Markov assumption) holds true in this dynamic Bayesian network. The complete state assumption means that the knowledge about the current state of the system contains all information about the future that is available from any past or present observations or control commands.

The basic idea behind the derivation of Equation (3.29) is to apply Bayes rule and the law of total probability on $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ and to use the conditional independence properties that, according to the rules of d-separation [16], follow from the complete state assumption in the HMM. For a detailed derivation, see Thrun *et al.* [102].

Derived from Equation (3.29), the Bayes filter algorithm (see Algorithm 1) updates the belief about the state of the system in two steps, a prediction and a correction step. The prediction step is shown in Line 2 of Algorithm 1. In this step, the Bayes filter algorithm predicts the current state $\mathbf{x}_t$ from the belief about the previous state $\mathbf{x}_{t-1}$ and the control

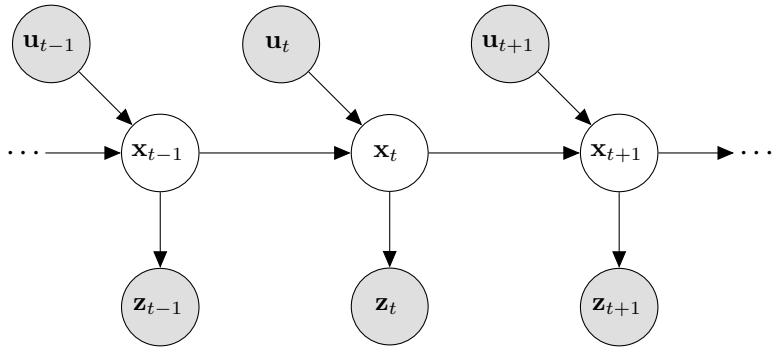

Figure 3.1: Dynamic Bayesian network describing the evolution of the state $\mathbf{x}_t$ of a system over time. The control commands are denoted as $\mathbf{u}_t$ and the observations as $\mathbf{z}_t$. The latent variables are shown in white and the observable variables in gray.

Algorithm 1 Bayes filter update

Input: $\mathbf{u}_t$, $\mathbf{z}_t$, $\forall \mathbf{x}_{t-1}$: $p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1})$
Output: $\forall \mathbf{x}_t$: $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$
1: **for all** $\mathbf{x}_t$ **do**
2: $\quad p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \leftarrow \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1}$
3: $\quad p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \leftarrow \eta_t\, p(\mathbf{z}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1})$
4: **end for**
5: **return** $\forall \mathbf{x}_t$: $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$

command $\mathbf{u}_t$. In the correction step (Line 3), the algorithm corrects the prediction using the current observation $\mathbf{z}_t$.

Note that the for-loop in this algorithm can only be computed explicitly if the space of all states $\mathbf{x}_t$ is finite. However, there exist several implementations of the Bayes filter that can also deal with infinite state spaces, either by approximation, or by assuming that the occurring probability density functions have special characteristics that allow for a closed form computation of the for-loop in Algorithm 1. The most common ones are the different variants of the particle filter [102] and of the Kalman filter [2]. The key idea behind the particle filter is to represent the belief about the state of the system by an empirical distribution, i.e., a set of samples called particles. In the prediction step, each particle is moved individually according to a random sample drawn from the transition distribution (Equation (3.27)), taking into account the previous state of the particle and the control command. In the correction step, the particle filter assigns a weight to every particle according to the current observation. Then, a new set of particles is randomly drawn with replacement from the current set according to the weights of the particles and the current set is discarded.

In the following we give a more detailed description of the Kalman filter, as in this thesis we apply mostly Kalman filter-type algorithms.

Algorithm 2 Kalman filter update

Input: $\mathbf{u}_t$, $\mathbf{z}_t$, $\boldsymbol{\mu}_{t-1}$, Σ_{t-1}
Output: $\boldsymbol{\mu}_t$, Σ_t
1: $\overline{\boldsymbol{\mu}}_t \leftarrow \mathbf{c}_t + A_t\boldsymbol{\mu}_{t-1} + B_t\mathbf{u}_t$
2: $\overline{\Sigma}_t \leftarrow A_t\Sigma_{t-1}A_t^T + V_tQ_tV_t^T$
3: $K_t \leftarrow \overline{\Sigma}_t H_t^T \left(H_t\overline{\Sigma}_t H_t^T + W_tR_tW_t^T\right)^{-1}$
4: $\boldsymbol{\mu}_t \leftarrow \overline{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - \mathbf{d}_t - H_t\overline{\boldsymbol{\mu}}_t)$
5: $\Sigma_t \leftarrow (I - K_tH_t)\overline{\Sigma}_t$
6: **return** $\boldsymbol{\mu}_t$, Σ_t

3.3.1 Kalman Filter

The Kalman filter is an efficient implementation of the Bayes filter for linear system dynamics and Gaussian distributions. Introduced in 1960 by Rudolf Kalman [48], the Kalman filter has been used successfully in many different applications [63, 67, 81, 87, 88]. However, it makes several restrictive assumptions. Concretely, it assumes that the transition between states follows the linear function

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{v}_t) = \mathbf{c}_t + A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t + V_t\mathbf{v}_t\,, \tag{3.30}$$

where the vector $\mathbf{c}_t$ and the matrices A_t, B_t and V_t are constants that depend only on the time step t, and $\mathbf{v}_t$ is a zero-mean Gaussian-distributed random variable capturing the noise in the transition, i.e., $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$. Furthermore, the Kalman filter assumes that the observations can also be described by a linear function

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{w}_t) = \mathbf{d}_t + H_t\mathbf{x}_t + W_t\mathbf{w}_t\,, \tag{3.31}$$

where $\mathbf{d}_t$, H_t, and W_t are constants, and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, R_t)$ is a zero-mean Gaussian-distributed random variable. Finally, the Kalman filter assumes that the initial distribution about the state of the system $p(\mathbf{x}_0)$ is a Gaussian with mean $\boldsymbol{\mu}_0$ and covariance Σ_0. Under these assumptions, the belief about the state of the system remains a Gaussian when propagated in the Bayes filter update. Since Gaussian distributions are fully specified by their mean and covariance, the Kalman filter only needs to update these two values. This makes it computationally highly efficient. Furthermore, Kalman [48] proved that if all above assumptions are satisfied, the Kalman filter is the minimum mean square error estimator for the state of the system. Therefore, the Kalman filter is called optimal.

Algorithm 2 states the recursive update of the Kalman filter. Lines 1 and 2 of the algorithm implement the prediction step of the Bayes filter. They calculate the predicted belief $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \sim \mathcal{N}(\overline{\boldsymbol{\mu}}_t, \overline{\Sigma}_t)$ at time t from the previous belief using the control command $\mathbf{u}_t$ and the transition function. Then, line 3 calculates the so-called Kalman gain matrix K_t. This matrix serves as a transformation from the state space to the

observation space. Additionally, it weights the uncertainty in the predicted belief against the uncertainty in the observation. Lines 4 and 5 use K_t to perform the correction step of the Bayes filter. They update the predicted belief with the observation $\mathbf{z}_t$ to produce the corrected belief $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$ about the state of the system. As can be seen from the algorithm, the computational complexity of the Kalman filter update is in $\mathcal{O}(\dim(\mathbf{x}_t)^{2.373} + \dim(\mathbf{z}_t)^{2.373})$, if the Williams algorithm [108] is applied both for the matrix multiplications and the matrix inversion.

There are several extensions and alternative versions of the Kalman filter, which aim at circumventing its restrictive assumptions or making the computations even more efficient. The Information filter [2], for example, updates the information matrix, which is the inverse of the covariance matrix, instead of the covariance matrix itself. It does not perform a matrix inversion in the correction step, which makes this part of the algorithm computationally more efficient. However, a matrix inversion is necessary in the prediction step of the information filter. Two extensions of the Kalman filter that alleviate its linearity assumption are the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). Since we use both these extensions in later chapters, we briefly describe them in the following.

The Extended Kalman Filter

The extended Kalman filter (EKF) [2] is an approximate implementation of the Bayes filter algorithm that allows to use the Kalman filter framework even for nonlinear transition functions and observation functions. It approximates the nonlinear functions with their first-order Taylor expansions around the mean of the filter $\boldsymbol{\mu}_t$ (resp. $\overline{\boldsymbol{\mu}}_t$) and around the control command $\mathbf{u}_t$. This leads to the linear approximations

$$\begin{aligned}
\mathbf{x}_t &= f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{v}_t) && (3.32)\\
&\approx f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0}) + A_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + B_t(\mathbf{u}_t - \mathbf{u}_t) + V_t(\mathbf{v}_t - \mathbf{0}) && (3.33)\\
&= f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0}) + A_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + V_t\mathbf{v}_t\,, && (3.34)\\
\mathbf{z}_t &= h(\mathbf{x}_t, \mathbf{w}_t) && (3.35)\\
&\approx h(\overline{\boldsymbol{\mu}}_t, \mathbf{0}) + H_t(\mathbf{x}_t - \overline{\boldsymbol{\mu}}_t) + W_t\mathbf{w}_t && (3.36)
\end{aligned}$$

of the transition function f and the observation function h. Here, the matrices A_t, B_t, V_t, H_t, and W_t are the Jacobians

$$A_t = \frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0}), \quad B_t = \frac{\partial f}{\partial \mathbf{u}}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0}), \quad V_t = \frac{\partial f}{\partial \mathbf{m}}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0}), \tag{3.37}$$

$$H_t = \frac{\partial h}{\partial \mathbf{x}}(\overline{\boldsymbol{\mu}}_t, \mathbf{0}), \quad W_t = \frac{\partial h}{\partial \mathbf{n}}(\overline{\boldsymbol{\mu}}_t, \mathbf{0}) \tag{3.38}$$

Algorithm 3 EKF update

Input: $\mathbf{u}_t$, $\mathbf{z}_t$, $\boldsymbol{\mu}_{t-1}$, Σ_{t-1}
Output: $\boldsymbol{\mu}_t$, Σ_t
1: $\overline{\boldsymbol{\mu}}_t \leftarrow f(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, \mathbf{0})$
2: $\overline{\Sigma}_t \leftarrow A_t \Sigma_{t-1} A_t^T + V_t Q_t V_t^T$
3: $K_t \leftarrow \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T \right)^{-1}$
4: $\boldsymbol{\mu}_t \leftarrow \overline{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - h(\overline{\boldsymbol{\mu}}_t, \mathbf{0}))$
5: $\Sigma_t \leftarrow (I - K_t H_t)\overline{\Sigma}_t$
6: **return** $\boldsymbol{\mu}_t$, Σ_t

resulting from the Taylor expansion. Algorithm 3 shows the EKF update. As can be seen from the algorithm, the EKF update with the approximated linear functions is identical to the standard Kalman filter update shown in Algorithm 2 with two exceptions: Lines 1 and 4, in which the values of the transition function and the observation function at the linearization points of the Taylor expansion appear. At these points, the values of the linear approximations of the EKF coincide with the values of the real nonlinear functions.

The approximation quality of the EKF depends largely on the real, nonlinear functions and on the covariances in the filter. If, for example, the real functions are linear combinations of trigonometric functions, which can be approximated quite well with linear functions in a small area around the linearization point, the EKF works typically well if the covariances are small enough that $\boldsymbol{\mu}_t$ does not deviate too far from the true state $\mathbf{x}_t$. On the other hand, if the real functions are highly nonlinear, e.g., with jump discontinuities or similar nonlinear parts, then the EKF might fail more often.

In Chapter 5, we use ideas similar to the ones in the EKF to linearize the whole navigation cycle of a mobile robot, which consists of executing a motion command, making an observation, localizing the robot, and selecting the next motion command depending on the localization.

The Unscented Kalman Filter

Like the EKF, the unscented Kalman filter (UKF) is an extension of the standard Kalman filter that allows using nonlinear transition functions and observation functions. It was introduced in 1995 by Julier *et al.* [45]. Instead of using Taylor expansions to linearize the functions, the UKF considers a deterministically selected set of points that represent the current belief and applies the real, nonlinear functions to these points, resulting in an updated point set. These points, called sigma points, are typically selected as the mean $\boldsymbol{\mu}_t$ of the belief and the vertices of the (scaled) covariance ellipsoid. In the prediction step, the UKF applies the nonlinear transition function to the sigma points and estimates the mean and covariance of the Gaussian for the next time step from the updated sigma

point set using weighted averages. In the correction step, the UKF applies the nonlinear observation function to each sigma point and calculates the differences between the resulting point set and the observed value $\mathbf{z}_t$. These differences are then used in an averaging procedure similar to the one in the prediction step to calculate the Kalman gain K_t and finally the parameters $\boldsymbol{\mu}_t$ and Σ_t of the updated belief.

In the experiments presented in Chapter 4, we use the UKF for state estimation.

3.3.2 Landmark-Based Mobile Robot Localization

In the following chapters, we apply the recursive Bayesian state estimation techniques described above to landmark-based mobile robot localization. In landmark-based mobile robot localization, the goal is to estimate the state of a mobile robot from its control inputs and from its sensor observations of a set of landmarks. The state of the robot is typically considered as its pose (i.e., its position and orientation), and the control inputs are typically the desired velocities, accelerations, or motor currents. The sensor observations that we consider are the observations of landmarks in the environment. In this context, landmarks can be anything that can be observed by the sensors of the robot. Examples of landmarks are corners, edges, and colored markers, which can be extracted from the images of a camera, reflective materials, which can be detected by laser scanners, or radio-frequency identification (RFID) tags. Depending on the type of sensor and landmark, the observations typically consist of the identity of the landmark and the relative distance and angle between the robot and the landmark. Given a map $\mathcal{A} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_N\}$ of the identities and positions of the finite number N of all landmarks in its environment, the robot can infer its own pose from the landmark observations.

In real-world applications, the motion and the sensor observations of mobile robots are typically subject to noise and errors. One way of dealing with these errors is to frame mobile robot localization as a recursive Bayesian state estimation problem and estimate the pose of the robot at every time step t not only as one deterministic value, but as a complete probability distribution on the space of all possible robot poses. Framed as a recursive Bayesian state estimation problem, the pose of the robot is considered as the state of the dynamic system $\mathbf{x}_t$, which is estimated from the control inputs $\mathbf{u}_t$ and the sensor observations $\mathbf{z}_t$ of the robot. In mobile robot localization, the state transition function

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{v}_t)\,, \tag{3.39}$$

is called the motion model of the robot. We typically assume the motion noise variable $\mathbf{v}_t$ to be zero-mean Gaussian distributed, i. e., $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$. The observation function

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{w}_t, \mathcal{A}) \tag{3.40}$$

in landmark-based mobile robot localization is called its sensor model. It depends not only

on the pose of the robot $\mathbf{x}_t$ and the (zero-mean Gaussian-distributed) noise variable $\mathbf{w}_t$, but also on the identities and positions $\mathcal{A}$ of the landmarks in the environment.

The standard motion models and sensor models [102] for mobile robots are nonlinear, but consist mostly of linear combinations of trigonometric functions. Therefore, their linearized versions tend to have a reasonable approximation quality, and variants of the Kalman filter such as the EKF and the UKF have often been successfully applied to mobile robot localization [63, 69, 102, 105].

3.4 Submodular Function Optimization

In this section, we give a brief overview of existing techniques for submodular function optimization, which we apply in later chapters when searching for the optimal subset of the set of all possible landmark positions in the environment of a mobile robot. See Krause and Golovin [52] for a more comprehensive survey on submodular function optimization.

The two general types of optimization problems that we consider are the *constrained set size problem* and the *constrained function value problem*. Given a set $\mathcal{V}$ and a maximum number n of elements $\boldsymbol{e} \in \mathcal{V}$ that may be selected, the constrained set size problem is to find the configuration of elements that maximizes a given objective function F:

$$\mathcal{A}_n^* = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A}) \,. \tag{3.41}$$

The constrained function value problem, on the other hand, is the problem of minimizing the number of elements needed to reach a given threshold d on the objective function:

$$\mathcal{A}_d^\star = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}; F(\mathcal{A}) \geq d} |\mathcal{A}| \,. \tag{3.42}$$

To efficiently select the elements in $\mathcal{A}_n^*$ and $\mathcal{A}_d^\star$, one important issue that has to be addressed is the combinatorial structure of the optimization problem. Many combinatorial optimization problems, including the one that we consider in Chapter 4, are $\mathbf{NP}$-hard. Therefore, in practice, one often uses methods to efficiently find an approximate solution to the optimization problem. For the special case in which the considered objective function is *submodular*, there exist approximation algorithms with tight performance guarantees.

3.4.1 Greedy Algorithm for Maximizing Submodular Functions

Submodularity is a property that is defined for *set functions*. A set function is a function $F : \mathcal{P}(\mathcal{V}) \to \mathbb{R}$ that specifies a value $F(\mathcal{A})$ for each subset $\mathcal{A}$ of a given set $\mathcal{V}$. We define submodularity for set functions in the following way:

Algorithm 4 Greedy algorithm for set function maximization

Input: $\mathcal{V}$, n or d (depending on the type of constraint)
Output: $\mathcal{A}$
  $\mathcal{A} \leftarrow \varnothing$
  repeat
    $\boldsymbol{e}^\star \leftarrow \operatorname{argmax}_{\boldsymbol{e} \in \mathcal{V}} \; F(\mathcal{A} \cup \{\boldsymbol{e}\})$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{e}^\star\}$
  until constraint satisfied
  return $\mathcal{A}$

Definition 3.1. *A set function $F : \mathcal{P}(\mathcal{V}) \to \mathbb{R}$ is called submodular if it holds that $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}, \boldsymbol{e} \in \mathcal{V} \setminus \mathcal{B}$:*

$$F(\mathcal{B} \cup \{\boldsymbol{e}\}) - F(\mathcal{B}) \leq F(\mathcal{A} \cup \{\boldsymbol{e}\}) - F(\mathcal{A}) \,. \tag{3.43}$$

Submodularity is a diminishing returns property that is natural for objective functions in many applications. For example, in the context of landmark placement, it means that the increase in the value of the objective function resulting from placing an additional landmark is lower when more landmarks have already been placed. In the following, we consider functions F that, in addition to being submodular, also fulfill $F(\varnothing) = 0$ and are monotonically increasing.

Definition 3.2. *A set function $F : \mathcal{P}(\mathcal{V}) \to \mathbb{R}$ is called monotonically increasing if $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ it holds that $F(\mathcal{A}) \leq F(\mathcal{B})$.*

The following two corollaries follow directly from Definition 3.1 and are useful for building complex submodular functions out of simple parts.

Corollary 3.1. *Nonnegative linear combinations of submodular functions are submodular, i. e., if $F_1, \ldots, F_k$ are submodular, then also $H(\mathcal{A}) = \sum_{i=1}^{k} a_i F_i(\mathcal{A})$ is submodular for arbitrary nonnegative constants $a_1, \ldots, a_k$.*

Corollary 3.2. *Truncations of monotonically increasing submodular functions are monotonically increasing and submodular, i. e., if F is monotonically increasing and submodular, then also $G(\mathcal{A}) = \min(F(\mathcal{A}), c)$ is monotonically increasing and submodular for arbitrary constants c.*

Corollary 3.1, for example, yields among other things that the average of submodular functions is submodular itself.

For finding approximate solutions to the constrained set size problem and to the constrained function value problem for monotonically increasing submodular objective

functions F, we use the iterative placement procedure defined in Algorithm 4. The constraint that needs to be satisfied in the until-statement of the algorithm is $(|\mathcal{A}| = n)$ for the constrained set size problem and $(F(\mathcal{A}) \geq d)$ for the constrained function value problem. The algorithm works iteratively. In each iteration, it greedily adds the one element $\boldsymbol{e}^\star \in \mathcal{V}$ to the set $\mathcal{A}$ of selected elements that maximizes the *marginal gain* $F(\mathcal{A} \cup \{\boldsymbol{e}^\star\}) - F(\mathcal{A})$. The algorithm evaluates the objective function at most $|\mathcal{V}|$ times for every selected element.

3.4.2 Approximation Guarantees

Applying greedy algorithms on submodular functions is a popular choice in many applications [26, 42, 49, 59, 62], as there exist tight bounds on the approximation performance. In this section, we state the performance guarantees for the greedy algorithms for the constrained set size problem and for the constrained function value problem. As these performance guarantees are the foundation on which we build in later chapters, we also present the basic ideas behind their proofs here.

One of the most famous results in submodular function optimization is the performance guarantee for the constrained set size problem by Nemhauser *et al.* [78] dating back to 1978:

Theorem 3.4 (Nemhauser *et al.* [78])**.** *Given a submodular, monotonically increasing function F with $F(\varnothing) = 0$, the result $\mathcal{A}_n$ of the greedy algorithm for the constrained set size problem satisfies the following approximation guarantee:*

$$F(\mathcal{A}_n) \geq (1 - 1/e) F(\mathcal{A}_n^\star) \,, \tag{3.44}$$

where $F(\mathcal{A}_n^\star) = \max_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A})$.

As e in Equation (3.44) refers to Euler's number, this approximation bound guarantees that the greedy algorithm results in a set with at least $(1 - 1/e) \approx 63\%$ of the maximum function value.

The proof of Nemhauser *et al.* for Theorem 3.4 starts with the inequality $F(\mathcal{A}_n^\star) \leq F(\mathcal{A}_n^\star \cup \mathcal{A}_n)$, which holds true due to the monotonicity of F. It then expresses the right hand side of the inequality in terms of the marginal gains

$$F(\mathcal{A}_n \cup \{\boldsymbol{e}^\star\}) - F(\mathcal{A}_n) = \Delta F(\{\boldsymbol{e}^\star\}) \,, \tag{3.45}$$

where $\boldsymbol{e}^\star$ are the elements of $\mathcal{A}^\star$. Since the greedy algorithm explicitly maximizes marginal gains, the definition of submodularity yields that the marginal gains $\Delta F(\{\boldsymbol{e}^\star\})$ can be bounded by the marginal gains that the elements selected by the greedy algorithm

actually yielded. Bounding all marginal gains by the gain $F(\{\boldsymbol{e}_1\}) - F(\varnothing) = F(\{\boldsymbol{e}_1\})$ of the first greedily selected element yields the desired result:

$$F(\mathcal{A}_n^\star) - F(\mathcal{A}_n) \leq \left(1 - \frac{1}{n}\right)^n F(\{\boldsymbol{e}_1\}) \leq \frac{1}{e} F(\mathcal{A}_n^\star)\,, \tag{3.46}$$

where the second inequality holds true due to the fact that $1/e$ is an upper bound for $(1 - 1/n)^n$ and the fact that $F(\mathcal{A}_n^\star) \geq F(\{\boldsymbol{e}\})$ for any single element $\boldsymbol{e}$ of $\mathcal{V}$.

For a large class of submodular functions, it has been shown that the bound in Theorem 3.4 is the best approximation guarantee that any polynomial time algorithm can achieve, unless $\mathbf{P} = \mathbf{NP}$. The two main results in this direction are Theorem 4.2 of Nemhauser and Wolsey [77] and Theorem 5.3 of Feige [31]. Nemhauser and Wolsey showed that the bound in Theorem 3.4 is tight if the submodular function F is available only in the form of a *value oracle*, i.e., a black box that takes a set $\mathcal{A}$ as input and outputs the value $F(\mathcal{A})$. Feige proved that the bound is tight for the special case of the maximum coverage problem (MAX-COVER), which can be framed as a submodular optimization problem. Given an arbitrary collection of sets and a number n, MAX-COVER is the problem of selecting the n sets out of the collection whose union has maximum size. Obviously, Feige's result also holds for all problems that MAX-COVER can be reduced to.

Nemhauser's bound in Theorem 3.4 gives a guarantee on the approximation performance of the greedy algorithm for the constrained set size problem. Wolsey [109] derived its counterpart for the greedy algorithm for the constrained function value problem:

Theorem 3.5 (Wolsey [109])**.** *Given an integer-valued, submodular, monotonically increasing function F with $F(\varnothing) = 0$ and a feasible threshold d, the result $\mathcal{A}_d$ of the greedy algorithm for the constrained function value problem satisfies the following approximation guarantee:*

$$|\mathcal{A}_d| \leq |\mathcal{A}_d^\star| \Big(1 + \log \max_{\boldsymbol{e} \in \mathcal{V}} F(\{\boldsymbol{e}\})\Big)\,, \tag{3.47}$$

where $|\mathcal{A}_d^\star| = \min\limits_{\mathcal{A} \subseteq \mathcal{V}; F(\mathcal{A}) \geq d} |\mathcal{A}|$.

Wolsey formulated this theorem only for integer-valued functions, but Krause *et al.* [57] extended it to real-valued functions by approximating the function values with their highest-order bits. For details on this, see Section 7.1 of [57].

To prove Theorem 3.5, Wolsey considers the problem defined in Equation (3.42) as an integer programming problem. He introduces a linear programming relaxation of the problem and finds a feasible solution to its dual problem. According to the weak duality theorem, this solution is a lower bound for the solution of the primal linear programming problem and hence also for the solution of the original integer programming problem.

In contrast to the bound in Theorem 3.4, the bound in Theorem 3.5 is not a constant factor approximation guarantee but depends on the set $\mathcal{V}$ and the function F. In [109], Wolsey shows that if F is available only in the form of a value oracle, then no polynomial time algorithm can achieve a constant factor approximation guarantee for the constrained function value problem, unless $\mathbf{P} = \mathbf{NP}$.

3.5 Simultaneous Localization and Mapping

One of the fundamental problems in mobile robot navigation is that of simultaneous localization and mapping (SLAM) [102]. For a mobile robot that moves through a previously unknown environment, the SLAM problem is the problem of mapping the environment while simultaneously localizing itself in this very map.

The SLAM problem can be considered in two different ways: The first one is called the online SLAM problem, where the goal is to estimate the posterior distribution

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \tag{3.48}$$

of the robot's momentary pose $\mathbf{x}_t$ and the map $\mathbf{m}$, given a set of robot motion commands $\mathbf{u}_{1:t}$ and a set of feature observations $\mathbf{z}_{1:t}$. The recursive Bayesian state estimation techniques presented in Section 3.3 can be adapted to estimate this distribution at every (discretized) time step t during operation of the robot (see [64, 97]).

The second SLAM problem is the so-called full SLAM problem. In contrast to online SLAM, the goal in full SLAM is to estimate the distribution of the entire path $\mathbf{x}_{0:T}$ of the robot and the map $\mathbf{m}$, instead of just the current pose $\mathbf{x}_t$. Concretely, full SLAM can be defined as the problem of estimating the joint posterior distribution

$$p(\mathbf{x}_{0:T}, \mathbf{m}_{1:n}, c_{1:T} \mid \mathbf{u}_{1:T}, \mathbf{z}_{1:T}) \,. \tag{3.49}$$

Here, we assume that the map $\mathbf{m}$ consists of n features $\mathbf{m}_{1:n}$. Furthermore, $c_{1:T} (= c_{1:T}(\mathbf{z}_{1:T}))$ refers to the data associations, i. e., the identities of the map features perceived in the observations $\mathbf{z}_{1:T}$. We explicitly take the data associations into account in our formulation of the full SLAM problem, as estimating them correctly is crucial for the accuracy of the estimation of the other variables $\mathbf{x}_{0:T}$ and $\mathbf{m}_{1:n}$.

Methods for solving the full SLAM problem [17, 30, 38] often estimate the data associations $c_{1:T}$ in a first step and then, given the estimated data associations, they estimate the path of the robot $\mathbf{x}_{0:T}$ and the map $\mathbf{m}_{1:n}$. In the following, we first describe the problem of estimating the data associations and then give a brief introduction to graph-based approaches to solve the full SLAM problem.

3.5.1 Data Association in SLAM

In the context of the SLAM problem, data association refers to the problem of identifying a map feature $\mathbf{m}_i$ in one observation $\mathbf{z}_{t_1}$ as the very same feature found in another observation $\mathbf{z}_{t_2}$. Unless the robot is able to recognize previously visited places, its position uncertainty increases without bound due to the accumulating odometry error. In the full SLAM formulation, integrating out the unknown data associations $c_{1:T}$ in Equation (3.49) leads to

$$p(\mathbf{x}_{0:T}, \mathbf{m}_{1:n} \mid \mathbf{u}_{1:T}, \mathbf{z}_{1:T}) = \sum_{c_1} \sum_{c_2} \dots \sum_{c_T} p(\mathbf{x}_{0:T}, \mathbf{m}_{1:n}, c_{1:T} \mid \mathbf{u}_{1:T}, \mathbf{z}_{1:T}) \,. \tag{3.50}$$

Consequently, the number of possible data associations grows exponentially with the number of observations.

Most approaches to data association are based on the innovation and its covariance, i. e., the difference between the actual observation $\mathbf{z}_t$ and the predicted observation $\hat{\mathbf{z}}_t$ under a given data association $c_t(\mathbf{z}_t)$. If the innovation is a Gaussian, the squared Mahalanobis distance $D_M^2(\mathbf{z}_t, \hat{\mathbf{z}}_t)$ is distributed according to the χ_d^2 distribution, where d is the dimensionality of the innovation. One of the most popular approaches to data association is the nearest neighbor filter. It computes a set of compatible candidate features and accepts only features whose innovation is within a certain region of the χ^2 distribution. It then chooses the candidate feature that best matches the observation.

There are more sophisticated data association techniques than the nearest neighbor filter [76, 79], which can handle significantly more challenging environments. However, even these approaches cannot guarantee to avoid false positives, particularly in the presence of perceptual ambiguities.

3.5.2 Graph-Based Approaches to Solve the SLAM Problem

Graph-based approaches to solve the SLAM problem model the poses of the robot and the positions of observed landmarks as nodes in a graph. The edges of such a graph correspond to spatial constraints between the individual nodes. These constraints arise from odometry measurements and from landmark observations. Graph-based approaches to the full SLAM problem are typically divided into a front end and a back end. The front end interprets the sensor data to extract spatial constraints. To do so, the data association problem has to be addressed. Using the solution of the data association problem, the back end finds the configuration of the nodes that best matches the extracted spatial constraints by applying an optimization technique [37, 46, 61]. In the back end, a key precondition for the successful computation of the map is getting the correct data associations from the front end.

In Chapter 9, we present an approach to improving the data association performance

in SLAM by deploying artificial landmarks. Our graph-based implementation of this approach uses the framework by Kümmerle *et al.* [61] as a back end.

3.6 Actor-Critic Monte Carlo Reinforcement Learning

In reinforcement learning [100], an agent interacts with its environment to learn how to behave so as to maximize a numerical reward. Formally, a reinforcement learning problem is given by a set $\mathfrak{S}$ of states, a set $\mathfrak{A}$ of actions that the agent may perform at discrete time steps t, transition probabilities that describe how the states change in response to the agent's actions, and a reward function $r : \mathfrak{S} \times \mathfrak{A} \to \mathbb{R}$ that determines the numerical reward that the agent receives for executing action $a \in \mathfrak{A}$ in state $s \in \mathfrak{S}$. A policy is a probability distribution $\pi(s, a) = p(a \mid s)$ of choosing action a given the agent is in state s. For each episode e, which is a sequence $(s_0, a_0, \ldots, s_T, a_T)$ of states and actions, the return $R(s_t, a_t)$ for executing action a_t in state s_t is given by the sum of the rewards gathered after the execution of a_t:

$$R(s_t, a_t) = \sum_{t'=t+1}^{T} r(s_{t'}, a_{t'}) \,, \tag{3.51}$$

which is sometimes also weighted with a discount factor. The goal of reinforcement learning is to find the policy $\pi^\star(s, a)$ that maximizes the expected return

$$Q^\pi(s, a) = \mathbb{E}_\pi[R(s_t, a_t) \mid s_t = s, a_t = a] \tag{3.52}$$

for all $s \in \mathfrak{S}$ and $a \in \mathfrak{A}$.

One class of reinforcement learning methods is Monte Carlo reinforcement learning. Monte Carlo reinforcement learning methods do not require prior knowledge of the environment's dynamics, i. e., the probability distributions of the transitions, to compute the optimal policy. Instead, these methods estimate $Q^\pi(s, a)$ using the return averaged over a number of sample episodes e. First-visit-only Monte Carlo learning takes into account only the first occurrence of each state-action pair (s, a) in each episode e, leading to the estimator

$$\hat{Q}^\pi(s, a) = \frac{1}{n_\mathcal{F}} \sum_{e \in \mathcal{F}(s,a)} R^e_{\text{first}}(s, a) \,, \tag{3.53}$$

where $\mathcal{F}(s, a) = \{e \mid (s, a) \in e\}$, $n_\mathcal{F} = |\mathcal{F}(s, a)|$, and $R^e_{\text{first}}(s, a)$ is the return at the first occurrence of (s, a) in episode e. In Monte Carlo reinforcement learning, the estimator $\hat{Q}^\pi$ converges to Q^π if all state-action pairs occur with non-zero probability when following the policy π. A common way to satisfy this condition is to use a so-called softmax policy of the form

$$\pi(s, a) = p(a \mid s) = \frac{\exp(\hat{Q}^\pi(s, a)/\tau)}{\sum_{a' \in \mathcal{A}} \exp(\hat{Q}^\pi(s, a')/\tau)} \,, \tag{3.54}$$

where $\tau \in \mathbb{R}^+$ is the so-called temperature.

One way to update the estimated Q-function $\hat{Q}^\pi$ in Monte Carlo reinforcement learning is the so-called *actor-critic* approach. In actor-critic learning, two Q-functions are stored separately. One works as the actor and is used for sampling the episodes. The other is the critic and is updated according to the rewards gained in the episodes. After the critic has observed enough episodes, it changes its role and becomes actor. At this moment, the old actor is discarded and a new critic is initialized.

Unlike the standard on-policy learning, which adjusts the policy for episode generation after every episode based on the updated estimate of the Q-function, actor-critic learning does not change the policy while learning a Q-function. This is favorable because the final estimate for the Q-function is not distorted by values generated with an outdated policy, which is the case in on-policy learning for the values integrated early in the learning process.

In Chapter 9, we apply first-visit-only actor-critic Monte Carlo reinforcement learning with a softmax policy to learn in which states a mobile robot should autonomously deploy artificial landmarks to optimize its performance in SLAM.

Chapter 4

Landmark Placement for Localization

In this chapter, we consider the problem of placing landmarks for optimizing the localization performance of a mobile robot that travels frequently on similar trajectories. The presented approach maximizes the information gain of the states of the robot resulting from its landmark observations. We prove that this maximization problem is NP-hard and apply a greedy algorithm to approximate its solution. Building on the concept of submodularity, we derive a tight constant-factor bound on the approximation error of this algorithm. For evaluating the information gain, we apply Monte Carlo simulation, which can deal with arbitrary system dynamics and control modes of the robot. Furthermore, we evaluate the placed landmark sets in extensive experiments both in simulation and with a real robot.

In this chapter, we consider landmark placement for mobile robot localization as an optimization problem. For a given trajectory, we place a given number n of landmarks so that the mutual information between all states of the robot along the trajectory and all landmark observations is maximized. The goal is to arrive at a landmark configuration that yields the highest information gain. We prove that this problem is NP-hard, and introduce a greedy algorithm that approximates its solution. Using the concept of submodularity introduced in the background chapter, we prove that our algorithm finds near-optimal sets of landmarks. In order to deal with non-linear system dynamics and different types of robot controllers, our algorithm employs Monte Carlo simulation to evaluate the objective function. We evaluate the placed landmark sets in simulation and on a real robot. The results of the experiments show that our algorithm significantly outperforms baseline approaches.

This chapter is organized as follows. In the following section, we introduce the landmark placement problem. We then present our approximation algorithm and show a tight bound on the approximation error in Section 4.2. Afterwards, in Section 4.3, we

discuss different models for controlling the robot and describe their influence on the properties of the approximation. Finally we present extensive experiments in which we evaluate our algorithm in simulation as well as with a real robot.

4.1 Problem Definition

In this section, we define the landmark placement problem that we consider in this chapter. Since this definition builds on the information theoretic concepts described in Section 3.2, we use the notation introduced in that section and explicitly distinguish between random variables, which we denote with capital letters (e.g., $\mathbf{X}_t$), and the values that they can take on, for which we use their lower-case counterparts (e.g., $\mathbf{x}_t$).

As described above, we consider landmark placement as a trajectory dependent problem. The evolution of the state $\mathbf{x}_t$ at time $t \in [0, T]$ of the robot executing a given trajectory is subject to random noise, which is induced by the random errors occurring in the execution of the control commands $\mathbf{u}_{1:t}$. To account for this randomness, we model the states $\mathbf{x}_{0:T}$ of the robot, the control commands $\mathbf{u}_{1:T}$, and the observations $\mathbf{z}_{1:T}^{1:N}$ of the landmarks $\boldsymbol{\ell}_{1:N}$ with the random variables $\mathbf{X}_{0:T}$, $\mathbf{U}_{1:T}$, and $\mathbf{Z}_{1:T}^{1:N}$, respectively. As described in Section 3.3, the dependencies of these random variables can be described by a Hidden Markov Model (HMM). For the task of placing landmarks, the HMM has to explicitly include all possible landmark positions $\boldsymbol{\ell}_{1:N}$ in the environment and the separate observations $\mathbf{Z}_t^{1:n}$ of the different landmarks at time t. Without the dashed arrows, Figure 4.1 shows the dynamic Bayesian network that describes this HMM. In the literature on mobile robot localization [2, 102], the control command $\mathbf{U}_t$ in this network is for simplicity typically considered to be randomly chosen.

For every environment and every kind of landmark detecting sensor, there exists a subspace of the environment in which it is possible to place landmarks. Here, we use a discrete representation $\mathcal{V} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_N\}$ of this subspace. On the power set of this discrete set, we define an objective function $F : \mathcal{P}(\mathcal{V}) \to \mathbb{R}$. For every subset $\mathcal{A} \in \mathcal{V}$, $F(\mathcal{A})$ describes the value of information gained by placing landmarks in all points in $\mathcal{A}$. Given a constrained number n of landmarks that the robot can use for localization, we aim at finding

$$\mathcal{A}_n^\star = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A}) \,. \tag{4.1}$$

In this chapter, we consider the number n of landmarks to be placed as given (e.g., by the constrained memory of the robot), but one could also use a threshold as in [71] to determine n.

We aim at placing the landmarks whose observations contain the most information about the states of the robot. Therefore, we use the mutual information between the states of the robot and the landmark observations in the Hidden Markov Model as objective

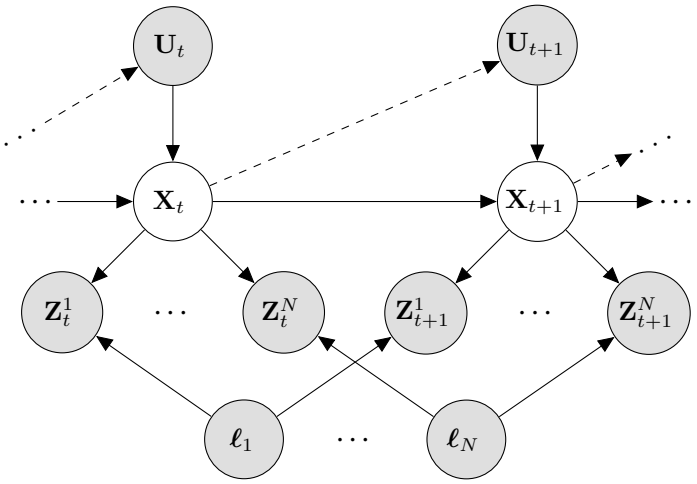

Figure 4.1: The dynamic Bayesian network for the localization of a mobile robot. It characterizes the dependencies between the random variables describing the controls $\mathbf{U}_t$, states $\mathbf{X}_t$, and measurements $\mathbf{Z}_t^k$ of the landmarks $\boldsymbol{\ell}_k$. The latent variables are shown in white and the observable variables in gray. The dashed arrows model the additional dependency for an external controller.

function F. For details on the information theoretic concept of mutual information, see Section 3.2. In the HMM, placing landmark $\boldsymbol{\ell}_k$ is equivalent to observing the values $\mathbf{z}_{1:T}^k$ of the random variables $\mathbf{Z}_{1:T}^k$. Note that $\mathbf{Z}_t^k$ can also take one special value indicating that $\boldsymbol{\ell}_k$ is not visible at time t (e.g., if $\boldsymbol{\ell}_k$ is not selected for placement). Using the notation $\mathbf{Z}_{1:T}^{\mathcal{A}} := \{\mathbf{Z}_{1:T}^i \mid \boldsymbol{\ell}_i \in \mathcal{A}\}$, the objective function has the following form:

$$\begin{aligned} F(\mathcal{A}) &= I(\mathbf{X}_{0:T}; \mathbf{Z}_{1:T}^{\mathcal{A}} \mid \mathbf{U}_{1:T}, \boldsymbol{\ell}_{1:N}) \\ &= h(\mathbf{X} \mid \mathbf{U}, \boldsymbol{\ell}_{1:N}) - h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}}, \boldsymbol{\ell}_{1:N}) \\ &= h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}}) \,. \end{aligned} \tag{4.2}$$

Note that the positions $\boldsymbol{\ell}_k$ of the landmarks are deterministic and globally known for all landmarks in $\mathcal{V}$, so conditioning on them does not change the considered probability distributions. Also, in Equation (4.2) and in the following, we omit the indices $0:T$ and $1:T$ for convenience if we consider the full sequences of states, observations, and controls, respectively. The objective function in Equation (4.2) describes how the entropy of the joint probability distribution of all the states of the robot along the trajectory is reduced by the placed set of landmarks (resp. their observations). Since the entropy can be viewed as a measure of uncertainty for the underlying distribution, this objective function is an intuitive choice to us.

Algorithm 5 Greedy landmark placement for entropy reduction

Input: $\mathcal{V}$, n
Output: $\mathcal{A}_n$
  $\mathcal{A}_0 \leftarrow \varnothing$
  for $i = 1$ **to** n **do**
    $\boldsymbol{\ell}^\star \leftarrow \underset{\boldsymbol{\ell} \in \mathcal{V}}{\operatorname{argmin}}\ h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}_{i-1} \cup \{\boldsymbol{\ell}\}})$
    $\mathcal{A}_i \leftarrow \mathcal{A}_{i-1} \cup \{\boldsymbol{\ell}^\star\}$
  end for
  return $\mathcal{A}_n$

4.2 Approximation Algorithm

As a result of Theorem 4.4 in this section, the problem defined in Equation (4.1) is NP-hard. Assuming that $\mathbf{P} \neq \mathbf{NP}$, there exists no deterministic polynomial time algorithm to determine the exact maximum of F. Therefore we have to be content with an approximation algorithm. Placing the landmark subset that maximizes the mutual information is equivalent to finding

$$\mathcal{A}_n^\star = \underset{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n}{\operatorname{argmin}}\ h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}})\,. \tag{4.3}$$

This follows directly from the definition of the objective function in Equation (4.2). Algorithm 5 approximates $\mathcal{A}_n^\star$ greedily using this equivalence. The complexity of the algorithm is in $\mathcal{O}(n|\mathcal{V}|)$, which makes it applicable even for large values of $|\mathcal{V}|$.

4.2.1 Submodularity of Conditional Mutual Information

To derive a constant factor approximation guarantee for Algorithm 5, we use the concept of submodularity, which we have described in detail in Section 3.4. Theorem 3.4 in that section gives a factor $(1 - 1/e)$ approximation guarantee for greedy algorithms for objective functions that satisfy the following three properties:

- $F(\varnothing) = 0$
- submodularity, i.e.,

$$\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V},\ \boldsymbol{\ell} \notin \mathcal{B}: \quad F(\mathcal{A} \cup \{\boldsymbol{\ell}\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{\boldsymbol{\ell}\}) - F(\mathcal{B}) \tag{4.4}$$

- monotonicity, i.e.,

$$\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}: \qquad F(\mathcal{A}) \leq F(\mathcal{B})\,. \tag{4.5}$$

In the landmark placement application, submodularity is an intuitive property. It means that the increase in information about the state of the robot resulting from adding a new landmark is lower or equal when more landmarks have already been placed.

All proofs in this section follow the concepts laid out by Krause and Guestrin [53]. The key difference between their approach and ours is that they consider a set of discrete random variables and aim at selecting a subset of this set, which then can be directly observed in a deterministic way. Our approach, on the other hand, considers random variables on continuous spaces and therefore applies the differential entropy instead of the discrete entropy. Furthermore, we do not select the states $\mathbf{X}_t$ directly for observation, but we select landmark positions $\boldsymbol{\ell}$. The observations $\mathbf{Z}^{\{\boldsymbol{\ell}\}}$ of our selected landmarks do not reveal the values $\mathbf{x}_t$ of the variables $\mathbf{X}_t$ deterministically, they only indirectly measure $\mathbf{x}_t$ and are prone to stochastic noise. Despite these differences, we can reproduce the main results of Krause and Guestrin [53] for our approach by closely following their techniques:

Theorem 4.1. $F(\mathcal{A}) = I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U})$ *is submodular.*

Proof. Applying the definition of the conditional entropy from Equation (3.22) two times yields

$$\begin{aligned} I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U}) &= h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}}) && (4.6)\\ &= h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{X}, \mathbf{Z}^{\mathcal{A}}, \mathbf{U}) + h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U}) && (4.7)\\ &= h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{Z}^{\mathcal{A}} \mid \mathbf{X}, \mathbf{U}) - h(\mathbf{X}, \mathbf{U}) + h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U})\,. && (4.8) \end{aligned}$$

In the Hidden Markov Model described above, it holds that for every $\mathcal{A}$, $\mathbf{Z}^{\mathcal{A}}$ is a set of random variables that are mutually independent conditioned on $\mathbf{X}$. This Markov property is a direct application of the rules of d-separation [16] on the dynamic Bayesian network depicted in Figure 4.1. Since the joint entropy of mutually independent random variables is the sum of the individual entropies, Equation (4.8) becomes

$$I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U}) = h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{X}, \mathbf{U}) + h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U}) - \sum_{\boldsymbol{\ell} \in \mathcal{A}} h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{U})\,. \qquad (4.9)$$

Here $h(\mathbf{X} \mid \mathbf{U}) - h(\mathbf{X}, \mathbf{U})$ does not depend on $\mathcal{A}$ and $\sum_{\boldsymbol{\ell} \in \mathcal{A}} h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{U})$ adds the same value to both sides of the submodularity inequation stated in (4.4). So to prove the submodularity of $I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U})$, it suffices to prove the submodularity of $h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U})$. We do this by applying the definition of the conditional entropy in the same way as in Equation (4.6). $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V},\ \boldsymbol{\ell} \notin \mathcal{B}$:

$$\begin{aligned} h(\mathbf{Z}^{\mathcal{A} \cup \{\boldsymbol{\ell}\}}, \mathbf{U}) - h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U}) &= && (4.10)\\ h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{Z}^{\mathcal{A}}, \mathbf{U}) &\geq h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{Z}^{\mathcal{B}}, \mathbf{U}) && (4.11)\\ &= h(\mathbf{Z}^{\mathcal{B} \cup \{\boldsymbol{\ell}\}}, \mathbf{U}) - h(\mathbf{Z}^{\mathcal{B}}, \mathbf{U})\,. && (4.12) \end{aligned}$$

The above inequation holds because of the "information never hurts"-principle defined in Equation (3.25). □

Theorem 4.2. *$F(\mathcal{A}) = I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U})$ is monotonically increasing in $\mathcal{A}$.*

Proof. Using the definition of the conditional entropy and the "information never hurts"-principle as in the proof above and applying the conditional independence property of $\mathbf{Z}^{\{\boldsymbol{\ell}\}}$ and $\mathbf{U}$ conditioned on $\mathbf{X}$ in the HMM, it holds that $\forall \mathcal{A} \subseteq \mathcal{V}$, $\boldsymbol{\ell} \notin \mathcal{A}$:

$$
\begin{aligned}
F(\mathcal{A} \cup \{\boldsymbol{\ell}\}) - F(\mathcal{A}) &\overset{(4.2)}{=} h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}}) - h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A} \cup \{\boldsymbol{\ell}\}}) && (4.13)\\
&\overset{(4.9)}{=} h(\mathbf{Z}^{\mathcal{A} \cup \{\boldsymbol{\ell}\}}, \mathbf{U}) - h(\mathbf{Z}^{\mathcal{A}}, \mathbf{U}) - h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{U}) && (4.14)\\
&= h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{Z}^{\mathcal{A}}, \mathbf{U}) - h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{U}) && (4.15)\\
&\geq h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{Z}^{\mathcal{A}}, \mathbf{U}) - h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}, \mathbf{U}) && (4.16)\\
&= h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}) - h(\mathbf{Z}^{\{\boldsymbol{\ell}\}} \mid \mathbf{X}) && (4.17)\\
&= 0\,. && (4.18)
\end{aligned}
$$

□

Theorems 4.1 and 4.2 provide the properties that are required for the following bound:

Theorem 4.3. *For the result $\mathcal{A}_n$ of Algorithm 5, the following approximation guarantee holds:*

$$F(\mathcal{A}_n) \geq (1 - 1/e) \max_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A})\,. \tag{4.19}$$

Proof. As can be seen in Equation (4.2), $F(\varnothing) = 0$ holds obviously. This fact, together with Theorems 4.1 and 4.2 provides all preconditions to apply Theorem 3.4 on our objective function, which directly produces the desired result. □

For the objective function considered in this chapter, this bound is a tight bound:

Theorem 4.4. *The optimization problem defined in Equation (4.1) is not approximable in polynomial time within a constant factor better than $(1 - 1/e)$, unless $\mathbf{P} = \mathbf{NP}$.*

Proof. We will reduce the so-called max-cover problem to our problem of maximizing the mutual information between the (continuous) random variables $\mathbf{X}$ and $\mathbf{Z}^{\{\boldsymbol{\ell}\}}$, which is defined in Equation (4.1). Given a set $\mathcal{S} = \{1, \ldots, k\}$ and a collection of subsets $S_1, \ldots, S_m$ of $\mathcal{S}$, max-cover is the problem of selecting the n subsets whose union contains the maximum number of elements. Feige [31] has shown that there exists no polynomial time algorithm that approximates max-cover with a constant factor better than $(1 - 1/e)$, unless $\mathbf{P} = \mathbf{NP}$.

For each instance of the max-cover problem, we create an instance of our optimization problem in the following way: Consider a sequence of one-dimensional robot states $\mathbf{X}_{0:k}$

that do not depend on each other, but only on the control commands $\mathbf{U}_{1:k}$. We assume a simplistic motion model that is defined by the equation $\mathbf{x}_t = \mathbf{u}_t + \mathbf{v}_t$, where $\mathbf{v}_t$ is a sample from a normal distribution with mean zero and variance

$$Q_t = \exp(1 - \log(2\pi)) \,. \tag{4.20}$$

Furthermore, we assume that there exists a set $\mathcal{V} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m\}$ of m possible landmark locations. We assume that the landmark $\boldsymbol{\ell}_i$ can be observed at time $t \in \{1, \ldots, k\}$ if and only if

$$(t \in S_i) \wedge (t \notin \bigcup_{j<i} S_j) \tag{4.21}$$

where S_i is the i-th subset of $\mathcal{S}$ defined by the instance of the max-cover problem. If $\boldsymbol{\ell}_i$ can be observed at time t, then we assume that the observation $\mathbf{z}_t^{\{\boldsymbol{\ell}_i\}}$ directly measures the state $\mathbf{x}_t$ according to the sensor model $\mathbf{z}_t^{\{\boldsymbol{\ell}_i\}} = \mathbf{x}_t + \mathbf{w}_t$, with $\mathbf{w}_t$ being a sample from a normal distribution with zero mean and variance

$$R_t = \frac{\exp(1 - \log(2\pi))}{\exp(2) - 1} \,. \tag{4.22}$$

Note that the described random variables and dependencies form a (simple) instance of the class of problems that can be described with the Bayesian network in Figure 4.1. The motion model and the sensor model in this problem instance are linear functions. Therefore, as described in Section 3.3, the Kalman filter is the optimal estimator for the state of the system. The matrices required in the Kalman filter update defined in Algorithm 2 in that section are in fact scalars with the following values:

$$\mathbf{c}_t = 0 \,, \quad A_t = 0 \,, \quad B_t = 1 \,, \quad V_t = 1 \,, \quad H_t = 1 \,, \quad W_t = 1 \,, \quad \mathbf{d}_t = 0 \,, \tag{4.23}$$

and Q_t and R_t as defined above. Plugging these variables into Algorithm 2 results in

$$\Sigma_t = \begin{cases} \exp(1 - \log(2\pi)) & \text{if no landmark can be observed at time } t \\ \exp(-1 - \log(2\pi)) & \text{if one landmark can be observed at time } t \,. \end{cases} \tag{4.24}$$

More than one landmark can never be observed at the same time step due to the definition of observability in Formula (4.21). Since Σ_t is the variance of $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}^{\mathcal{A}})$ and the differential entropy h of a one-dimensional Gaussian can be computed as $\frac{1}{2} + \frac{1}{2}\log(2\pi\Sigma_t)$, it follows that

$$\begin{aligned} &h(\mathbf{X}_t \mid \mathbf{U}_{1:t} = \mathbf{u}_{1:t}, \mathbf{Z}_{1:t}^{\mathcal{A}} = \mathbf{z}_{1:t}^{\mathcal{A}}) \\ &\quad = \begin{cases} 1 & \text{if no landmark in } \mathcal{A} \text{ can be observed at time } t \\ 0 & \text{if one landmark in } \mathcal{A} \text{ can be observed at time } t \,. \end{cases} \end{aligned} \tag{4.25}$$

Since in the system constructed in this proof, the above entropy does not depend on the concrete values of $\mathbf{u}_{1:t}$ and $\mathbf{z}^{\mathcal{A}}_{1:t}$, but only on the (deterministic) landmark observability defined in Equation (4.21), it holds that

$$\begin{aligned}&\int h(\mathbf{X}_t \mid \mathbf{U}_{1:t} = \mathbf{u}_{1:t}, \mathbf{Z}^{\mathcal{A}}_{1:t} = \mathbf{z}^{\mathcal{A}}_{1:t})\, p(\mathbf{u}_{1:t}, \mathbf{z}^{\mathcal{A}}_{1:t})\, \mathrm{d}(\mathbf{u}_{1:t}, \mathbf{z}^{\mathcal{A}}_{1:t}) \\ &\quad = h(\mathbf{X}_t \mid \mathbf{U}_{1:t} = \mathbf{u}_{1:t}, \mathbf{Z}^{\mathcal{A}}_{1:t} = \mathbf{z}^{\mathcal{A}}_{1:t})\,. \end{aligned} \tag{4.26}$$

Therefore, with Equation (3.24), it follows that

$$h(\mathbf{X}_t \mid \mathbf{U}_{1:t}, \mathbf{Z}^{\mathcal{A}}_{1:t}) = \begin{cases} 1 & \text{if no landmark in } \mathcal{A} \text{ can be observed at time } t \\ 0 & \text{if one landmark in } \mathcal{A} \text{ can be observed at time } t\,. \end{cases} \tag{4.27}$$

In the above construction of the instance of the landmark placement problem, we assume independence properties that yield that the random variables $\mathbf{X}_t$ are independent from each other, also when conditioned on the controls and observations. Therefore, in this instance, the differential entropy of the joint distribution of all robot states is the sum of the individual differential entropies, which are independent of the controls and observations at later time steps:

$$h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}, \mathbf{Z}^{\mathcal{A}}_{1:k}) = \sum_{t=0}^{k} h(\mathbf{X}_t \mid \mathbf{U}_{1:k}, \mathbf{Z}^{\mathcal{A}}_{1:k}) = \sum_{t=0}^{k} h(\mathbf{X}_t \mid \mathbf{U}_{1:t}, \mathbf{Z}^{\mathcal{A}}_{1:t})\,. \tag{4.28}$$

Applying Equation (4.27) and Formula (4.21) now yields

$$h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}, \mathbf{Z}^{\mathcal{A}}_{1:k}) = h(\mathbf{X}_0) + \sum_{t=1}^{k} (1 - \mathbb{1}(\exists i \mid \boldsymbol{\ell}_i \in \mathcal{A} \wedge t \in S_i))\,, \tag{4.29}$$

where $\mathbb{1}$ is the indicator function. With the notation $\mathcal{B} = \{S_i \mid \boldsymbol{\ell}_i \in \mathcal{A}\}$, it follows that

$$\min_{\mathcal{A}} h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}, \mathbf{Z}^{\mathcal{A}}_{1:k}) = h(\mathbf{X}_0) + \min_{\mathcal{A}} \sum_{t=1}^{k} (1 - \mathbb{1}(\exists i \mid \boldsymbol{\ell}_i \in \mathcal{A} \wedge t \in S_i)) \tag{4.30}$$

$$= h(\mathbf{X}_0) + \min_{\mathcal{B}} \sum_{t=1}^{k} (1 - \mathbb{1}(\exists S_i \in \mathcal{B} \mid t \in S_i)) \tag{4.31}$$

$$= h(\mathbf{X}_0) + k - \max_{\mathcal{B}} \sum_{t=1}^{k} \mathbb{1}(\exists S_i \in \mathcal{B} \mid t \in S_i) \tag{4.32}$$

$$= h(\mathbf{X}_0) + k - \max_{\mathcal{B}} \Big| \bigcup_{S_i \in \mathcal{B}} S_i \Big|\,, \tag{4.33}$$

and therefore

$$\max_{\mathcal{A}} I(\mathbf{X}_{0:k}; \mathbf{Z}^{\mathcal{A}}_{1:k} \mid \mathbf{U}_{1:k}) \tag{4.34}$$

$$= h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}) - \min_{\mathcal{A}} h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}, \mathbf{Z}^{\mathcal{A}}_{1:k}) \tag{4.35}$$

$$= h(\mathbf{X}_{0:k} \mid \mathbf{U}_{1:k}) - h(\mathbf{X}_0) - k + \max_{\mathcal{B}} \Big| \bigcup_{S_i \in \mathcal{B}} S_i \Big| \tag{4.36}$$

$$= h(\mathbf{X}_0) + \sum_{t=1}^{k} h(\mathbf{X}_t \mid \mathbf{U}_{1:t}) - h(\mathbf{X}_0) - k + \max_{\mathcal{B}} \Big| \bigcup_{S_i \in \mathcal{B}} S_i \Big| \tag{4.37}$$

$$= \max_{\mathcal{B}} \Big| \bigcup_{S_i \in \mathcal{B}} S_i \Big| . \tag{4.38}$$

Now assume that there exists a polynomial time algorithm that finds a set $\mathcal{A}^\dagger$ of landmarks that approximates the solution to our mutual information maximization problem within a constant factor better than $(1 - 1/e)$. Then, according to Equation (4.38), the set $\mathcal{B}^\dagger = \{S_i \mid \ell_i \in \mathcal{A}^\dagger\}$ approximates the solution to the max-cover problem within a constant factor better than $(1 - 1/e)$, which is a contradiction to the hardness-of-approximation result for the max-cover problem proven by Feige [31]. □

Theorem 4.4 states that for the information gain in the Bayesian network from Figure 4.1, no other polynomial time algorithm can have a better worst case approximation guarantee than the greedy algorithm that we apply.

4.2.2 Entropy Calculation for the Joint Distribution

In our greedy landmark placement algorithm, we need to evaluate conditional differential entropies of the form $h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}})$. Applying Equations (3.23) and (3.24) on $h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}})$ results in

$$h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}}) \tag{4.39}$$

$$= -\int\int p(\mathbf{x} \mid \mathbf{u}, \mathbf{z}^{\mathcal{A}}) \log p(\mathbf{x} \mid \mathbf{u}, \mathbf{z}^{\mathcal{A}}) \, \mathrm{d}\mathbf{x} \; p(\mathbf{u}, \mathbf{z}^{\mathcal{A}}) \, \mathrm{d}(\mathbf{u}, \mathbf{z}^{\mathcal{A}}) \tag{4.40}$$

$$= \int h(\mathbf{X} \mid \mathbf{U} = \mathbf{u}, \mathbf{Z}^{\mathcal{A}} = \mathbf{z}^{\mathcal{A}}) \; p(\mathbf{u}, \mathbf{z}^{\mathcal{A}}) \, \mathrm{d}(\mathbf{u}, \mathbf{z}^{\mathcal{A}}) \; , \tag{4.41}$$

As can be seen from the equation, calculating $h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}})$ includes solving a $(T \cdot (\dim(\mathbf{Z}^{\mathcal{A}}_t) + \dim(\mathbf{U}_t)))$-dimensional integral. Because $\mathbf{Z}^{\mathcal{A}}$ and $\mathbf{U}$ are sets of highly correlated random variables, in general there exists no closed form solution for this integral.

In this chapter, we deal with this problem by applying Monte Carlo simulations to obtain an approximation $\hat{h}$ of the entropy $h(\mathbf{X} \mid \mathbf{U}, \mathbf{Z}^{\mathcal{A}})$. We calculate $\hat{h}$ as the average over N simulated runs of the robot. In each run of the simulation, we compute the differential

entropy $h(\mathbf{X}_{0:T} \mid \mathbf{U}_{1:T} = \mathbf{u}_{1:T}, \mathbf{Z}^{\mathcal{A}}_{1:T} = \mathbf{z}^{\mathcal{A}}_{1:T})$ of the joint posterior distribution of all robot states $\mathbf{X}_{0:T}$. As can be seen from Equation (4.40), we need the corresponding density $p(\mathbf{x}_{0:T} \mid \mathbf{u}_{1:T}, \mathbf{z}^{\mathcal{A}}_{1:T})$ to do so. To efficiently compute this density from the simulated states, controls, and observations, we use the well-known factorization:

$$
\begin{aligned}
&p(\mathbf{x}_{0:T} \mid \mathbf{u}_{1:T}, \mathbf{z}^{\mathcal{A}}_{1:T}) && (4.42)\\
&= \eta_T \, p(\mathbf{z}^{\mathcal{A}}_T \mid \mathbf{x}_T) \, p(\mathbf{x}_{0:T} \mid \mathbf{u}_{1:T}, \mathbf{z}^{\mathcal{A}}_{1:T-1}) && (4.43)\\
&= \eta_T \, p(\mathbf{z}^{\mathcal{A}}_T \mid \mathbf{x}_T) \, p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, \mathbf{u}_T) \, p(\mathbf{x}_{0:T-1} \mid \mathbf{u}_{1:T}, \mathbf{z}^{\mathcal{A}}_{1:T-1}) && (4.44)\\
&= \eta_T \, p(\mathbf{z}^{\mathcal{A}}_T \mid \mathbf{x}_T) \, p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, \mathbf{u}_T) \, p(\mathbf{x}_{0:T-1} \mid \mathbf{u}_{1:T-1}, \mathbf{z}^{\mathcal{A}}_{1:T-1}) && (4.45)\\
&= \dots \\
&= p(\mathbf{x}_0) \prod_{t=1}^{T} \eta_t \, p(\mathbf{z}^{\mathcal{A}}_t \mid \mathbf{x}_t) \, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \,, && (4.46)
\end{aligned}
$$

where (4.43) is an application of Bayes' theorem, (4.44) holds true by definition of the conditional density, (4.45) uses a Markov property resulting from d-separation on the Bayesian network depicted in Figure 4.1 (without the dashed arrows), and (4.46) follows from a straightforward induction applying the above steps as induction step. See Equation (11.9) in [102] for a more detailed derivation of this factorization. In Equation (4.46), $p(\mathbf{z}^{\mathcal{A}}_t \mid \mathbf{x}_t)$ is the sensor model and $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ is the motion model of the robot. The value η_t is a normalizing constant.

The Monte Carlo approximation $\hat{h}$ of h of course increases the approximation error in Algorithm 5. However, the approximation guarantee in Theorem 4.3 can be extended to account for this error. To achieve this, we first introduce Lemma 4.1, which deals with approximations with deterministic bounds, and then extend this result in Theorem 4.5 to approximations with probabilistic bounds with specified confidence values.

Lemma 4.1. *Suppose that there exists a function $\hat{F}$ that approximates F with an absolute error of at most $\frac{\epsilon}{2n}$ for a given threshold ϵ. Then using $\hat{F}$, Algorithm 5 returns a set $\hat{\mathcal{A}}_n$ for which*

$$
F(\hat{\mathcal{A}}_n) \geq (1 - 1/e) \max_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A}) \; - \epsilon \,. \tag{4.47}
$$

Proof. For $i \leq n$, we define $\hat{\mathcal{A}}_i$ as the set of landmarks placed by Algorithm 5 if $\hat{F}$ is used instead of F. When the algorithm places the i-th landmark, the worst case that can happen is that $\hat{F}$ assigns the best landmark

$$
\boldsymbol{\ell}^\star = \operatorname*{argmax}_{\boldsymbol{\ell} \in \mathcal{V}} F(\hat{\mathcal{A}}_{i-1} \cup \{\boldsymbol{\ell}\}) \tag{4.48}
$$

a value that is $\frac{\epsilon}{2n}$ too small while assigning another landmark $\boldsymbol{\ell}'$ a value that is $\frac{\epsilon}{2n}$ too high, i.e., $\hat{F}(\boldsymbol{\ell}^\star) = F(\boldsymbol{\ell}^\star) - \frac{\epsilon}{2n}$ and $\hat{F}(\boldsymbol{\ell}') = F(\boldsymbol{\ell}') + \frac{\epsilon}{2n}$. Therefore, even in the worst

case, it holds true that

$$F(\hat{\mathcal{A}}_i) - F(\hat{\mathcal{A}}_{i-1}) \geq \max_{\boldsymbol{\ell} \in \mathcal{V}} (F(\hat{\mathcal{A}}_{i-1} \cup \{\boldsymbol{\ell}\}) - F(\hat{\mathcal{A}}_{i-1})) - \frac{\epsilon}{n}\,. \tag{4.49}$$

The following steps follow the original proof of the approximation guarantee without errors by Nemhauser *et al.* [78] (Theorem 3.4 in the background chapter) and relax it according to Equation (4.49) where needed. We define

$$\mathcal{A}_n^\star = \max_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A}) \tag{4.50}$$

and assume without loss of generality that $\mathcal{A}_n^\star = \{\boldsymbol{\ell}_1^\star, \ldots, \boldsymbol{\ell}_n^\star\}$, i.e., that $|\mathcal{A}_n^\star| = n$. Same as in the original proof of Theorem 3.4, we use the monotonicity of F, a straightforward telescopic sum, and the definition of submodularity, which yields $\forall i < n$:

$$\begin{aligned}
&F(\mathcal{A}_n^\star) && (4.51)\\
&\leq F(\mathcal{A}_n^\star \cup \hat{\mathcal{A}}_i) && (4.52)\\
&= F(\hat{\mathcal{A}}_i) + \sum_{j=1}^{n} \Big(F(\hat{\mathcal{A}}_i \cup \{\boldsymbol{\ell}_1^\star, \ldots, \boldsymbol{\ell}_j^\star\}) - F(\hat{\mathcal{A}}_i \cup \{\boldsymbol{\ell}_1^\star, \ldots, \boldsymbol{\ell}_{j-1}^\star\}) \Big) && (4.53)\\
&\leq F(\hat{\mathcal{A}}_i) + \sum_{j=1}^{n} \Big(F(\hat{\mathcal{A}}_i \cup \{\boldsymbol{\ell}_j^\star\}) - F(\hat{\mathcal{A}}_i) \Big)\,. && (4.54)
\end{aligned}$$

In contrast to the original proof, we now apply Equation (4.49). Additionally, we define $\delta_i = F(\mathcal{A}_n^\star) - F(\hat{\mathcal{A}}_i)$. With these two properties, it follows that

$$\begin{aligned}
F(\mathcal{A}_n^\star) &\leq F(\hat{\mathcal{A}}_i) + \sum_{j=1}^{n} \Big(F(\hat{\mathcal{A}}_{i+1}) - F(\hat{\mathcal{A}}_i) + \frac{\epsilon}{n} \Big) && (4.55)\\
\Leftrightarrow F(\mathcal{A}_n^\star) - F(\hat{\mathcal{A}}_i) &\leq n \Big(F(\hat{\mathcal{A}}_{i+1}) - F(\hat{\mathcal{A}}_i) \Big) + \epsilon && (4.56)\\
\Leftrightarrow \quad \delta_i &\leq n\,(\delta_i - \delta_{i+1}) + \epsilon && (4.57)\\
\Leftrightarrow \quad \delta_{i+1} &\leq (1 - 1/n)\delta_i + \frac{\epsilon}{n}\,. && (4.58)
\end{aligned}$$

It follows by induction that

$$\delta_n \leq (1 - 1/n)^n\, \delta_0 + \sum_{j=0}^{n-1} \Big((1 - 1/n)^j\, \frac{\epsilon}{n} \Big)\,. \tag{4.59}$$

Now applying the fact that $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$ (see, e.g., 4.2.29 in Abramowitz and Stegun [1]) yields

$$\delta_n \leq \frac{1}{e}\, \delta_0 + \sum_{j=0}^{n-1} \Big((1 - 1/n)^j\, \frac{\epsilon}{n} \Big) \leq \frac{1}{e}\, \delta_0 + n\, \frac{\epsilon}{n} = \frac{1}{e}\, \delta_0 + \epsilon\,. \tag{4.60}$$

By the definition of δ_i, it holds that $\delta_n = F(\mathcal{A}_n^\star) - F(\hat{\mathcal{A}}_n)$ and $\delta_0 = F(\mathcal{A}_n^\star) - F(\varnothing) = F(\mathcal{A}_n^\star)$. Finally, plugging these values in Equation (4.60) results in

$$(1 - 1/e)F(\mathcal{A}_n^\star) \leq F(\hat{\mathcal{A}}_n) + \epsilon\,. \tag{4.61}$$

□

Theorem 4.5. *Suppose that a function $\hat{F}$ approximates the objective function F with an absolute error of at most $\frac{\epsilon}{2n}$ with probability at least $1 - \frac{\delta}{n|\mathcal{V}|}$ for given thresholds ϵ and δ. Then using $\hat{F}$, Algorithm 5 returns a set $\hat{\mathcal{A}}_n$ for which*

$$F(\hat{\mathcal{A}}_n) \geq (1 - 1/e) \max_{\mathcal{A} \subseteq \mathcal{V}; |\mathcal{A}| \leq n} F(\mathcal{A}) \; - \epsilon \tag{4.62}$$

with probability at least $1 - \delta$.

Proof. We use the same notation as in the proof above, namely that for $i \leq n$, $\hat{\mathcal{A}}_i$ denotes the set of landmarks placed by Algorithm 5 if $\hat{F}$ is used instead of F. In contrast to the proof above, here the values of $\hat{F}$ are the results of random experiments. We denote the set of elementary events for which the final approximation bound from Equation (4.62) holds as $\mathcal{S}$, and its complement as $\mathcal{S}^c$. Furthermore, we denote the sets of elementary events for which the worst-case bounds on the individual gains from Equation (4.49) hold as $\mathcal{S}_i$, and their complements as $\mathcal{S}_i^c$.

From the worst-case bounds on the individual gains in the deterministic scenario (Equation (4.49)) we can deduce that if the final approximation bound from Equation (4.62) does not hold true, then there has to be at least one $i \in \{1, \ldots, n\}$ for which the individual worst-case bound is also not satisfied, i.e.,

$$F(\hat{\mathcal{A}}_i) - F(\hat{\mathcal{A}}_{i-1}) < \max_{\boldsymbol{\ell} \in \mathcal{V}} (F(\hat{\mathcal{A}}_{i-1} \cup \{\boldsymbol{\ell}\}) - F(\hat{\mathcal{A}}_{i-1})) - \frac{\epsilon}{n}\,. \tag{4.63}$$

Therefore, it holds true that

$$P(\mathcal{S}^c) \leq P\left(\bigcup_{i=1}^{n} \mathcal{S}_i^c\right) \tag{4.64}$$

By definition, $\mathcal{S}_i^c$ is the set of elementary events for which Equation (4.63) holds true. Algorithm 5 creates $\hat{\mathcal{A}}_i$ from $\hat{\mathcal{A}}_{i-1}$ by checking all landmarks in $\mathcal{V}$ and adding the one for which $\hat{F}$ is maximized. Therefore, it is a necessary (but not a sufficient) precondition for Equation (4.63) to hold true that there exists at least one landmark $\boldsymbol{\ell}$ in $\mathcal{V}$ for which

$$|\hat{F}(\hat{\mathcal{A}}_{i-1} \cup \{\boldsymbol{\ell}\}) - F(\hat{\mathcal{A}}_{i-1} \cup \{\boldsymbol{\ell}\})| > \frac{\epsilon}{2n}\,. \tag{4.65}$$

With the notation $\mathcal{S}_{\boldsymbol{\ell},i}^c$ for the set of elementary events for which Equation (4.65) holds true, it follows from Equation (4.64) that

$$P(\mathcal{S}^c) \leq P\left(\bigcup_{i=1}^{n} \bigcup_{\boldsymbol{\ell} \in \mathcal{V}} \mathcal{S}_{\boldsymbol{\ell},i}^c\right) \tag{4.66}$$

With the union bound and the fact that by definition $P(S^c_{\boldsymbol{\ell},i}) \leq \frac{\delta}{n|\mathcal{V}|}$, it follows that

$$P(\mathcal{S}^c) \leq \sum_{i=1}^{n} \sum_{\boldsymbol{\ell} \in \mathcal{V}} P(\mathcal{S}^c_{\boldsymbol{\ell},i}) \leq n|\mathcal{V}| \frac{\delta}{n|\mathcal{V}|} = \delta \,. \tag{4.67}$$

□

In the Monte Carlo simulation, the absolute error $\frac{\epsilon}{2n}$ and the confidence $1 - \frac{\delta}{n|\mathcal{V}|}$ of the approximation can be easily estimated via the standard deviation of the estimator for h. Hence, by increasing the number N of simulated runs, ϵ and δ in Equation (4.62) can be moved arbitrarily close to zero.

4.3 Control Model

In Section 4.1 we assumed the control commands $\mathbf{u}_{1:T}$ to be randomly chosen. However, in practice the robot often gets control commands by a human operator or operates autonomously, i.e., selects control commands depending on its current belief of the state. The control behaviors considered in this chapter can be classified into the following modes:

- *Random control*, where the control commands do not depend on any other random variable (see Figure 4.1 without dashed arrows).
- *External control*, where the control commands only depend on the last state of the robot, which is assumed to be known by the controller (see Figure 4.1 with the dased arrows).
- *Autonomous control*, where the control commands depend on the belief of the robot $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$, i.e., all previous control commands and observations.

Note that for approaches that only aim at localizing a robot in a given fixed environment, the random control mode can be assumed even for external or autonomous controllers, because it only discards the information on how the control commands are selected. Therefore, the random control assumption applied by many state-of-the-art localization approaches (see Thrun *et al.* [102] for an overview) only leads to less peaked posterior distributions. However, in our landmark placement method, the type of control mode is crucial for the submodularity of the mutual information. In the following we show the effect of the non-random control modes to our landmark placement method.

4.3.1 External Controls

The external control mode is modeled by the Bayesian network in Figure 4.1 with the dashed arrows. Theorem 4.1 still holds for this Bayesian network, because the Markov

property for $\mathbf{Z}^{\mathcal{A}}$ conditioned on $\mathbf{X}$ is still applicable. Therefore, also the tight Nemhauser-bound (i.e., Theorems 4.3 and 4.4) holds. In contrast to the random control mode, in the external control mode the Markov property that $p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1})$ does not hold true because of the additional dependency (dashed arrows) in the Bayesian network. Instead, with this dependency, applying Bayes' rule leads to

$$p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) = \eta \, p(\mathbf{u}_t \mid \mathbf{x}_{t-1}) \, p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) \,, \tag{4.68}$$

where $p(\mathbf{u}_t \mid \mathbf{x}_{t-1})$ models the policy of the controller. In the derivation of the factorization of the full posterior (Equation (4.45)), replacing the Markov property with this result yields the following extended factorization for the external control mode:

$$\begin{aligned} &p(\mathbf{x}_{0:T} \mid \mathbf{u}_{1:T}, \mathbf{z}^{\mathcal{A}}_{1:T}) \\ &\quad = p(\mathbf{x}_0) \prod_{t=1}^{T} \eta'_t \, p(\mathbf{z}^{\mathcal{A}}_t \mid \mathbf{x}_t) \, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \, p(\mathbf{u}_t \mid \mathbf{x}_{t-1}) \,. \end{aligned} \tag{4.69}$$

If the control model $p(\mathbf{u}_t \mid \mathbf{x}_{t-1})$ is known and normally distributed, then the entropy calculation can be easily extended to the external control mode.

4.3.2 Autonomous Controls

In the case of fully autonomous operation of the robot, each control $\mathbf{u}_t$ depends on the belief of the state $p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1})$, i.e., all previous control commands $\mathbf{u}_{1:t-1}$ and observations $\mathbf{z}^{\mathcal{A}}_{1:t-1}$. Extended by these dependencies, the Bayesian network no longer fulfills the Markov property for $\mathbf{Z}^{\mathcal{A}}$, i.e., it no longer holds true that the observations of two different landmarks are independent from each other conditioned on all robot states $\mathbf{X}$. However, this Markov property is an essential building block in the proof of the submodularity of the conditional mutual information (see Equation (4.9)). In fact, one can easily find a counter-example that contradicts the submodularity property for the conditional mutual information in the autonomous control mode.

Concretely, for autonomous controls the probability that landmarks further along the trajectory become visible to the robot depends on how well the robot is already localized, i.e., which landmarks it has observed before. So, adding a landmark to a small set of placed landmarks can result in a lower increase of conditional mutual information than adding the same landmark to a superset of the placed landmarks for which the probability of reaching this last landmark is higher. Such a landmark configuration obviously breaks the submodularity property. Since submodularity is a pre-condition for the approximation guarantee specified in Theorem 4.3, this guarantee does not hold true for the autonomous control mode. However, in the simulation experiments (see Section 4.4.1) the results of our greedy algorithm are still close to the optimal results even for autonomous controls.

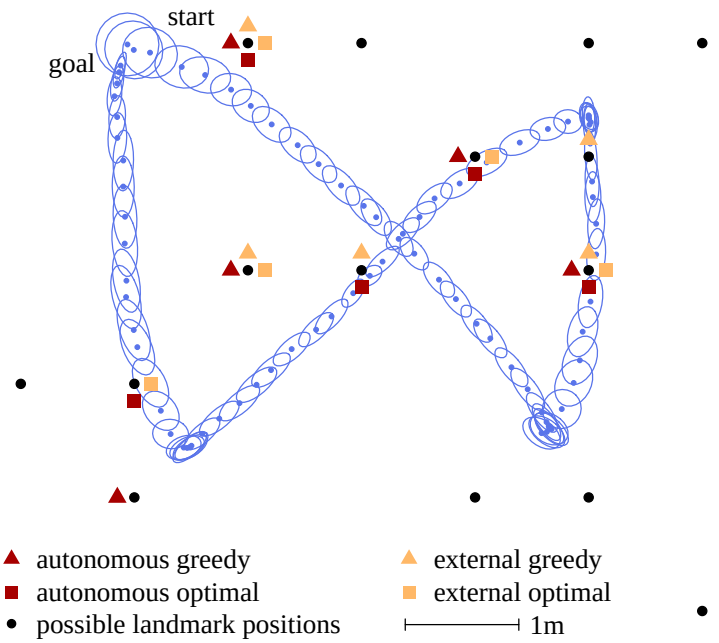

Figure 4.2: The landmarks placed by our greedy algorithm and optimal landmark placements for autonomous and external controls. The posterior means and 99% confidence ellipses of a sample run are plotted in blue.

4.4 Experimental Evaluation

To evaluate our approximation algorithm, we performed extensive experiments both in simulation and with a real robot. In the experiments, we consider a sensor that observes range and bearing to unique landmarks and has a limited field of view. The field of view is considered circular with radius $2\,\mathrm{m}$ in simulation, while for the real robot the rectangular field of view of its camera is used. For estimating the state of the robot in the individual runs of the Monte Carlos simulations, we use a variant of the unscented Kalman filter (UKF), a square-root unscented Kalman filter [70], which approximates the real probability distributions with normal distributions $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. This allows for a straightforward calculation of the differential entropy as $h(\mathcal{N}) = \log \sqrt{(2\pi e)^k |\Sigma|}$, where k is the dimension of the state space.

4.4.1 Simulation Experiments

We performed two different kinds of simulation experiments, each with $N = 10,000$ runs: In the first experiment, we compare the solution produced by Algorithm 5 to the optimal solution. We selected $n = 5$ out of $|\mathcal{V}| = 15$ possible landmark positions for a trajectory forming a pointed figure eight. This is the maximum problem size for which we were

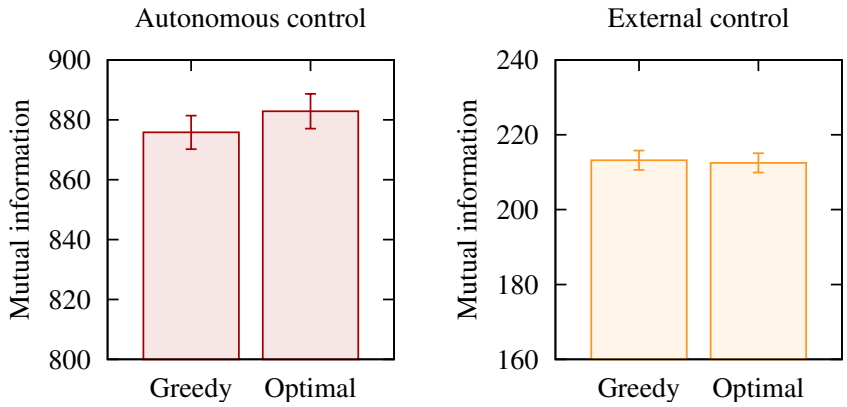

Figure 4.3: The mutual information values for the four landmark sets from Figure 4.2 together with the 95% confidence intervals of the Monte Carlo simulation.

able to determine the optimal solution via brute force. Figure 4.2 shows the landmark set $\hat{\mathcal{A}}_5$ placed by the greedy approach and the optimal landmark set $\hat{\mathcal{A}}_5^\star = \operatorname{argmax} \hat{F}(\mathcal{A})$ both for autonomous and external controls. In both control modes, the greedy strategies place two of the five landmarks on different positions than the brute force strategies. Two landmarks are placed on the same positions in all four cases. The mutual information values $\hat{F}(\mathcal{A})$ of the four landmark sets are shown in Figure 4.3. In this scenario, the approximation quality of the greedy algorithm is 99.1% of the value achieved by the brute force strategy for autonomous controls and 100.3% for external controls. The percentage higher than 100% is due to the standard deviation in the Monte Carlo simulation, which is displayed in the figure. So even for the case of autonomous controls, for which we did not derive a bound on the approximation error, the greedy algorithm performs well in the experiment. The absolute values in Figure 4.3 are much lower for the external control case, as $h(\mathbf{X} \mid \mathbf{U})$ already is much lower due to the additional integration of $p(\mathbf{u}_t \mid \mathbf{x}_{t-1})$ into the posterior distribution.

In the second set of simulation experiments, we evaluated our approach independently of a specific trajectory and a set of possible landmark positions $\mathcal{V}$. In order to achieve this abstraction, we randomly sampled ten trajectories and ten associated sets $\mathcal{V}$, each containing 100 landmarks, in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ environment. For each trajectory, we evaluated the placements for five and ten landmarks, so together we got 20 landmark placement tasks. Figure 4.4 displays the five first landmarks our algorithm placed for two of the ten sampled tasks and the placements for two handcrafted sweeping and fetch-and-return tasks.

In the sampled tasks, we evaluated our approach of greedily placing landmarks to maximize mutual information against three different approaches, namely randomly placing landmarks, no landmarks, and the uniqueness maximization algorithm [71]. The latter approach maximizes the average uniqueness over all possible states $\mathbf{x} \in \mathcal{X}$ of the robot.

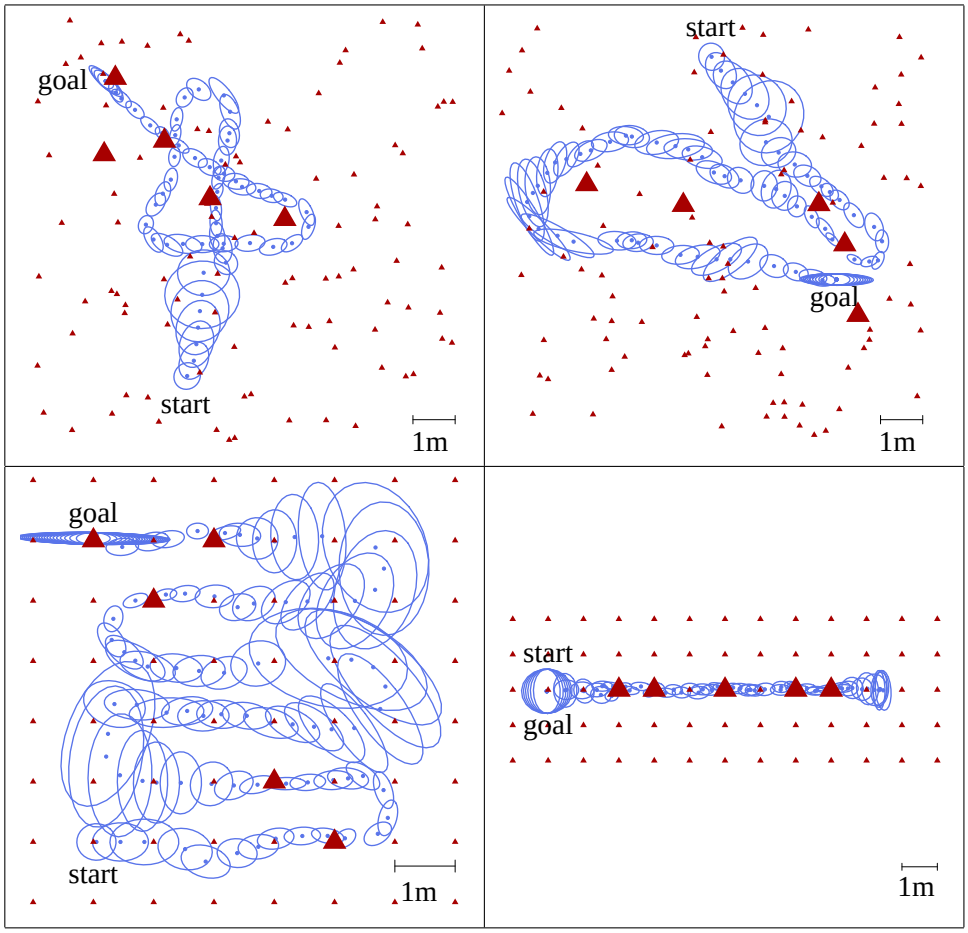

Figure 4.4: Four instances in which our algorithm placed five landmarks assuming autonomous controls. The placed landmarks are shown as big red triangles, the sets $\mathcal{V}$ as small red triangles, and the means and 99% confidence ellipses of one sample execution of the desired trajectories in blue. First row: two of the ten sampled tasks. Second row: a sweeping task and a fetch-and-return task.

	I	h_T	$\overline{d}$[m]	d_T[m]
Our approach	1121.11	-4.32	0.191	0.126
Uniqueness	791.11	-2.64	0.285	0.254
Random	697.09	-1.70	0.348	0.409
No landmarks	0.00	2.59	1.084	1.807

Table 4.1: Quantitative results of the comparison to baseline approaches

It defines the uniqueness of a pose $\mathbf{x}$ given a map $\mathbf{m}$ as

$$\mathcal{U}(\mathbf{x}, \mathbf{m}) = \left(\int_{\tilde{\mathbf{x}} \in \mathcal{X}} p(\mathbf{z}^{[\mathbf{x},\mathbf{m}]} \mid \tilde{\mathbf{x}}, \mathbf{m}) \, d\tilde{\mathbf{x}} \right)^{-1}, \tag{4.70}$$

where $\mathbf{z}^{[\mathbf{x},\mathbf{m}]}$ is the maximum likelihood observation at pose $\mathbf{x}$ given the map $\mathbf{m}$.

To evaluate the landmark placements, we consider four different measures of quality: the mutual information $I := I(\mathbf{X}; \mathbf{Z}^{\mathcal{A}} \mid \mathbf{U})$, the entropy at the final state $h_T := h(\mathbf{X}_T \mid \mathbf{U}_{1:T}, \mathbf{Z}^{\mathcal{A}}_{1:T})$, and the average and final distances $\overline{d}$ and d_T, respectively, between the true pose of the robot and the mean of the belief about the robot state. Table 4.1 summarizes the evaluation results for autonomous controls. Our approach based on mutual information maximization yields the best results for all four criteria. In paired sample t-tests, the differences between the results of our approach and the results of all other approaches were statistically significant at a 5% level. In fact, the highest p-value was 0.21%. For external controls, the differences in the criteria of quality are similar to the ones shown in Table 4.1. They are also all statistically significant at a 5% level.

For an experimental comparison between our approach and the other approaches described in this thesis, see Chapter 8.

4.4.2 Experiments with a Real Robot

To further validate the simulation results, we evaluated the four quality criteria also in experiments carried out with a real Pioneer P3-DX robot. The robot and the executed trajectory are shown in Figure 4.5. We used a standard webcam pointing towards the ceiling as a sensor for detecting unique ARToolkit markers [15]. The considered set $\mathcal{V}$ of possible landmark locations is disconnected by areas where lamps and supporting beams

	I	h_T	$\overline{d}$[m]	d_T[m]
Simulation	2278.09	-4.62	0.061	0.073
Reality	2570.94	-5.33	0.068	0.040

Table 4.2: Quantitative results in simulation and reality

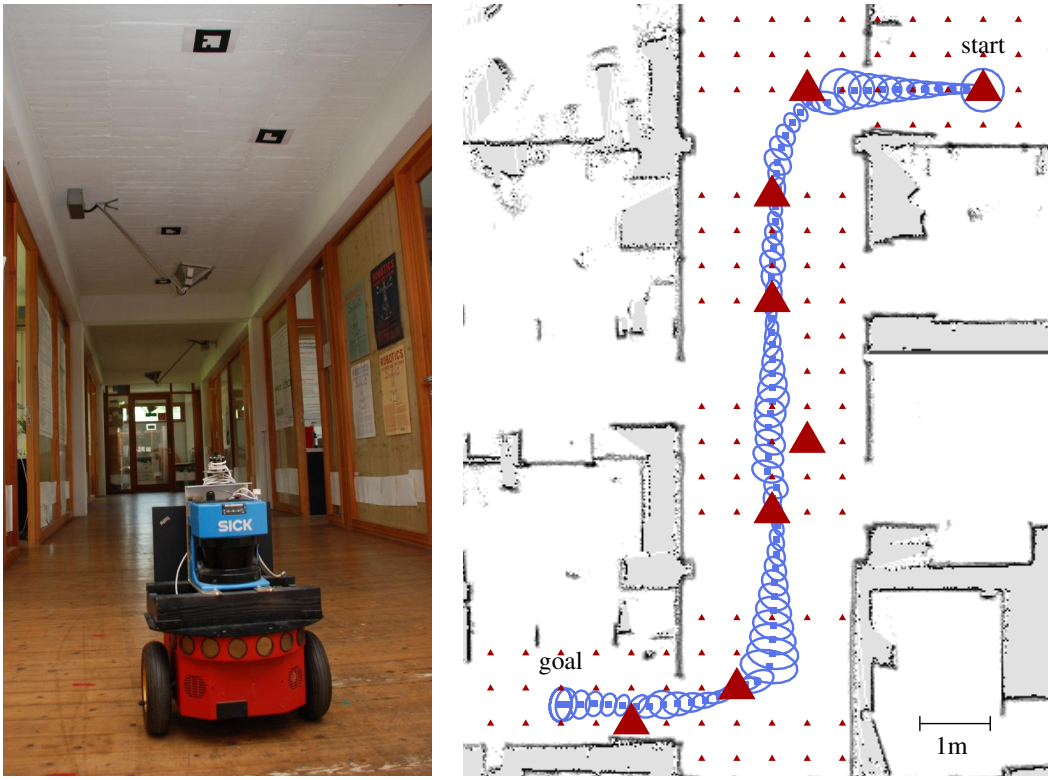

Figure 4.5: Eight landmarks placed by our algorithm (large red triangles). The set of all possible landmark locations is depicted as small red triangles. One execution of the trajectory, for which the landmarks were placed, is plotted in blue together with its 99% confidence ellipses. The picture shows the Pioneer P3-DX robot used in the experiments. It is equipped with a SICK LMS laser range finder (only used for reference) and a webcam pointing upwards. The markers on the ceiling correspond to the landmarks shown in the graph.

do not allow landmark placement (see Figure 4.5). To evaluate the criteria $\overline{d}$ and d_T, we obtained reference positions from laser-based Monte Carlo localization [102].

The eight landmarks depicted in Figure 4.5 were placed by our algorithm. Using these landmarks, the real robot executed ten autonomous runs. Table 4.2 shows the quality criteria for this scenario evaluated in simulation together with the average values of the ten real runs. Despite of the imperfect reference positions and remaining systematic errors the quality values in the real world experiments corresponded to the ones obtained in simulation.

4.5 Discussion

In this chapter, we presented a landmark placement method for mobile robot localization along pre-planned trajectories. We formulated landmark placement as the optimization

problem of maximizing the mutual information between the states of the robot and the observations of the landmarks and proved that this problem is NP-hard. Using the concept of submodularity, we derived a tight constant-factor bound on the error of our polynomial time approximation algorithm. For estimating the required mutual information values, our algorithm uses Monte Carlo simulations, which can deal with arbitrary system dynamics and control modes of the robot. Extensive experiments for different types of control modes demonstrate that our approach outperforms baseline approaches and works well in practice.

Chapter 5

Estimation of Expected Distributions for Mobile Robot Navigation

In the landmark placement algorithm presented in the previous chapter, we used Monte Carlo simulation to estimate the expected localization accuracy of a mobile robot. This method is applicable to general robotic systems but computationally demanding. In contrast to that, when considering the navigation accuracy of the robot, making assumptions about the type of robotic system allows for a highly efficient recursive estimation of the expected distributions of the deviation of the robot from its desired trajectory. In this chapter, we present a novel approach to efficiently estimating these expected distributions for a mobile robot that uses a linear-quadratic regulator to travel along a pre-defined trajectory. We exploit the structure of the stochastic dependencies in the navigation framework for reducing the dimensionality of the matrix multiplications in the calculation of the expected distributions by half, compared to the state of the art. This efficient calculation scheme is the main building block for evaluating the objective functions of the landmark placement methods presented in the next chapters.

The main challenge in landmark placement is to deal with the combinatorial nature of the problem of selecting the optimal subset of the set of all possible landmark positions. One way to deal with this challenge is to apply the efficient subset selection algorithms from submodular function optimization described in the background chapter. However, when searching for the optimal landmark configuration, even those algorithms need to evaluate the optimization criterion for a large number of candidate configurations. Therefore, the efficiency of the method for calculating the optimization criterion is essential for our landmark placement approaches.

In this chapter, we present a novel method to efficiently estimate the expected probability distributions of the deviation of a mobile robot from its desired trajectory with respect to its control commands and observations, assuming that the robot is controlled with linear-quadratic regulation (LQR) control [13]. In the literature [11, 82, 106], these expected distributions are often called a priori distributions. They are the main building block of the objective functions that we optimize in the landmark placement approaches presented in the following chapters.

In order to achieve the desired efficiency, our estimation method linearizes the model of the whole navigation cycle, including control, motion, observation, and localization, and recursively calculates the expected distributions in the linearized system. The derivation of our method builds on the fact that before the robot starts operation, the localization estimate $\boldsymbol{\mu}_t$ of the robot can be considered as a random variable, which is highly correlated with the random variable describing the state of the robot $\mathbf{x}_t$. We show that the structure of this correlation allows us to decouple the calculation of the covariances of $\boldsymbol{\mu}_t$ and $\mathbf{x}_t$. This enables us to recursively update the distributions of $\boldsymbol{\mu}_t$ and $\mathbf{x}_t$ individually, whereas the state of the art [11] recursively updates their joint distribution. Therefore, compared to [11], our approach reduces the dimensionality of the occurring matrix multiplications by half, which results in a substantial reduction of the runtime of the computations. In extensive experiments we demonstrate that our approach significantly outperforms state-of-the-art approaches in terms of runtime, while still producing exactly the same results.

This chapter is organized as follows: In the next section, we formally introduce expected distributions and present the linearization that we apply to the navigation system. Then, in Section 5.2, we derive our efficient estimation scheme for expected distributions and compare it to the state of the art. Finally, in Section 5.3 we present extensive experiments that show that our approach outperforms the state of the art also in practice.

5.1 Robotic System

We consider the problem of estimating the expected distributions of the states $\mathbf{x}_t$ of a mobile robot traveling along a pre-defined trajectory $\mathcal{T}$. Hereby, the trajectory $\mathcal{T} = (\mathbf{x}^\star_{0:T}, \mathbf{u}^\star_{1:T})$ is a time-discrete sequence specifying the desired robot state $\mathbf{x}^\star_t$ at each time step t and the desired control command $\mathbf{u}^\star_t$ that, if executed without noise, moves the robot from one desired state to the next. The actual robot state $\mathbf{x}_t$ changes over time according to the stochastic motion model

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{v}_t)\,, \tag{5.1}$$

where $\mathbf{u}_t$ is the actual control command executed at time t and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$ is the motion noise, which we assume to be Gaussian distributed. Due to the stochastic nature

of the motion model, the actual robot state $\mathbf{x}_t$ differs from the desired state $\mathbf{x}_t^\star$. To reduce this difference, the robot needs to execute a control command $\mathbf{u}_{t+1}$ that differs from the desired control command $\mathbf{u}_{t+1}^\star$. We assume that the controller $\mathcal{C}$ that selects the control commands is an LQR controller [13]. For each time step t, the LQR controller selects the control $\mathbf{u}_t$ that minimizes the expected quadratic error term

$$\mathbb{E}\Big[\sum_{\ell=t}^{t'}((\mathbf{x}_\ell - \mathbf{x}_\ell^\star)^T C(\mathbf{x}_\ell - \mathbf{x}_\ell^\star) + (\mathbf{u}_\ell - \mathbf{u}_\ell^\star)^T D(\mathbf{u}_\ell - \mathbf{u}_\ell^\star))\Big], \tag{5.2}$$

where C and D are positive definite weight matrices and t' is the maximum time step for which the LQR controller minimizes the quadratic errors. Since we always consider trajectories with fixed final time steps T, we set $t' = T$.

For localization, the robot has a map of the set $\mathcal{A}$ of all landmark positions in its environment and a sensor that takes noisy observations $\mathbf{z}_t$ of these landmarks according to a sensor model

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{w}_t, \mathcal{A})\,, \tag{5.3}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, R_t)$ is the sensor noise.

5.1.1 Expected Distributions

As described in the background chapter, for mobile robot localization one typically applies some kind of filter, e.g., a Kalman filter [48] or a particle filter [102], to estimate the posterior probability distributions $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ of the states of the robot, which are conditioned on the already executed controls and observations. In contrast to that, at the time of landmark placement, the concrete control commands and observations of the robot are not yet known. Therefore, for landmark placement, we are interested in the expected distributions of the states $\mathbf{x}_t$ of the robot with respect to the control commands and observations. The only information about the distributions of the states $\mathbf{x}_t$ of the robot that we have available at the time of landmark placement stems from the desired trajectory $\mathcal{T}$ of the robot, the controller $\mathcal{C}$ that selects the control commands $\mathbf{u}_t$, the motion model f, the sensor model h, and the positions $\mathcal{A}$ of the landmarks in the environment. The density $p(\mathbf{x}_t \mid \mathcal{T}, \mathcal{C}, f, h, \mathcal{A})$ of the expected distribution for the time step t is the integral over the posterior density $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ at time t:

$$\begin{aligned} &p(\mathbf{x}_t \mid \mathcal{T}, \mathcal{C}, f, h, \mathcal{A}) \\ &= \int p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathcal{T}, \mathcal{C}, f, h, \mathcal{A})\, p(\mathbf{u}_{1:t}, \mathbf{z}_{1:t} \mid \mathcal{T}, \mathcal{C}, f, h, \mathcal{A})\, \mathrm{d}(\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \qquad (5.4) \\ &= \int p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\, p(\mathbf{u}_{1:t}, \mathbf{z}_{1:t} \mid \mathcal{T}, \mathcal{C}, f, h, \mathcal{A})\, \mathrm{d}(\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \qquad (5.5) \end{aligned}$$

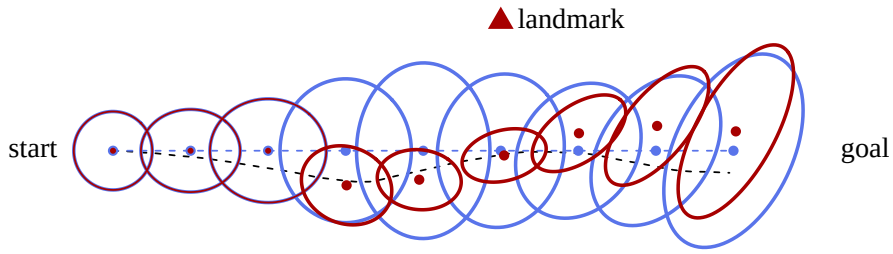

Figure 5.1: Example navigation task with one landmark. Shown are the desired trajectory (dashed blue line) and the actually executed trajectory (dashed black line) in one simulated run of the robot. Additionally, the figure shows the means and 99% confidence ellipses of the posterior distributions (red) in the simulated run and of the expected distributions (blue).

Equation (5.4) is an application of the law of total probability (Theorem 3.1). Equation (5.5) uses the Markov assumption that, if the concrete observations and controls are known, the controller and the desired trajectory do not add any additional information about the state of the robot. The posterior distribution of course still depends on the motion model, the sensor model, and the landmark positions, but in Chapter 3, in Equation (5.5), and in the literature [2, 35, 102], they are typically left out for convenience.

In the following chapters, we are mostly interested in the influence of the positions $\mathcal{A}$ of the landmarks on the expected distributions. Therefore, we use the shorthand notation

$$p(\mathbf{x}_t \mid \mathcal{A}) := p(\mathbf{x}_t \mid \mathcal{T}, \mathcal{C}, f, h, \mathcal{A}) \tag{5.6}$$

for the density of the expected distribution. Note that the variables that we omit, $\mathcal{T}$, $\mathcal{C}$, f, and h, are deterministically known and do not change in any step of the following derivations. Figure 5.1 visualizes the expected distributions and the posterior distributions of a simulated run of the robot in an example navigation task, in which the desired trajectory of the robot describes a straight line from the left to the right. The means of the posterior distributions are often used in mobile robot localization as estimates for the states of the robot. Therefore, the confidence regions of the posterior distributions in the figure can be considered a a visualization of the uncertainty in the localization of the robot. In contrast to that, the confidence regions of the expected distributions stem from the expectation over all possible executions of the trajectory and therefore can be seen as a way of describing the uncertainty in navigation.

In general, the expected distribution $p(\mathbf{x}_t \mid \mathcal{A})$ cannot be estimated in closed form. One solution that is often applied is to approximate the high-dimensional integral defined in Equation (5.6) via Monte-Carlo simulation. Monte-Carlo simulation [23] can deal with arbitrary controllers, motion models, and sensor models, but is computationally demanding.

In contrast to that, we efficiently estimate $p(\mathbf{x}_t \mid \mathcal{A})$ by linearizing the whole navigation system, consisting of observation, localization, control, and motion, resulting in

a Gaussian expected distribution that can be calculated efficiently via standard matrix manipulations.

5.1.2 Linearized System

For linearizing the motion model (5.1) and the sensor model (5.3), it is convenient to consider the deviations of the states, controls, and observations, from their desired or expected values instead of the absolute values themselves. Therefore, we define

$$\Delta \mathbf{x}_t := \mathbf{x}_t - \mathbf{x}_t^\star \,, \tag{5.7}$$

$$\Delta \mathbf{u}_t := \mathbf{u}_t - \mathbf{u}_t^\star \,, \tag{5.8}$$

$$\Delta \mathbf{z}_t := \mathbf{z}_t - h(\mathbf{x}_t^\star, \mathbf{0}) \,. \tag{5.9}$$

For the linearization procedure, we follow the approach by van den Berg *et al.* [11] and use first-order Taylor expansions around the desired trajectory $(\mathbf{x}_{0:T}^\star, \mathbf{u}_{1:T}^\star)$, leading to the approximate identities

$$\Delta \mathbf{x}_t \approx A_t \Delta \mathbf{x}_{t-1} + B_t \Delta \mathbf{u}_t + V_t \, \mathbf{v}_t \,, \tag{5.10}$$

$$\Delta \mathbf{z}_t \approx H_t \Delta \mathbf{x}_t + W_t \, \mathbf{w}_t \,. \tag{5.11}$$

with the Jacobians

$$A_t = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_{t-1}^\star, \mathbf{u}_t^\star, \mathbf{0}) \,, \tag{5.12}$$

$$B_t = \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_{t-1}^\star, \mathbf{u}_t^\star, \mathbf{0}) \,, \tag{5.13}$$

$$V_t = \frac{\partial f}{\partial \mathbf{m}}(\mathbf{x}_{t-1}^\star, \mathbf{u}_t^\star, \mathbf{0}) \,, \tag{5.14}$$

$$H_t = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}_t^\star, \mathbf{0}, \mathcal{A}) \,, \tag{5.15}$$

$$W_t = \frac{\partial h}{\partial \mathbf{n}}(\mathbf{x}_t^\star, \mathbf{0}, \mathcal{A}) \,. \tag{5.16}$$

The difference between this linearization and the one applied in the standard EKF described in Section 3.3.1 is that here, we linearize around the desired trajectory $(\mathbf{x}_{0:T}^\star, \mathbf{u}_{1:T}^\star)$, which is known before the robot starts operation, and instead of that the EKF linearizes around the filter means and control commands $(\boldsymbol{\mu}_{0:T}, \mathbf{u}_{1:T})$, which are unknown beforehand. This is the key idea by van den Berg *et al.* [11] that makes a recursive computation of expected distributions possible in the first place.

As described in the background chapter, in the linearized system considered here, the Kalman filter is the optimal estimator for the posterior distribution, leading to Gaussian densities

$$p(\Delta \mathbf{x}_t \mid \Delta \mathbf{u}_{1:t}, \Delta \mathbf{z}_{1:t}) \sim \mathcal{N}(\Delta \boldsymbol{\mu}_t, \Sigma_t) \tag{5.17}$$

for all time steps $t \in [0, T]$. Adjusting the recursive update equations of the Kalman filter to the linearized system yields

$$\overline{\Delta\boldsymbol{\mu}_t} = A_t \Delta\boldsymbol{\mu}_{t-1} + B_t \Delta\mathbf{u}_t \tag{5.18}$$

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + V_t Q_t V_t^T \tag{5.19}$$

$$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T)^{-1} \tag{5.20}$$

$$\Delta\boldsymbol{\mu}_t = \overline{\Delta\boldsymbol{\mu}_t} + K_t(\Delta\mathbf{z}_t - H_t \overline{\Delta\boldsymbol{\mu}_t}) \tag{5.21}$$

$$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t \,. \tag{5.22}$$

The mean $\Delta\boldsymbol{\mu}_t$ of the posterior distribution depends via $\Delta\mathbf{u}_t$, $\Delta\mathbf{z}_t$, and $\Delta\boldsymbol{\mu}_{t-1}$ on the actual values of $\mathbf{u}_{1:t}$ and $\mathbf{z}_{1:t}$ (see Equations (5.18), and (5.21)). Hence, when calculating the expected distributions, it is not yet available. However, the covariance Σ_t and the Kalman gain K_t depend, via the Jacobians, on $\mathbf{x}^\star_{0:t}$ and $\mathbf{u}^\star_{1:t}$ but not on the actual values of $\mathbf{u}_{1:t}$ and $\mathbf{z}_{1:t}$ (see Equations (5.19), (5.20), and (5.22)). Therefore they can be calculated before the robot starts operation.

Applied on the mean $\Delta\boldsymbol{\mu}_{t-1}$ in the Kalman filter, the LQR controller selects motion commands $\mathbf{u}_t$ according to

$$\Delta\mathbf{u}_t = L_t \Delta\boldsymbol{\mu}_{t-1} \,, \tag{5.23}$$

where L_t is the feedback matrix that minimizes the quadratic error defined in Equation (5.2). With the Jacobians defined in Equation (5.16), also the feedback matrices L_t can be calculated before the robot starts operating via the recursive update rule resulting from the discrete-time Riccati equation [13]

$$E_T = C \,, \tag{5.24}$$

$$\forall \ell \in [t, T]: \quad L_\ell = -(B_{\ell+1}^T E_{\ell+1} B_{\ell+1} + D)^{-1} B_{\ell+1}^T E_{\ell+1} A_{\ell+1} \,, \tag{5.25}$$

$$E_\ell = C + A_{\ell+1}^T E_{\ell+1} A_{\ell+1} + A_{\ell+1}^T E_{\ell+1} B_{\ell+1} L_\ell \,. \tag{5.26}$$

Here, C and D are the weight matrices of the LQR controller defined in Equation (5.2) and E_t is an auxiliary variable.

Summing up, we express the whole navigation cycle, which consists of executing a motion command, making an observation, localizing, and selecting the next motion command depending on the localization, by linear functions.

5.2 Expected Distributions in Linearized Systems

In this section, we present our novel method to efficiently calculate expected distributions via recursion for the above-described linearized robotic system.

5.2.1 Efficient Calculation Scheme

For the derivation of our efficient calculation scheme, the mean $\Delta\boldsymbol{\mu}_t$ of the posterior distribution plays a key role. Before the robot starts operating, i.e., before making any observations and selecting any motion commands, $\Delta\boldsymbol{\mu}_t$ can be considered as a random variable, which deterministically depends on $\mathbf{u}_{0:t-1}$ and $\mathbf{z}_{1:t}$. An example of the random behavior of $\Delta\boldsymbol{\mu}_t$ can be seen in Figure 5.1. To prove that the mean of the expected distribution of $\Delta\mathbf{x}_t$ equals zero for all time steps $t \in [0, T]$, we consider the dependencies between $\Delta\boldsymbol{\mu}_t$ and $\Delta\mathbf{x}_t$, taking into account the LQR controller.

Lemma 5.1. *Assuming that $\Delta\mathbf{x}_0$ is zero-mean Gaussian distributed, the expected distributions of $\Delta\mathbf{x}_t$ and of $\Delta\boldsymbol{\mu}_t$ are Gaussians with zero mean , i.e., $\forall t \in [0, T]$:*

$$p(\Delta\mathbf{x}_t \mid \mathcal{A}) \sim \mathcal{N}(\mathbf{0}, S_t)\,, \tag{5.27}$$

$$p(\Delta\boldsymbol{\mu}_t \mid \mathcal{A}) \sim \mathcal{N}(\mathbf{0}, M_t)\,. \tag{5.28}$$

Proof. We first prove that the means of $\Delta\mathbf{x}_t$ and $\Delta\boldsymbol{\mu}_t$ are zero. Plugging the equation for the LQR control selection (5.23) into the linearized motion model (5.10) leads to

$$\Delta\mathbf{x}_t = A_t\Delta\mathbf{x}_{t-1} + B_tL_t\Delta\boldsymbol{\mu}_{t-1} + V_t\,\mathbf{v}_t\,. \tag{5.29}$$

Since $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$, and the expectation is a linear operator, it follows that

$$\mathbb{E}[\Delta\mathbf{x}_t \mid \mathcal{A}] = A_t\,\mathbb{E}[\Delta\mathbf{x}_{t-1} \mid \mathcal{A}] + B_tL_t\,\mathbb{E}[\Delta\boldsymbol{\mu}_{t-1} \mid \mathcal{A}]\,. \tag{5.30}$$

To derive a similar formula for the expectation of $\Delta\boldsymbol{\mu}_t$, we plug Equation (5.18) into Equation (5.21), which yields

$$\Delta\boldsymbol{\mu}_t = A_t\Delta\boldsymbol{\mu}_{t-1} + B_t\Delta\mathbf{u}_t + K_t(\Delta\mathbf{z}_t - H_tA_t\Delta\boldsymbol{\mu}_{t-1} + B_t\Delta\mathbf{u}_t)\,. \tag{5.31}$$

Plugging in Equations (5.23) and (5.11) results in

$$\Delta\boldsymbol{\mu}_t = (A_t + B_tL_t - K_tH_tA_t - K_tB_tL_t)\Delta\boldsymbol{\mu}_{t-1} + K_tH_t\Delta\mathbf{x}_t + K_tW_t\,\mathbf{w}_t\,. \tag{5.32}$$

Again using the linearity of the expectation operator and the fact that $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, R_t)$, we get

$$\begin{aligned}&\mathbb{E}[\Delta\boldsymbol{\mu}_t \mid \mathcal{A}]\\ &\quad= (A_t + B_tL_t - K_tH_tA_t - K_tB_tL_t)\,\mathbb{E}[\Delta\boldsymbol{\mu}_{t-1} \mid \mathcal{A}] + K_tH_t\,\mathbb{E}[\Delta\mathbf{x}_t \mid \mathcal{A}]\,.\end{aligned} \tag{5.33}$$

Because of the assumption that $\mathbb{E}[\Delta\mathbf{x}_0 \mid \mathcal{A}] = \mathbf{0}$, also $\Delta\boldsymbol{\mu}_0 = \mathbf{0}$, and therefore also $\mathbb{E}[\Delta\boldsymbol{\mu}_0 \mid \mathcal{A}] = \mathbf{0}$. An induction with this as the base case and Equations (5.30) and (5.33) as inductive step yields that $\mathbb{E}[\Delta\mathbf{x}_t \mid \mathcal{A}] = \mathbf{0}$ and $\mathbb{E}[\Delta\boldsymbol{\mu}_t \mid \mathcal{A}] = \mathbf{0}$.

A second induction with $\Delta\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, S_0)$ and $\Delta\boldsymbol{\mu}_0 = \mathbf{0}$ as base case and with Equations (5.29) and (5.32) as inductive step finally shows that the expected distributions are Gaussians. □

Up until here, we loosely followed the approach by van den Berg *et al.* [11]. However, in the following, we apply different techniques to derive a more efficient way for computing the covariance S_t of the expected distribution.

Theorem 5.1. *The covariance S_t of the expected distribution of $\Delta\mathbf{x}_t$ is the sum of the covariance Σ_t of the posterior distribution of $\Delta\mathbf{x}_t$ and the covariance M_t of the mean $\Delta\boldsymbol{\mu}_t$ in the Kalman filter, when considered as a random variable, i.e., $\forall t \in [0, T]$:*

$$S_t = \Sigma_t + M_t \,. \tag{5.34}$$

Proof. From Lemma 5.1, we know that $\mathbb{E}[\Delta\mathbf{x}_t \mid \mathcal{A}] = \mathbf{0}$. Therefore, the covariance of $p(\Delta\mathbf{x}_t \mid \mathcal{A})$ is

$$S_t = \mathrm{Cov}(\Delta\mathbf{x}_t \mid \mathcal{A}) = \int \Delta\mathbf{x}_t \Delta\mathbf{x}_t^T \; p(\Delta\mathbf{x}_t \mid \mathcal{A}) \; \mathrm{d}\Delta\mathbf{x}_t \,. \tag{5.35}$$

Applying the law of total probability (Theorem 3.1) on $p(\Delta\mathbf{x}_t)$ yields

$$S_t = \int \Delta\mathbf{x}_t \Delta\mathbf{x}_t^T \int p(\Delta\mathbf{x}_t \mid \Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; p(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \, \mathrm{d}(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; \mathrm{d}\Delta\mathbf{x}_t \,. \tag{5.36}$$

Fubini's theorem [47] allows us to reorder the integrals:

$$S_t = \int\int \Delta\mathbf{x}_t \Delta\mathbf{x}_t^T \; p(\Delta\mathbf{x}_t \mid \Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; \mathrm{d}\Delta\mathbf{x}_t \; p(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; \mathrm{d}(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \,. \tag{5.37}$$

In the following, we use the shorthand notations

$$\mathrm{d}P_{\mathbf{x}} := p(\Delta\mathbf{x}_t \mid \Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; \mathrm{d}\Delta\mathbf{x}_t \,, \tag{5.38}$$

$$\mathrm{d}P_{\mathbf{u},\mathbf{z}} := p(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \; \mathrm{d}(\Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t}) \,. \tag{5.39}$$

Next, we add a zero to Equation (5.37):

$$\begin{aligned} S_t = \int\int \big(&(\Delta\mathbf{x}_t - \Delta\boldsymbol{\mu}_t)(\Delta\mathbf{x}_t - \Delta\boldsymbol{\mu}_t)^T + \Delta\mathbf{x}_t \Delta\boldsymbol{\mu}_t^T \\ &+ \Delta\boldsymbol{\mu}_t \Delta\mathbf{x}_t^T - \Delta\boldsymbol{\mu}_t \Delta\boldsymbol{\mu}_t^T \big) \mathrm{d}P_{\mathbf{x}} \; \mathrm{d}P_{\mathbf{u},\mathbf{z}} \,. \end{aligned} \tag{5.40}$$

By definition of the covariance it holds that $\Sigma_t = \int (\Delta\mathbf{x}_t - \Delta\boldsymbol{\mu}_t)(\Delta\mathbf{x}_t - \Delta\boldsymbol{\mu}_t)^T \; \mathrm{d}P_{\mathbf{x}}$, and therefore

$$S_t = \int \Big(\Sigma_t + \int \Delta\mathbf{x}_t \Delta\boldsymbol{\mu}_t^T \; \mathrm{d}P_{\mathbf{x}} + \int \Delta\boldsymbol{\mu}_t \Delta\mathbf{x}_t^T \; \mathrm{d}P_{\mathbf{x}} - \int \Delta\boldsymbol{\mu}_t \Delta\boldsymbol{\mu}_t^T \; \mathrm{d}P_{\mathbf{x}} \Big) \; \mathrm{d}P_{\mathbf{u},\mathbf{z}} \,. \tag{5.41}$$

Since $\Delta\boldsymbol{\mu}_t$ is the expected value of the posterior distribution of $\Delta\mathbf{x}_t$, it is by definition $\Delta\boldsymbol{\mu}_t = \int \Delta\mathbf{x}_t \, \mathrm{d}P_{\mathbf{x}}$, which is independent of $\Delta\mathbf{x}_t$ given the values of $\Delta\mathbf{u}_{1:t}$ and $\Delta\mathbf{z}_{1:t}$. Applying this definition on (5.41) and using the independence property to reorder the integrals yields

$$S_t = \int \left(\Sigma_t + \int \Delta\mathbf{x}_t \, \mathrm{d}P_{\mathbf{x}} \int \Delta\mathbf{x}_t^T \, \mathrm{d}P_{\mathbf{x}} \right) \mathrm{d}P_{\mathbf{u},\mathbf{z}} \,. \tag{5.42}$$

Again using the definition of $\Delta\boldsymbol{\mu}_t$ and the fact that the transpose is a linear transformation, which therefore can be moved out of the integral, yields

$$S_t = \int \Sigma_t + \Delta\boldsymbol{\mu}_t \Delta\boldsymbol{\mu}_t^T \, \mathrm{d}P_{\mathbf{u},\mathbf{z}} \,. \tag{5.43}$$

In the linearized system that we consider, Σ_t is independent of the values of $\Delta\mathbf{u}_{1:t}$ and $\Delta\mathbf{z}_{1:t}$ (see Equations (5.19), (5.20), and (5.22)). Therefore, we get

$$S_t = \Sigma_t + \int \Delta\boldsymbol{\mu}_t \Delta\boldsymbol{\mu}_t^T \, \mathrm{d}P_{\mathbf{u},\mathbf{z}} \,. \tag{5.44}$$

Since the random variable $\Delta\boldsymbol{\mu}_t$ is a deterministic function of the random variables $\Delta\mathbf{u}_{1:t}$ and $\Delta\mathbf{z}_{1:t}$, and since its expectation is $\mathbf{0}$, this results in

$$S_t = \Sigma_t + M_t \,. \tag{5.45}$$

□

This also has strong implications on the cross-covariance of $\Delta\mathbf{x}_t$ and $\Delta\boldsymbol{\mu}_t$:

Corollary 5.1. *Cov*$(\Delta\mathbf{x}_t, \Delta\boldsymbol{\mu}_t \mid \mathcal{A}) =$ *Cov*$(\Delta\boldsymbol{\mu}_t \mid \mathcal{A})$ $(= M_t)$.

Proof. The result follows from the proof of Theorem 5.1 by considering the transformations applied to $\Delta\mathbf{x}_t \Delta\boldsymbol{\mu}_t^T$ in Equation (5.40). □

The next corollary will be useful for comparing different measures for the quality of landmark sets:

Corollary 5.2.

$$tr(S_t) \geq tr(\Sigma_t) \,. \tag{5.46}$$

Proof. From Theorem 5.1 it follows that $tr(S_t) = tr(\Sigma_t) + tr(M_t)$. Since M_t is a covariance matrix and therefore positive semi-definite, its trace is greater or equal than zero, which finishes the proof. □

For being able to use Theorem 5.1 to calculate the covariance S_t of the expected distribution, we need to be able to calculate both Σ_t and M_t before the robot starts operation. How to calculate Σ_t beforehand in the linearized system is already clear from Equations (5.19), (5.20), and (5.22). In the following, we derive an efficient recursive update formula for the covariance M_t of $\Delta\boldsymbol{\mu}_t$. To do so, we first consider the covariance of the difference between $\Delta\mathbf{x}_t$ and the mean $\overline{\Delta\boldsymbol{\mu}_t}$ of the posterior distribution $p(\Delta\mathbf{x}_t \mid \Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t-1})$ before the integration of the observation $\Delta\mathbf{z}_t$.

Lemma 5.2.

$$\mathit{Cov}(\Delta\mathbf{x}_t - \overline{\Delta\boldsymbol{\mu}_t} \mid \mathcal{A}) = \mathit{Cov}(\Delta\mathbf{x}_t \mid \Delta\mathbf{u}_{1:t}, \Delta\mathbf{z}_{1:t-1}) = \overline{\Sigma}_t.$$

Proof. The second equation holds by definition of $\overline{\Sigma}_t$. The first equation follows directly from the construction of the Kalman filter as described by Kalman [48]. For linear systems like the one defined in Section 5.1.2, Kalman has proven that the filter mean $\overline{\Delta\boldsymbol{\mu}_t}$ is the minimum mean square error estimator for $\Delta\mathbf{x}_t$ and that the posterior covariance $\overline{\Sigma}_t$ equals the covariance of the estimation error $\Delta\mathbf{x}_t - \overline{\Delta\boldsymbol{\mu}_t}$ of this estimator. □

With this, we can now derive an efficient recursive update formula for the covariance M_t of $\Delta\boldsymbol{\mu}_t$.

Lemma 5.3.

$$M_0 = \mathbf{0}\,, \tag{5.47}$$
$$\overline{M}_t = (A_t + B_t L_{t-1}) M_{t-1} (A_t^T + L_{t-1}^T B_t^T)\,, \tag{5.48}$$
$$M_t = \overline{M}_t + K_t H_t \overline{\Sigma}_t\,, \tag{5.49}$$

where $M_t = \mathit{Cov}(\Delta\boldsymbol{\mu}_t \mid \mathcal{A})$ *and* $\overline{M}_t = \mathit{Cov}(\overline{\Delta\boldsymbol{\mu}_t} \mid \mathcal{A})$.

Proof. The initial belief in the Kalman filter for calculating the posterior covariance is deterministically given, therefore Equation (5.47) holds true. Plugging the LQR control policy from Equation (5.23) into the recursive update scheme for $\overline{\Delta\boldsymbol{\mu}_t}$ stated in Equation (5.18) yields

$$\overline{\Delta\boldsymbol{\mu}_t} = (A_t + B_t L_{t-1})\,\Delta\boldsymbol{\mu}_{t-1}\,. \tag{5.50}$$

Since the covariance is a bilinear form [47], this proves Equation (5.48). To prove Equation (5.49), we start by plugging Equation (5.11) into Equation (5.21), resulting in

$$\Delta\boldsymbol{\mu}_t = \overline{\Delta\boldsymbol{\mu}_t} + K_t H_t (\Delta\mathbf{x}_t - \overline{\Delta\boldsymbol{\mu}_t}) + K_t W_t \mathbf{w}_t, \tag{5.51}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, R_t)$. Again with the bilinearity of the covariance, this yields

$$M_t = \overline{M}_t + K_t H_t\, \mathrm{Cov}(\Delta\mathbf{x}_t - \overline{\Delta\boldsymbol{\mu}_t} \mid \mathcal{A})\, H_t^T K_t^T + K_t W_t R_t W_t^T K_t^T\,. \tag{5.52}$$

Algorithm 6 Recursive calculation of the covariances of the expected distributions

Input: $S_0(=\Sigma_0)$, $M_0 = \mathbf{0}$
Output: $S_{0:T}$

1: **for** $t = 1$ **to** T **do**
2: $\quad \overline{M}_t \leftarrow (A_t + B_t L_{t-1}) M_{t-1} (A_t^T + L_{t-1}^T B_t^T)$
3: $\quad \overline{\Sigma}_t \leftarrow A_t \Sigma_{t-1} A_t^T + V_t Q_t V_t^T$
4: $\quad K_t \leftarrow \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T)^{-1}$
5: $\quad M_t \leftarrow \overline{M}_t + K_t H_t \overline{\Sigma}_t$
6: $\quad \Sigma_t \leftarrow (I - K_t H_t) \overline{\Sigma}_t$
7: $\quad S_t \leftarrow \Sigma_t + M_t$
8: **end for**
9: **return** $S_{0:T}$

Applying Lemma 5.2 results in

$$M_t = \overline{M}_t + K_t H_t \overline{\Sigma}_t H_t^T K_t^T + K_t W_t R_t W_t^T K_t^T \tag{5.53}$$
$$= \overline{M}_t + K_t (H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T) K_t^T \,. \tag{5.54}$$

Next, we replace K_t^T according to its definition from Equation (5.20), leading to

$$M_t = \overline{M}_t + K_t (H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T)(H_t \overline{\Sigma}_t H_t^T + W_t R_t W_t^T)^{-1} H_t \overline{\Sigma}_t \tag{5.55}$$
$$= \overline{M}_t + K_t H_t \overline{\Sigma}_t \,, \tag{5.56}$$

which finishes the proof. □

This lemma is the last part needed to calculate the expected distribution of $\Delta\mathbf{x}_t$ (and therefore also the one of $\mathbf{x}_t$) via recursion. Summing up, we know from Lemma 5.1 that the expected distribution of $\Delta\mathbf{x}_t$ is $\mathcal{N}(\mathbf{0}, S_t)$. With Equation (3.7) from the background chapter, it follows that the expected distribution of $\mathbf{x}_t$ is $\mathcal{N}(\mathbf{x}_t^\star, S_t)$. Hence, the mean of the distribution is already known from the desired trajectory, and we only need to calculate the covariance S_t. We calculate S_t using the fact that $S_t = \Sigma_t + M_t$, as shown in Theorem 5.1. We know the recursive calculation schemes for Σ_t and for M_t from the equations on Page 60 and from Lemma 5.3, respectively. Both only depend on terms that can be computed before the robot starts operating. The complete recursive calculation of the covariance of the expected distribution can be seen in Algorithm 6. The input to the algorithm are the initial covariances S_0, Σ_0, and M_0. Since in the initial state, no previous control commands and observations are available, it holds that $S_0 = \Sigma_0$. From Lemma 5.3, we know that $M_0 = \mathbf{0}$.

5.2.2 Comparison to the State of the Art

To our knowledge, the first approach for calculating expected distributions of series of states of dynamic systems recursively was derived by van den Berg *et al.* [11]. In the same linearized system that we described above, they consider the joint expected distribution of $\Delta\mathbf{x}_t$ and $\Delta\boldsymbol{\mu}_t$, which is a Gaussian

$$p\left(\begin{bmatrix}\Delta\mathbf{x}_t\\ \Delta\boldsymbol{\mu}_t\end{bmatrix} \mid \mathcal{A}\right) \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, J_t = \begin{bmatrix}S_t & \mathrm{Cov}(\mathbf{x}_t, \boldsymbol{\mu}_t \mid \mathcal{A})\\ \mathrm{Cov}(\mathbf{x}_t, \boldsymbol{\mu}_t \mid \mathcal{A})^T & M_t\end{bmatrix}\right). \quad (5.57)$$

They do not decouple the calculation of S_t and M_t as we do through Theorem 5.1, but they derive the following recursive update scheme for the covariance J_t of the joint expected distribution:

$$J_0 = \begin{bmatrix}\Sigma_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \quad (5.58)$$

$$J_t = F_t J_{t-1} F_t^T + G_t \begin{bmatrix}Q_t & \mathbf{0}\\ \mathbf{0} & R_t\end{bmatrix} G_t^T, \quad (5.59)$$

with

$$F_t = \begin{bmatrix}A_t & B_t L_{t-1}\\ K_t H_t A_t & A_t + B_t L_{t-1} - K_t H_t A_t\end{bmatrix}, \quad (5.60)$$

$$G_t = \begin{bmatrix}V_t & 0\\ K_t H_t V_t & K_t W_t\end{bmatrix}. \quad (5.61)$$

They then can extract S_t as the upper left block of J_t. As can be seen from the equations, they multiply matrices of dimension $(2\dim(\mathbf{x}_t) \times 2\dim(\mathbf{x}_t))$, while our approach multiplies only matrices of dimension $(\dim(\mathbf{x}_t) \times \dim(\mathbf{x}_t))$. Besides this, their approach needs exactly the same calculations as ours, because they also need to calculate Σ_{t-1} in order to compute K_t. So all other computations being equal, the final recursive calculation of S_t in our approach is 2^3 times faster than the one by van den Berg *et al.* (if the standard matrix multiplication algorithm for $n \times n$-matrices with runtime n^3 is applied). In the next section, we present a detailed comparison of the runtimes in practice.

5.3 Experimental Evaluation

We evaluated our approach in extensive experiments with a differential drive robot motion model and a sensor model for measurements consisting of the distance and the relative angle between the robot and a set of uniquely identifiable landmarks. In the experiments, we described the state $\mathbf{x}_t$ of the robot by its pose $[x_t, y_t, \theta_t]$ in the 2d-plane. We used a discretization of $10\,\mathrm{Hz}$ for the time steps, resulting in trajectories with approximately 20

time steps per meter. We measured all runtimes using a single-threaded implementation on an Intel® Core™ i7 2.8GHz with 12GB RAM.

In the experiments, we compared the runtimes of our approach, the approach by van den Berg *et al.* [11] (described in Section 5.2.2), and a Monte-Carlo simulation. To produce results that are independent of a specific scenario, we considered randomly generated trajectories of different lengths, ten for each length in $\{25\,\mathrm{m}, 50\,\mathrm{m}, \ldots, 300\,\mathrm{m}\}$, which makes 120 trajectories in all. Each trajectory connects a different set of randomly sampled goal points in a $15\,\mathrm{m} \times 15\,\mathrm{m}$ large environment. Furthermore, for each trajectory we individually sampled a map consisting of 20 landmarks. Figure 5.2 shows four of

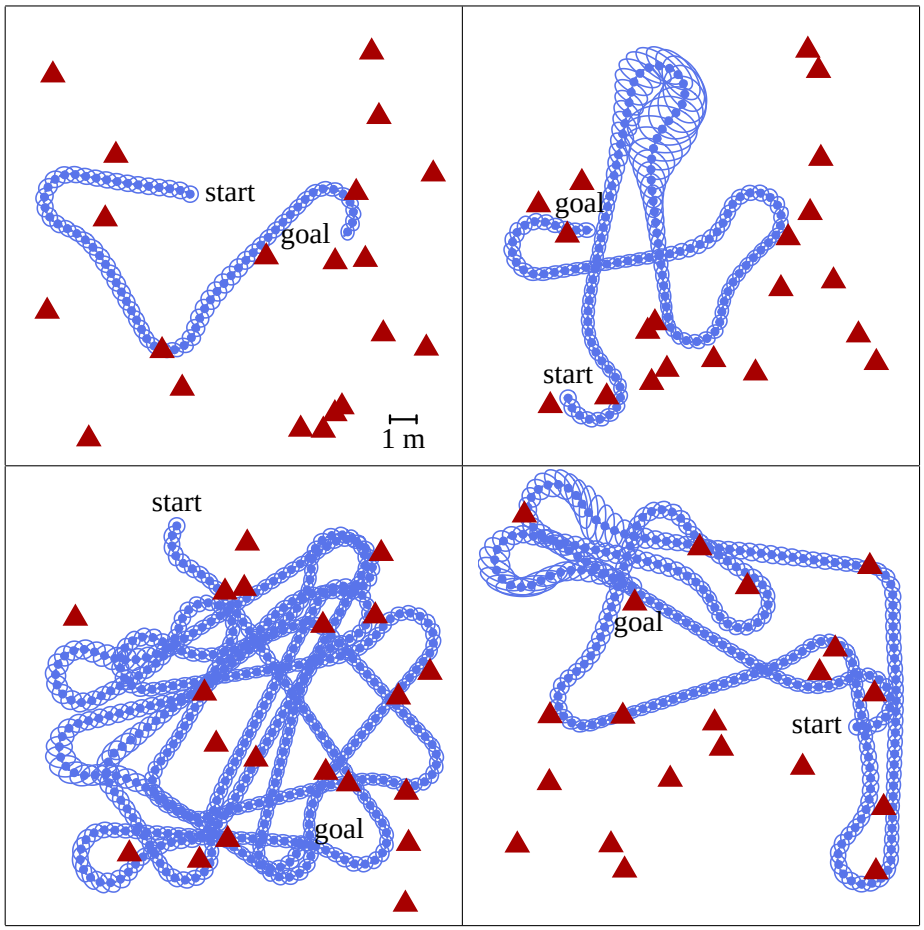

Figure 5.2: Randomly sampled trajectories and 99% confidence ellipses of the calculated expected distributions. The depicted trajectories have lengths of $25\,\mathrm{m}$, $50\,\mathrm{m}$, $100\,\mathrm{m}$, and $200\,\mathrm{m}$, respectively (clockwise from upper left). The sampled landmark positions are shown as red triangles.

these trajectories together with the sampled landmark positions.

A detailed comparison of the runtimes of our approach and of the approach by van den Berg *et al.* can be seen in Figure 5.3. The figure shows the runtimes needed for the complete calculation of the covariances of the expected distributions, which includes the

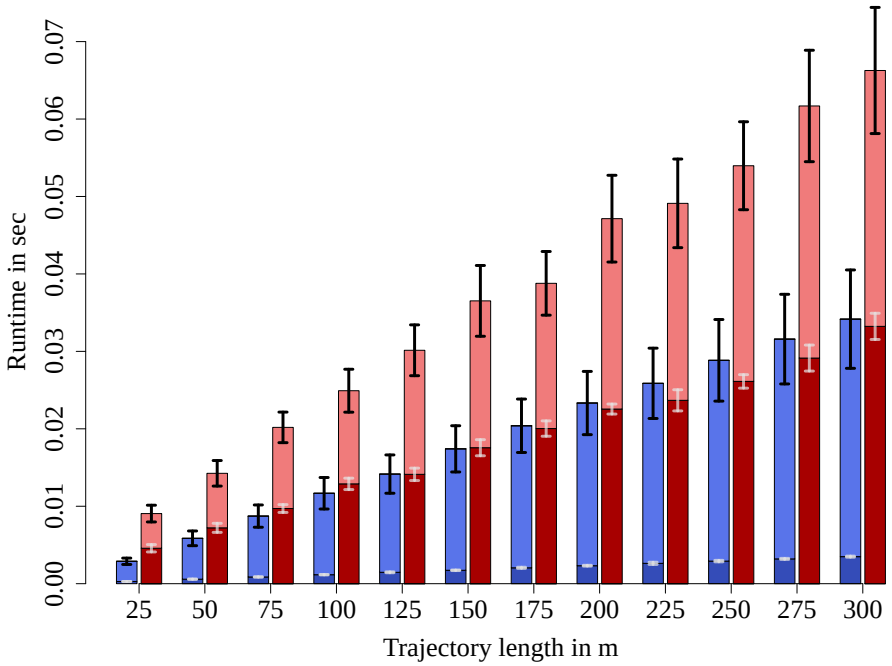

Figure 5.3: Comparison between the runtimes (stated in sec) of our approach (blue) and the approach by van den Berg *et al.* (red) on randomly sampled trajectories with lengths between $25\,\mathrm{m}$ and $300\,\mathrm{m}$. Shown are the means and 95% error bars. Indicated in dark blue and dark red are the fractions of the runtimes that were spent for the actual matrix multiplications that our approach accelerates.

calculation of Σ_t and of the Jacobians defined in (5.16). Additionally, in darker colors, it shows the fractions of the runtimes that were actually spent on the matrix multiplications that our approach speeds up, i.e., the calculation of S_t via M_t as in Algorithm 6 and via J_t as in (5.59), respectively. As can be seen from the figure, our approach significantly speeds up the matrix multiplication part, and thereby reduces the overall runtime approximately by half. The values of the calculated covariances resulting from the two approaches differed by at most $3.64 \cdot 10^{-12}$, which is within the range of machine precision.

Both our approach and the one by van den Berg *et al.* are orders of magnitude faster than Monte-Carlo simulation. For example, estimating the expected distributions for the sampled trajectories with length $100\,\mathrm{m}$ took $36.5\,\mathrm{sec}$ on average with Monte Carlo simulation. For the same trajectories, our approach spent $0.0117\,\mathrm{sec}$ and van den Berg's approach spent $0.0249\,\mathrm{sec}$ on average. For the trajectories with other lengths, the results were similar. The runtime of the Monte-Carlo simulation depends strongly on the number of simulated episodes used. In our comparison, we used 1000 episodes, while in practical applications, typically more episodes are needed to achieve the desired accuracy, e.g., in Chapter 4 we use $10,000$ episodes for simulating the states of a mobile robot.

In the following chapters, we utilize our approach to estimate expected distributions

when placing landmarks for mobile robot navigation. The experimental results presented in these chapters demonstrate the efficiency of our estimation scheme for expected distributions when applied in the context of landmark placement. Table 8.1, for example, shows a comparison of the overall runtimes of two of our landmark placement approaches when using our estimation scheme and when using van den Berg's method inside the landmark placement algorithms. The next chapters also contain experiments with real robots demonstrating that, if calibrated correctly, our estimated expected distributions consistently capture the deviations of real mobile robots in practice.

5.4 Discussion

In this chapter, we presented a novel recursive calculation scheme for estimating the expected probability distributions of the states of a mobile robot even before it starts operation. Our approach decouples the calculation of the covariances of the states of the robot and of its localization estimates, which reduces the runtime of the method. In extensive experiments, we showed that our approach significantly reduces the computation time, compared to state-of-the-art approaches. The method introduced in this chapter is the main building block of the objective functions that we optimize in the landmark placement approaches presented in the following chapters.

Chapter 6

Landmark Placement for Navigation

In this chapter, we consider the problem of placing a minimum set of landmarks in the environment of a mobile robot in order to optimize its navigation performance. Concretely, the landmark sets placed by our method guarantee that the maximum deviation of the robot from its desired trajectory stays below a user-defined bound with high confidence. In contrast to the landmark placement method for localization presented in Chapter 4, the approach presented in this chapter assumes that the robot is controlled by an LQR controller, which allows us to use the efficient method from the previous chapter to compute the deviation guarantee. We evaluate our approach in extensive experiments carried out both in simulation and with real robots. The experiments demonstrate that our method is applicable even to large-scale indoor scenarios with trajectory lengths of more than $200\,\mathrm{m}$, that it outperforms baseline approaches, and that it is suitable for long-term operation of mobile robots.

In this chapter, we present a novel algorithm for landmark placement, which aims at finding a minimum landmark configuration such that the deviation of the robot from its desired trajectory stays below a user-defined threshold d_g with high confidence. Our method works in two stages. In the first stage, it places landmarks in an incremental fashion. Thereby, it assumes that the robot is controlled by an LQR controller and uses the linearization and the calculation scheme from the previous chapter to efficiently evaluate the deviation guarantee. This stage aims at placing the smallest number of landmarks for which the deviation guarantee holds. In the second stage, our method employs a Monte Carlo simulation for the computed landmark configuration to validate that the guarantee also holds for the possibly non-linear models.

In contrast to the landmark placement approach for localization presented in Chapter 4, the method from this chapter uses Monte Carlo simulation only for validation and not in

the landmark placement algorithm itself. The resulting efficiency of our method allows us to deal even with large instances of the landmark placement problem (i.e., long trajectories). Another difference between the landmark placement approach for localization and the approach from this chapter is that here, we do not apply techniques from submodular function optimization, as the deviation guarantee exhibits a non-submodular behavior.

Our approach has several characteristics that make it especially useful for mobile robot navigation. It can deal with arbitrary trajectories, and the maximum allowed deviation of the robot can be defined individually for every part of the trajectories. Furthermore, it takes into account the properties of the individual robotic system, which results in customized landmark sets: while high-precision robots need only a few landmarks for reaching the deviation guarantee, low-cost systems typically require more landmarks. Note that our incremental method simultaneously determines the number and positions of the landmarks needed to meet the desired guarantee.

This chapter is organized as follows. After formalizing the deviation guarantee that our approach aims to satisfy in the next section, we describe the prediction of the deviation from the trajectory in linearized systems in Section 6.2. Afterwards, in Section 6.3, we present our incremental landmark placement algorithm. In Section 6.4, we give a theoretical evaluation of our method. Finally, in Section 6.5, we provide extensive experiments in which we evaluate the algorithm in simulations and in real-world applications.

6.1 Deviation Guarantee

When placing landmarks, we consider a mobile robot that can be described by the navigation system previously introduced in Section 5.1. To briefly recapitulate, we assume the time to be discretized into steps of equal duration. At each time step t, the robot changes its state and makes observations according to stochastic motion- and sensor models that are subject to random errors. We define a navigation task as a trajectory that the robot should follow. A trajectory $\mathcal{T} = (\mathbf{x}^\star_{0:T}, \mathbf{u}^\star_{1:T})$ can be considered as a series of states and desired controls the robot should execute to reach these states. We assume that the trajectory will be executed using a linear-quadratic regulator (LQR) [13] feedback controller, as described in Section 5.2.

The localization uncertainty and, as a result, also the deviation from the desired trajectory strongly depend on the specific configuration of landmarks $\mathcal{A} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n\}$ that are observed during operation. Our approach selects landmark positions $\boldsymbol{\ell}_i \in \mathcal{L}$ from a continuous space of possible landmark locations. We evaluate the quality of a landmark configuration based on the deviation of the (real) state $\mathbf{x}_t$ from the desired state $\mathbf{x}^\star_t$ at each time step t (ignoring the control part $\mathbf{u}^\star_{0:T}$ of the trajectory). In particular, we consider the Euclidean distance between the part of the state $\mathbf{x}^p_t$ describing the position of the robot

and $\mathbf{x}_t^{\star p}$. We focus on limiting the deviation

$$\|\mathbf{x}_t^p - \mathbf{x}_t^{\star p}\|_2 \tag{6.1}$$

of the robot from its trajectory at all time steps $t \in [0, T]$. Note that limiting the deviation of the position of the robot implicitly limits the deviation of the other relevant parts of the state, too. A large error in rotation, for example, would result in increasing deviations of the positions in consecutive time steps, and is therefore restricted. Our approach aims at finding the landmark configuration $\mathcal{A}$ with the fewest elements for which the *deviation guarantee*

$$\forall t \in [0, T] : \; p\left(\|\mathbf{x}_t^p - \mathbf{x}_t^{\star p}\|_2 \leq d_g(\mathbf{x}_t^\star) \mid \mathcal{A}\right) \geq p_g \tag{6.2}$$

holds. This guarantee ensures that the probability of deviating at most d_g from the desired trajectory is at least p_g. Note that d_g can be either a globally constant value or depend on the position or time.

6.2 Predicting the Deviation from the Trajectory

To validate the guarantee (6.2) for a certain landmark configuration $\mathcal{A}$, we need to compute $p\left(\|\mathbf{x}_t^p - \mathbf{x}_t^{\star p}\|_2 \leq d_g(\mathbf{x}_t^\star) \mid \mathcal{A}\right)$. For this, we consider the expected probability distribution

$$p(\mathbf{x}_t - \mathbf{x}_t^\star \mid \mathcal{A}) = p(\Delta\mathbf{x}_t \mid \mathcal{A}) \tag{6.3}$$

that we discussed in detail in Section 5.1.

For general nonlinear systems, expected distributions can be estimated via Monte Carlo simulation by sampling observations and controls and averaging over numerous runs. However, this is computationally expensive, especially for large instances of the landmark placement problem (i.e., long trajectories). Therefore, in the main part of our landmark placement algorithm, we locally linearize the system around the desired trajectory and apply the efficient recursive calculation scheme for the expected distributions that we introduced in Section 5.2.

6.2.1 Evaluation of the Deviation Guarantee

In the linearized system, we can efficiently check whether the deviation guarantee (6.2) holds. Let S_t^p be the part of the covariance S_t of the expected distribution $p(\Delta\mathbf{x}_t \mid \mathcal{A})$ corresponding to the position of the robot. The length $a_t(\mathcal{A})$ of the major semi-axis of the p_g-confidence ellipsoid of S_t^p can be calculated using

$$a_t(\mathcal{A}) = c\sqrt{\lambda_{t,1}}\,, \tag{6.4}$$

where $\lambda_{t,1}$ is the largest eigenvalue of S_t^p and c is a scaling factor corresponding to p_g via the regularized Gamma function as described in [11]. If $a_t(\mathcal{A}) \leq d_g$, then the p_g-ellipsoid

of S_t^p is inside a circle with radius d_g and guarantee (6.2) holds for the linearized system. Figure 6.1 visualizes this fact in a two-dimensional example.

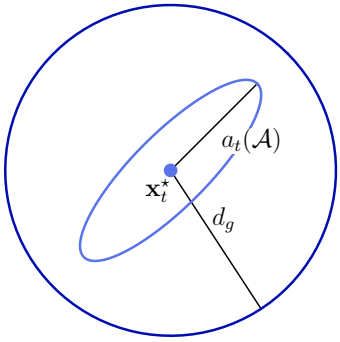

Figure 6.1: Example of a desired robot state $\mathbf{x}_t^\star$ and a corresponding p_g-confidence ellipse (light blue) together with a circle of radius d_g. If $a_t(\mathcal{A}) \leq d_g$, then the probability that the deviation of the robot is less than d_g is larger than p_g.

Note that this test is a conservative approximation and is exact if the p_g-ellipsoid is a sphere.

6.2.2 Observability of Landmarks

In the linearized system described in Section 5.2, we linearize the sensor model of the robot at time t around the desired robot state $\mathbf{x}_t^\star$. For sensors with a limited field of view, this means that a landmark is considered as observable by the robot if it is observable from within the desired state $\mathbf{x}_t^\star$ of the robot. However, the landmark might not be observable from within other states at which the robot is with a high likelihood at time t. Therefore, when considering robots with a limited sensor range or occlusions due to objects inside the field of view, the probability that the deviation guarantee holds might be overestimated with this approximation. To avoid this, we approximate landmark observability in a conservative way when placing landmarks. This conservative approximation is directly linked to the deviation guarantee (6.2), and the general idea behind it was first introduced by Vitus and Tomlin [105]:

We consider a landmark as observable at time t only if it is p_g-*observable*, i.e., it is observable from every pose inside the p_g-ellipsoid of S_t around $\mathbf{x}_t^\star$. If a landmark configuration satisfies the deviation guarantee (6.2) when only p_g-observable landmarks are considered, it also satisfies it when using all observable landmarks.

If the environment of the robot is a planar free space and the robot has a circular field of view, we can test analytically if a landmark is p_g-observable. For other types of scenarios, we apply an approximative test utilizing a sigma point method.

Observability in Free Space

For the two-dimensional case (i.e., $\mathbf{x}_t^p = [x,\, y]^T$) without occlusions or other restrictions, and for a robot with a circular field of view with radius r, there exists a closed-form solution to testing if a given landmark $\boldsymbol{\ell}$ is p_g-observable. Because of the circular field of view, the orientation of the robot does not matter for testing the observability of landmarks. Therefore, for testing if $\boldsymbol{\ell}$ is p_g-observable, it suffices to consider the p_g-ellipse of S_t^p instead of the p_g-ellipsoid of S_t. Consider the p_g-ellipse of S_t^p centered at the origin of the coordinate system. We apply a principal axis transformation on the ellipse so that afterwards its semi-axes lie on the axes of the coordinate system. Applying this transformation on S_t^p yields a diagonal matrix $S_t^{p\prime} = \operatorname{diag}(\lambda_{t,1}, \lambda_{t,2})$ with the diagonal elements identical to the eigenvalues of the matrix. Any point $\mathbf{x}' = [x_1',\, x_2']^T$ on the transformed ellipse $\mathcal{E}$ can then be described as $\mathbf{x}' = c\, \operatorname{diag}(\sqrt{\lambda_{t,1}}, \sqrt{\lambda_{t,2}})\, \mathbf{x}$ for a point $\mathbf{x} = [x_1,\, x_2]^T$ on the unit circle $\mathcal{C}$ and the scaling factor c from Equation (6.4). To test if landmark $\boldsymbol{\ell}$ is p_g-observable, we apply the same principal axis transformation also on the relative position $(\boldsymbol{\ell} - \mathbf{x}_t^\star)$ of the landmark, resulting in $\boldsymbol{\ell}' = [\ell_1',\, \ell_2'] = c\, \operatorname{diag}(\sqrt{\lambda_{t,1}}, \sqrt{\lambda_{t,2}})\, (\boldsymbol{\ell} - \mathbf{x}_t^\star)$. Checking if $\boldsymbol{\ell}$ is p_g-observable means testing if

$$\max_{\mathbf{x}' \in \mathcal{E}} \|\mathbf{x}' - \boldsymbol{\ell}'\|_2 \leq r \tag{6.5}$$

$$\Leftrightarrow \max_{\mathbf{x} \in \mathcal{C}} \|c\, \operatorname{diag}\left(\sqrt{\lambda_{t,1}}, \sqrt{\lambda_{t,2}}\right)\, \mathbf{x} - \boldsymbol{\ell}'\|_2 \leq r \tag{6.6}$$

$$\Leftrightarrow \max_{x_1 \in [-1,1],\, \operatorname{sgn} \in \{-1,1\}} \Bigg(\left(c\sqrt{\lambda_{t,1}} x_1 - \ell_1'\right)^2 + \left(c\sqrt{\lambda_{t,2}}\sqrt{1 - x_1^2}\, \operatorname{sgn} - \ell_2'\right)^2 - r^2 \Bigg) \leq 0\,. \tag{6.7}$$

We can test analytically if this inequality holds: Applying a distinction of cases for sgn, we set the derivative with respect to x_1 of the function inside the $\max$-operator to zero. If the values of the function inside the $\max$-operator at the nulls of its derivative are all less or equal than zero, then $\boldsymbol{\ell}$ is p_g-observable. Reordering and squaring the derivative removes the remaining square-root in the formula and leads to a fourth order polynomial. We calculate the nulls of this polynomial analytically in an efficient way. For an overview of the analytical methods for solving fourth order polynomial equations, see Shmakov [95].

Observability in Structured Environments

For general sensor models and environments in which structures like walls restrict the field of view of the sensor, the analytical solution to finding the p_g-observable landmarks is not applicable. In these cases, we approximate the solution by selecting a set $\mathcal{S}$ of $(2\dim(\mathbf{x}_t^\star) + 1)$ poses on the p_g-ellipsoid of S_t similar to the set of sigma points used

in the unscented Kalman filter [45]. The set $\mathcal{S}$ consists of the center of the p_g-ellipsoid and the $2\dim(\mathbf{x}_t^\star)$ vertices of the ellipsoid. Knowing the setup of the environment, we can test for a given landmark $\boldsymbol{\ell}$ and a given pose $\mathbf{s} \in \mathcal{S}$ if the sensor of the robot would be able to observe $\boldsymbol{\ell}$ when positioned at $\mathbf{s}$. If this test evaluates to true for all $\mathbf{s} \in \mathcal{S}$, we consider $\boldsymbol{\ell}$ as p_g-observable.

6.3 Incremental Landmark Placement Algorithm

With the techniques described above, we can test if the deviation guarantee holds for a specific landmark configuration. Our landmark placement approach uses these techniques and aims at minimizing the number of landmarks that have to be placed for the deviation guarantee to hold. Since the dimensionality of the search space grows with the length of the trajectory, globally searching for the optimal landmark configuration is in general computationally intractable. However, using an incremental placement algorithm, we can efficiently find an approximate solution to the landmark placement problem.

In contrast to our other landmark placement approaches, this incremental placement algorithm does not apply the submodular function optimization techniques presented in the background chapter. The reason for this is that the evaluation of the deviation guarantee relies on highly non-submodular components. An example is the conservative approximation of the observability of landmarks described above. At time t the p_g-observability of a certain landmark $\boldsymbol{\ell}$ depends on the size of the confidence ellipsoid at that time step, which itself depends on the landmarks that the robot could observe at earlier time steps. So adding a landmark $\boldsymbol{\ell}$ to a larger superset of a small landmark set increases its chances of being observable and therefore might also increase the impact that this landmark has on the deviation guarantee. This behavior is in contradiction to the definition of submodularity (see Section 3.4.1).

6.3.1 Landmark Placement for the Linearized System

In a first stage, our algorithm employs the linearized system and the conservative approximation of the landmark observability to incrementally place landmarks. Considering linearized Gaussian models is beneficial because the expected distributions can be efficiently calculated recursively as described in Section 5.2. The objective of our approach is to minimize the number of landmarks needed for the deviation guarantee to hold over the whole trajectory $(\mathbf{x}_{0:T}^\star, \mathbf{u}_{1:T}^\star)$. We approximate this minimum by placing each landmark so that it maximizes the number of time steps for which the deviation guarantee (6.2) holds. Let

$$t_g(\mathcal{A}) = \max\{t \mid a_s(\mathcal{A}) \leq d_g \; \forall s \leq t\} \tag{6.8}$$

Algorithm 7 Incremental landmark placement for bounding the deviation guarantee

Input: $\mathcal{T}$, $\mathcal{L}$
Output: $\mathcal{A}$

```
A ← ∅
τ ← 0
while τ < T do
    ℓ* ← argmax_{ℓ∈L} t_g(A ∪ {ℓ})
    τ* ← t_g(A ∪ {ℓ*})
    if τ* = τ then
        ℓ* ← argmin_{ℓ∈L} a_{t_g}(A ∪ {ℓ})
    end if
    A ← A ∪ {ℓ*}
    τ ← τ*
end while
return A
```

be the maximum time step for which the landmark set $\mathcal{A}$ guarantees that Equation (6.2) holds in the linearized system for the first part of the trajectory $(\mathbf{x}^\star_{0:t_g}, \mathbf{u}^\star_{1:t_g})$. The value of $t_g(\mathcal{A})$ obviously depends on $\mathbf{x}^\star_{0:T}$ and $\mathbf{u}^\star_{1:T}$, but for readability, we drop this dependency in the formula. In every iteration, our algorithm adds the landmark $\boldsymbol{\ell}^\star$ that maximizes $t_g(\mathcal{A} \cup \{\boldsymbol{\ell}^\star\})$ to the already placed set of landmarks $\mathcal{A}$. In some cases, one additional landmark is not enough to increase t_g. This can happen for example if $d_g(\mathbf{x}^\star_{t_g+1})$ is chosen considerably smaller than $d_g(\mathbf{x}^\star_{t_g})$. In these cases, the algorithm places the landmark that minimizes $a_{t_g}(\mathcal{A})$ instead. Reducing $a_{t_g}(\mathcal{A})$ increases the likelihood that in the next step a landmark can be found that increases t_g again (see Equation (6.8)). Algorithm 7 describes the incremental landmark placement for the linearized system. As input, it takes the considered navigation task $\mathcal{T}$ and the space of all possible landmark locations $\mathcal{L}$. Note that for different structures of the space $\mathcal{L}$, there exist different efficient implementations for the argmax and the argmin operators in the algorithm. See Section 6.5.1 for the details of our implementation.

6.3.2 Monte Carlo Validation

In a second stage, we check the computed landmark configuration $\mathcal{A}$ for the deviation guarantee via Monte Carlo simulation using the real (possibly nonlinear, non-Gaussian) models. We do this to account for approximation errors due to the linearization, the Gaussian assumption, and the conservative approximation of landmark observability. The Monte Carlo simulation samples robot states $\mathbf{x}_{0:T}$, controls $\mathbf{u}_{1:T}$, and observations $\mathbf{z}_{1:T}$ of the landmarks in $\mathcal{A}$. It counts the number of time steps t in which $\|\mathbf{x}^p_t - \mathbf{x}^{\star p}_t\|_2 \leq d_g(\mathbf{x}^\star_t)$,

as required in guarantee (6.2). Averaging over numerous runs yields an estimate p_{MC} of p_g for which the deviation guarantee in the real system holds. If $p_{\text{MC}} < p_g$, one can use arbitrary heuristics to place additional landmarks. For example, one could run our algorithm for increased values of p_g or decreased values of d_g.

6.3.3 Continuous Operation on Round Trips

For round-trip tasks, for which $\mathbf{x}_0^\star = \mathbf{x}_T^\star$, Algorithm 7 can be used for finding a landmark set that guarantees the error bound for multiple successive executions of the task. This can be achieved by adjusting the part S_0^p of the covariance of the initial expected distribution corresponding to the position of the robot. If we design S_0^p so that the p_g-ellipsoid of S_T^p at the final time step is inside the p_g-ellipsoid of S_0^p, then, when reaching the goal, the robot is inside the p_g-ellipsoid of S_0^p with probability greater than or equal to p_g. Therefore when starting a next run of the same navigation task, the deviation guarantee is still satisfied.

6.4 Relation between Deviation Guarantee and Localization Uncertainty

In mobile robotics, often the trace of the posterior covariance $tr(\Sigma_t)$ is used as a measure for the localization uncertainty [65, 105]. In this section, we show that by enforcing the deviation guarantee, our approach also guarantees a bound on $tr(\Sigma_t)$.

Lemma 6.1. *If the state $\mathbf{x}_t$ consists only of the position of the robot, i.e., $S_t = S_t^p$, then observing the landmarks placed by Algorithm 7 guarantees that*

$$tr(\Sigma_t) \leq \dim(\mathbf{x}_t) \left(\frac{d_g}{c}\right)^2 . \tag{6.9}$$

Proof. From Section 6.2.1 we know that $a_t(\mathcal{A}) \leq d_g$. With the definition of a_t in Equation (6.4), it follows that

$$c\sqrt{\lambda_{t,1}} \leq d_g \tag{6.10}$$

$$\Leftrightarrow \quad c^2 \lambda_{t,1} \leq d_g^2 , \tag{6.11}$$

where $\lambda_{t,1}$ is the largest of the $\dim(\mathbf{x}_t)$ many eigenvalues of S_t (because $S_t = S_t^p$). Therefore, it also holds that

$$\forall i \in \{1, \ldots, \dim(\mathbf{x}_t)\} : \qquad c^2 \lambda_{t,i} \leq d_g^2 \tag{6.12}$$

$$\Rightarrow c^2 \sum_{i=1}^{\dim(\mathbf{x}_t)} \lambda_{t,i} \leq \dim(\mathbf{x}_t)\, d_g^2 . \tag{6.13}$$

Since the trace of a matrix is equivalent to the sum of its eigenvalues [36], it follows that

$$c^2\, tr(S_t) \leq \dim(\mathbf{x}_t) d_g^2\,. \tag{6.14}$$

From Corollary 5.2 in the previous chapter, we know that $tr(S_t) \geq tr(\Sigma_t)$. Plugged into Equation (6.14), this finishes the proof. □

6.5 Experimental Evaluation

We evaluated our landmark placement algorithm and compared it to baseline landmark placement approaches in extensive experiments both in simulation and with real robots.

6.5.1 Experimental Setup

In our experiments we considered wheeled robots navigating on a plane. Since the most common drive types used in industry are (non-holonomic) differential drive robots and holonomic robots equipped with Mecanum wheels, we carried out our experiments with robots of these types. For self-localization we considered three different types of sensors detecting uniquely identifiable landmarks: a *range-only* sensor, measuring only the distances to the landmarks, a *bearing-only* sensor, measuring only the relative angles between the robot and the landmarks, and a *range-and-bearing* sensor, measuring both. Hence, in the following experiments all motion models and all sensor models have nonlinear components.

We evaluated our landmark placement approach on two different kinds of spaces of possible landmark locations, depending on the environment the robot operates in. In the free-space setting without any obstacles in the environment of the robot, we considered the complete two-dimensional plane the robot was navigating on as space $\mathcal{L}_{\text{free}}$ of possible landmark locations. We also evaluated our approach in structured environments, which were defined by walls. Here, we allowed landmark placement only on the surfaces of predefined walls resulting in a one-dimensional space $\mathcal{L}_{\text{walls}}$ of possible landmark locations.

We implemented the argmax and argmin operators used in Algorithm 7 in a two-stage procedure that is robust to local optima. In the first step, we discretized the relevant part of $\mathcal{L}$ that is observable from the trajectory and evaluated each of these points. Around the optimum on the discrete point set, we then performed a fine search with Powell's method [83] in the second step.

6.5.2 Placement in Free Space

In the first set of experiments, we considered environments without obstacles and allowed landmarks to be placed in the complete two-dimensional plane $\mathcal{L}_{\text{free}}$. For these

experiments, we used the analytical method for testing landmark observability in the placement algorithm as described in Section 6.2.2. For all types of sensors, we assumed a circular field of view around the robot with radius $2\,\mathrm{m}$. We evaluated our algorithm on five navigation tasks T1-T5 (see Figures 6.2 and 6.3) for a differential drive robot. We simulated every task for all three sensor models, resulting in 15 experiments. Figure 6.2 shows the landmarks our algorithm computed for the three sensor models in the first task T1, together with expected and posterior distributions.

Figure 6.3 depicts the landmark configurations and expected distributions for the other four tasks T2-T5 for a range-only sensor. For all trajectories, we set $p_g = 99\%$ and $d_g = 0.5\,\mathrm{m}$. For the pick-and-place task T5, we changed d_g to $0.2\,\mathrm{m}$ in the pick-up and the deposit zones (gray rectangles). Because of the high accuracy necessary to fulfill this task, we simulated a more precise robotic system than for the other tasks, i.e., we scaled down the noise values of the motion model and the sensor model of the robot.

Influence of the Sensor Model

As can be seen in Figure 6.2, the number of landmarks our algorithm computes and their configuration strongly depend on the chosen sensor model. For the range-only sensor, the landmarks tend to be further away from the trajectory than for the other two sensor models. The numbers of landmarks needed are stated in the first row of Table 6.1. Furthermore, the results of the Monte Carlo simulations in our algorithm varied strongly for the different sensor models. In every Monte Carlo simulation, we performed 1,000 simulated runs and calculated the proportion of time steps in which the deviation of the robot from its trajectory exceeded d_g. This proportion yields an estimate p_{MC} of p_g for the nonlinear models. For all trajectories and all sensor models, the values of p_{MC} for the landmark sets our approach computed are stated in the fifth row of Table 6.1.

For the range-only sensor, p_{MC} is considerably above the intended value of 99% in all tasks. For the range-and-bearing sensor, p_{MC} is slightly below 99% in the pick-and-place task and for the bearing-only sensor, p_{MC} is below 99% in three of the five tasks. These results indicate that the nonlinear components of the range measurements are less critical for landmark placement than those of the bearing measurements.

Comparison to other Landmark Placement Strategies

For comparison, we evaluated three baseline methods for placing landmarks in a way that the deviation guarantee is satisfied. Each method starts with a minimum number of landmarks and successively increases the number (or density) of landmarks until it finds a set for which the guarantee in the linearized system holds.

- *On trajectory* places landmarks at positions that are distributed equidistantly on the desired trajectory.

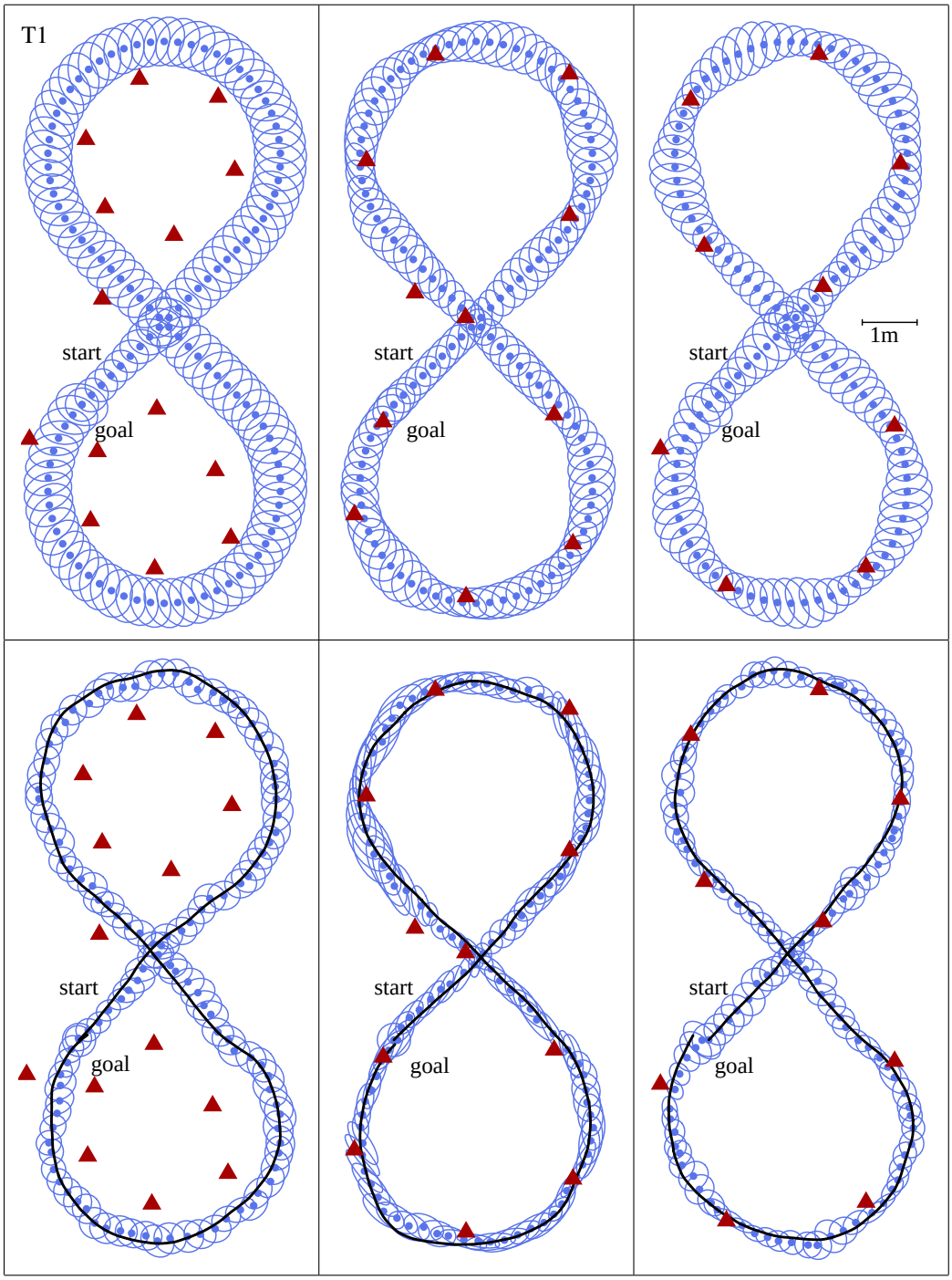

Figure 6.2: The landmark configurations (red triangles) our algorithm computed for the figure-eight trajectory T1 for three different sensor models: range-only (left), bearing-only (middle) and range-and-bearing (right). The blue points and ellipses in the upper row correspond to the means and the 99% confidence ellipses of the expected distributions, and in the lower row to the posterior distributions of simulated sample runs. The true positions of the robot in the sample runs are depicted as black lines.

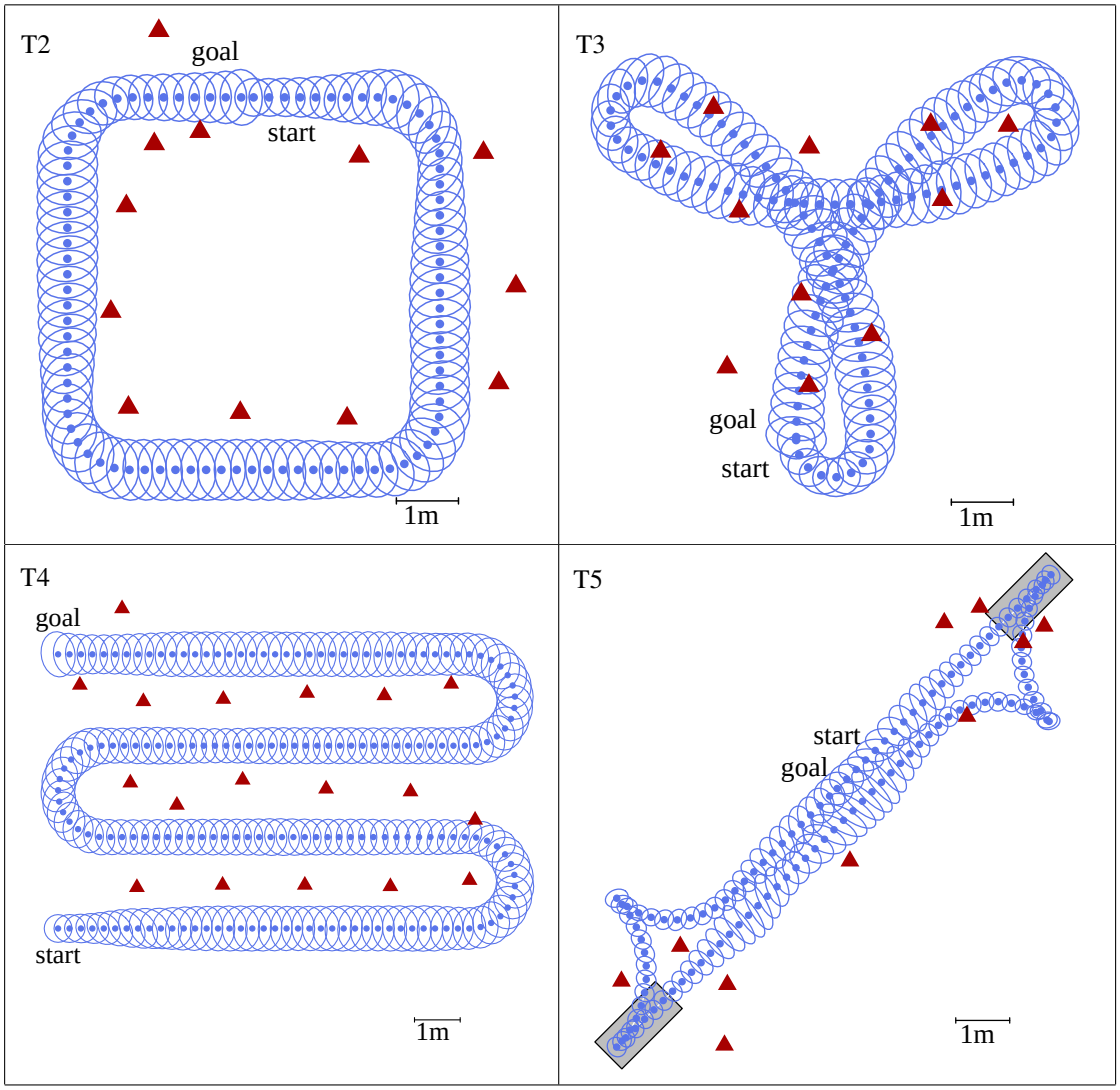

Figure 6.3: The landmark configurations (red triangles) our algorithm computed for four sample trajectories T2-T5 using a range-only sensor. T2 is a square, T3 a curved shape, T4 a sweeping trajectory, and T5 a pick-and-place task. The blue points and ellipses correspond to the means and the 99% confidence ellipses of the expected distributions. In T5, the pick-up zone and the deposit zone are marked as gray rectangles.

	Range-only sensor					Bearing-only sensor					Range-and-bearing sensor				
	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5
Number of landmarks															
Our approach	14	12	11	18	10	11	9	7	16	7	9	8	6	13	5
On trajectory	–	–	25	–	58	41	–	12	–	23	12	9	10	17	7
On grid	48	32	38	56	23	26	19	17	30	18	20	20	17	25	16
Random	108	63	62	138	62	75	66	51	88	31	38	29	38	37	15
p_{MC}															
Our approach	0.999	0.998	0.999	0.999	0.999	0.979	0.978	0.991	0.994	0.826	0.999	0.997	0.998	0.994	0.986
On trajectory	–	–	0.996	–	0.962	0.353	–	0.955	–	0.773	0.996	0.999	0.983	0.999	0.980
On grid	0.999	0.999	0.999	0.999	0.998	0.997	0.981	0.995	0.999	0.931	0.996	0.999	0.995	0.999	0.999
Random	0.999	0.999	0.999	0.999	0.998	0.996	0.996	0.999	0.999	0.999	0.999	0.999	0.996	0.999	0.999

Table 6.1: Numbers of placed landmarks and results of the Monte Carlo simulations for several types of landmark placement approaches and landmark detection sensors

- *On grid* places a landmark in the center of each cell of a regular grid. Starting with one cell covering the whole environment, the cell size is decreased at every iteration until the deviation guarantee holds. For efficiency, landmark positions that are outside the field of view of all states $\mathbf{x}^{\star}_{0:T}$ on the desired trajectory are not used.
- *Random* successively places landmarks at randomly chosen positions observable from the desired trajectory, until the deviation guarantee is satisfied.

The number of placed landmarks and the values of p_{MC} for all landmark placement strategies are stated in Table 6.1. Dashes in the table indicate that no valid landmark configuration could be found. For all experiments, our approach placed fewer landmarks than the other approaches. The *on trajectory* method is the best method after ours for the *range-and-bearing* sensor, measured by the number of placed landmarks. However, for the other two sensor models, the *on trajectory* method was not always able to find a landmark configuration that satisfied the guarantee in the linearized system. For this method, especially the nonlinearities in the *bearing-only* sensor model resulted in low values for p_{MC}.

For a comparison between our approach and the other approaches described in this thesis, see Chapter 8.

Experiments with a Miniature Robot

To further validate the simulation experiments for the scenarios in free space, we evaluated one of the landmark sets our algorithm generated also on the real e-puck robot [74] depicted in Figure 6.4. As a range-and-bearing sensor, we used a webcam pointing upwards detecting uniquely identifiable ARToolkit markers [15] attached to the ceiling at the positions that our algorithm computed. We considered the navigation task T1 scaled down to suit the miniature size of our robot (diameter $75\,\mathrm{mm}$) and the lower ceiling. Scaling the task by the factor 0.08 yields $d_g = 0.04\,\mathrm{m}$. To validate the deviation guarantee in practice, we measured the deviation $\|\mathbf{x}^p_t - \mathbf{x}^{\star p}_t\|_2$ of the position of the e-puck robot from its desired trajectory during operation. For this, we obtained the reference positions $\mathbf{x}_t$ from a MotionAnalysis motion capture system with four digital Raptor-E cameras. During 20 autonomous runs, $\|\mathbf{x}^p_t - \mathbf{x}^{\star p}_t\|_2$ was below d_g in 99.7% of the time steps.

6.5.3 Placement in Structured Environments

In the second set of experiments, we considered structured environments with walls and other obstacles at known positions. To represent the walls and obstacles, we used sets of lines, which are often referred to as line maps. In these experiments, we evaluated our algorithm for both spaces of possible landmark positions, the surfaces of the walls and

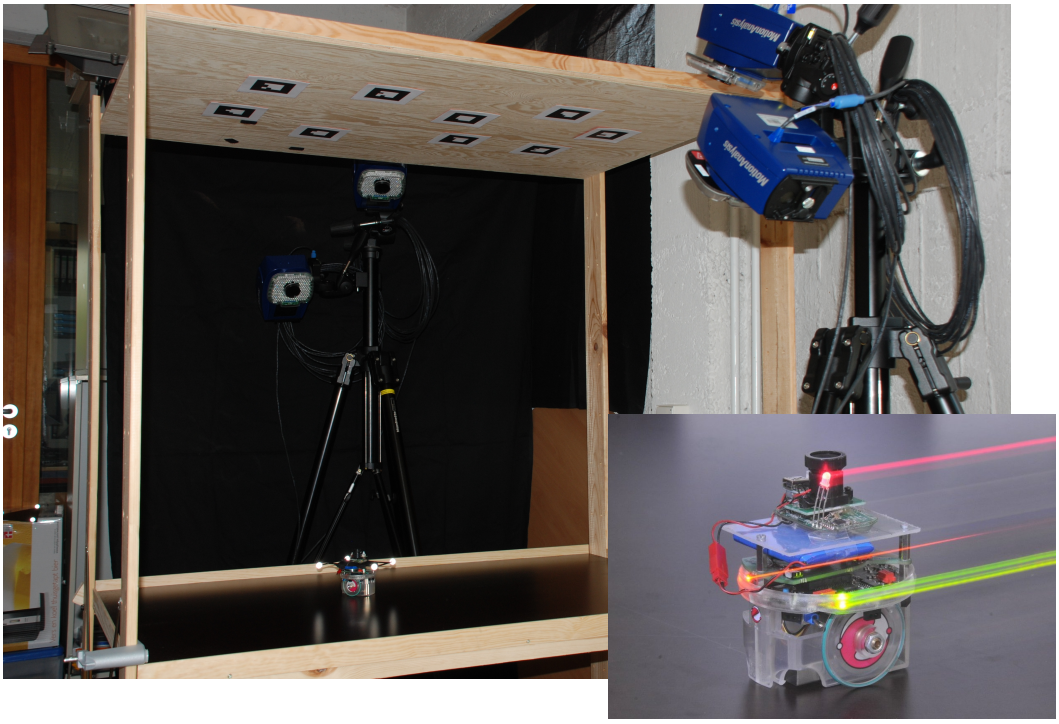

Figure 6.4: The miniature e-puck robot and its experimental environment. Mounted on top of the robot is a wireless webcam detecting the uniquely identifiable visual markers attached to the ceiling. The reference positions of the robot are obtained by the four-camera motion capture system.

obstacles $\mathcal{L}_{\text{walls}}$, and the plane $\mathcal{L}_{\text{free}}$. In the landmark placement algorithm, we used the sigma point approach described in Section 6.2.2 to test the landmark observability. This approach allows us to take into account occlusions from walls when testing the landmark observability, and it also allows us to consider non-circular fields of view of the robot. We simulated two navigation tasks in structured environments: T6 and T7 (see Figure 6.5).

The line maps in both tasks were manually extracted from the grid maps shown in light gray in the figures. The map in the task T6 corresponds to the Willow Garage building (grid map recorded by Brian Gerkey) and the map in T7 corresponds to building 079 on the Freiburg University campus. For these tasks we simulated a differential drive robot equipped with a range-and-bearing sensor having a maximum range of $5\,\text{m}$ and a half-circular field of view in front of the robot. Just as in the experiments in free space, we set $p_g = 99\%$ and $d_g = 0.5\,\text{m}$.

Comparison between Placement on Walls and Placement in Free Space

We compared the runtime and the number of placed landmarks of our algorithm for the landmark placement in $\mathcal{L}_{\text{walls}}$ to that of the placement in $\mathcal{L}_{\text{free}}$ on tasks T6 and T7. For the landmark placement in $\mathcal{L}_{\text{free}}$, we used an insight gained from empirical evaluations to

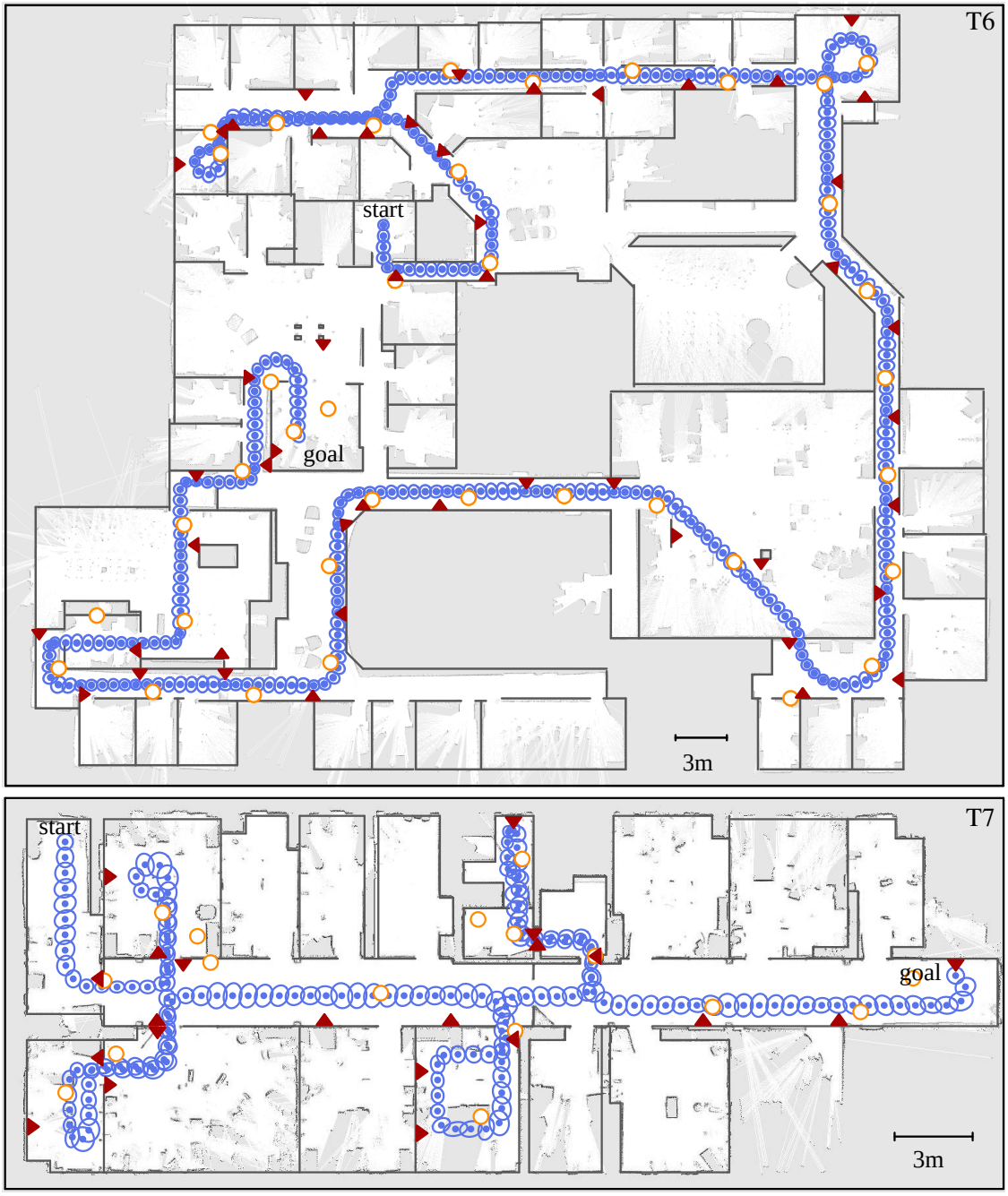

Figure 6.5: The landmark configurations computed by our algorithm when placing landmarks on the walls of buildings (red triangles) and in free space (orange circles). The blue points and ellipses correspond to the means and the 99% confidence ellipses of the expected distributions computed using the landmarks placed on the walls. The line maps of the buildings are depicted in dark gray, and the grid maps, from which the line maps were extracted, are shown in light gray.

		$\mathcal{L}_{\text{walls}}$	$\mathcal{L}_{\text{free}}$
T6	Length of Trajectory	218.5 m	
	Runtime	2 h 11 min	57 h 51 min
	Number of Landmarks	48	37
T7	Length of Trajectory	100.8 m	
	Runtime	9 min	6 h 59 min
	Number of Landmarks	21	16

Table 6.2: Comparison of landmark placement spaces

speed up the computation. In the experiments, our approach typically selected landmark positions $\boldsymbol{\ell}^\star$ inside the field of view of the current state of the robot $\mathbf{x}_\tau^\star$. Here, τ is the time step that is currently in the focus of the placement algorithm (see Algorithm 7). Therefore, we restricted the search space for the optimization to this area. For all evaluated tasks in our experiments, Algorithm 7 with the full search space did not place fewer landmarks than our implementation with the restricted search space. In structured environments, a similar restriction of $\mathcal{L}_{\text{walls}}$ was not applicable, as in our experiments the field of view of $\mathbf{x}_\tau^\star$ often did not contain the best landmark position $\boldsymbol{\ell}^\star$. In fact, at some time steps it did not contain any possible landmark positions at all. Hence, we restricted the search space in the optimizations to all walls that are observable from the trajectory up to time τ.

For both placement methods, in $\mathcal{L}_{\text{walls}}$ and in $\mathcal{L}_{\text{free}}$, we executed our algorithm single-threaded on an Intel® Core™ i7 2.8GHz with 12GB RAM. For T6 and T7, the lengths of the trajectories, the numbers of placed landmarks and the runtime of Algorithm 7 are listed in Table 6.2. As can be seen in the table, landmark placement on the plane $\mathcal{L}_{\text{free}}$ managed to fulfill guarantee (6.2) with fewer landmarks than landmark placement on the walls $\mathcal{L}_{\text{walls}}$. Hence, the landmark positions on the walls appear to be suboptimal. On the other hand, the runtime of the landmark placement in $\mathcal{L}_{\text{free}}$ is considerably longer than the runtime of the landmark placement in $\mathcal{L}_{\text{walls}}$. This is due to the fact that the search space $\mathcal{L}_{\text{free}}$ is considerably larger than $\mathcal{L}_{\text{walls}}$. Note that the space of possible landmark positions depends on the physical properties of the utilized sensor and the landmarks. For example, for a camera detecting landmarks attached to the ceiling of the environment, like in the experiments with the e-puck robot, typically $\mathcal{L}_{\text{free}}$ is considered for landmark placement. On the other hand, for a laser range finder that observes the walls of an environment, like in the application presented in the next section, $\mathcal{L}_{\text{walls}}$ is the appropriate space for landmark placement.

Long Term Experiments with a Holonomic Robot

We evaluated the continuous operation capability of our approach described in Section 6.3.3 in a long-term experiment with a real holonomic robot. The environment and

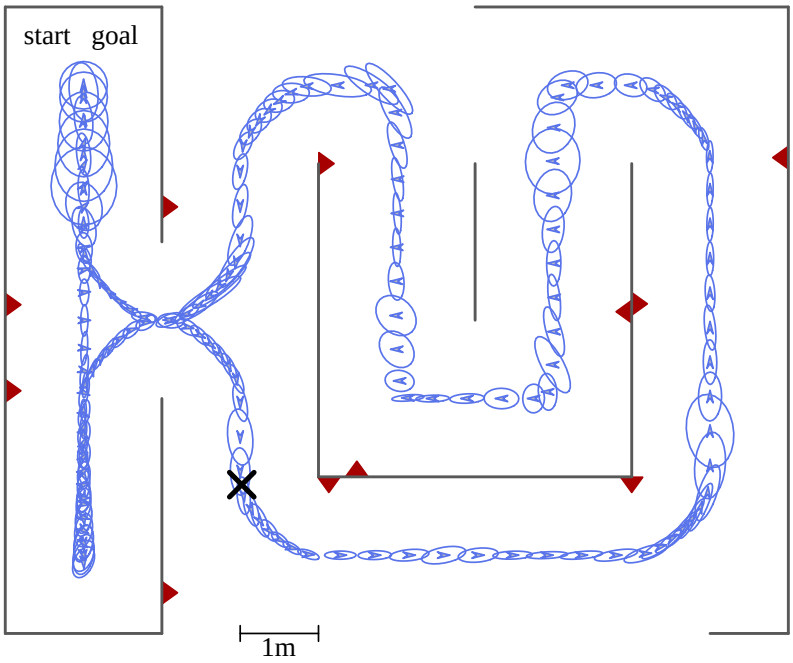

Figure 6.6: The landmark configuration (red triangles) our algorithm placed on the walls of the experimental environment of the holonomic robot. The blue arrows and ellipses correspond to the means and the 99% confidence ellipses of the expected distributions. The line map of the experimental environment is shown in dark gray. The black cross marks the spot on which the robot is located in Figure 6.7.

Figure 6.7: The holonomic KARIS robot operating in its experimental environment. It is equipped with two laser range finders mounted on opposite sides of the base for a 360° field of view. They detect reflective markers, which are attached to the walls, and are used as landmarks for localization. The reference positions of the robot are obtained by a motion capture system, part of which is shown in the back.

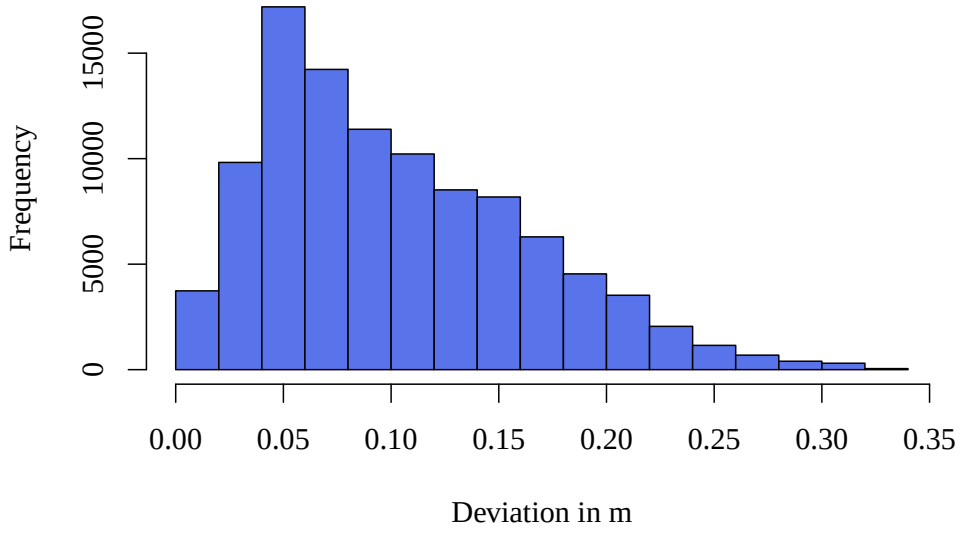

Figure 6.8: The histogram of the deviation of the KARIS robot from its desired trajectory during operation on a round trip for about three hours.

the round-trip trajectory of this navigation task are shown in Figure 6.6.

As robotic system, we used the KARIS robot [51], which is depicted in Figure 6.7. This robot is designed as a logistics robot for industrial application in storage facilities and production sites. It is equipped with Mecanum wheels, which allow for holonomic motion. Two SICK S300 laser range finders are mounted on opposite sides of the robot. Together, they return range measurements and intensity values in a $360°$ field of view with a resolution of $0.5°$. Based on the intensity values, the robot can detect landmarks made of reflective tape mounted on walls. Experiments showed that these reflective markers are reliably observable by the robot only if the angle between the landmark orientation and the robot position is at least $22°$. For landmark placement, we therefore used a sensor model with that observability constraint and with a circular field of view.

The number and positions of the landmarks placed by our algorithm highly depend on the covariance matrices Q_t and R_t of the noise in the motion model and the sensor model, respectively. Therefore, before placing the landmarks, we did a calibration run with the robot, in which the pose of the robot was determined precisely by a MotionAnalysis motion capture system with nine digital Raptor-E cameras. We estimated the noise matrices as the maximum likelihood values with respect to the reference positions and the odometry or landmark measurements, respectively.

Taking into account the calibrated noise values, Algorithm 7 placed 11 landmarks that can be seen in Figure 6.6. Using the observations of these landmarks for localization, the KARIS robot performed a continuous round trip on the defined trajectory for about three hours, corresponding to one battery charge. During this time, the robot completed the navigation task 62 times. The deviation of the robot from its desired trajectory

was monitored at a $10\,\mathrm{Hz}$ rate by the MotionAnalysis motion capture system. The captured deviation values of the robot never exceeded the specified bound $d_g = 0.5\,\mathrm{m}$ for all $102,300$ recorded time steps. Figure 6.8 shows the deviations from the trajectory in a histogram plot. As can be seen in the plot, the deviation of the robot from its desired trajectory was typically around $0.05\,\mathrm{m}$. The largest deviation value measured was $0.337\,\mathrm{m}$.

These experiments demonstrate that our approach is suitable for efficient placement of landmarks in unstructured or structured environments. The placed landmark configurations were proven to allow for a reliable navigation in extensive simulation and real-world experiments with different robot platforms and sensing technologies.

6.6 Discussion

In this chapter, we presented a landmark placement method for mobile robot navigation. With high confidence, it guarantees a bound on the maximum deviation of the robot from its planned trajectory. In the landmark placement phase, our approach approximates the real motion and sensor models by their linearized versions to efficiently evaluate the guarantee. In the subsequent validation stage, we apply a Monte Carlo simulation using the real system dynamics to check if the placed landmark set satisfies the deviation guarantee also for the possibly nonlinear models. Our algorithm is customizable to specific robotic systems and navigation tasks and inherently chooses the appropriate number of landmarks needed. In extensive experiments, we demonstrated that our method can efficiently handle large-scale scenarios and that it outperforms baseline approaches. Furthermore, we applied our algorithm successfully to create landmark configurations for several simulated and real-world navigation tasks in which common robot platforms navigated reliably, also in long-term experiments.

Chapter 7

Robust Landmark Placement for Navigation

As described before, accurate navigation is a key capability of autonomous mobile robots. Robots that observe the landmarks placed by our approach from the previous chapter stay close to their desired trajectories with high confidence. In this chapter, we modify and extend the methods from the previous chapter in order to handle scenarios in which the robot's line of sight to some of the placed landmarks is unpredictably obstructed, for example by other vehicles operating in the same environment. In order to achieve the desired robustness against missing landmarks, our approach typically needs to increase the number of placed landmarks, as unpredicted obstructions of landmarks can drastically reduce the navigation performance of the robot. Our algorithm is highly efficient, as it applies the linearization of the navigation cycle from Chapter 5 and combines it with submodular optimization techniques. In extensive experiments, also carried out with a real robot, we demonstrate the capabilities of our method and show that it enables robust autonomous navigation in practice.

In many robotics applications like logistics and transportation, mobile robots typically share the environment with other vehicles or with humans. These dynamic obstacles can block the robot's line of sight to a landmark unpredictably. Furthermore, some types of landmarks, like colored markers painted to walls or battery-powered active beacons, can wear out and fail over time, leading to a reduced navigation accuracy of the robot. Therefore, in applications in which these situations are likely, it is important to take them into account when placing artificial landmarks.

In this chapter, we modify and extend the methods from Chapter 6 resulting in an approach to landmark placement that robustly handles scenarios in which a certain number

of landmarks can be missing during operation. Our approach bounds the trace of the covariance of the robot's deviation throughout the whole trajectory. Thereby, it effectively bounds the deviation of the robot from its desired trajectory for all dimensions of the state space with high confidence. Our approach aims at minimizing the number of landmarks needed, while still satisfying the bound on the trace of the covariance even if any k of the placed landmarks are simultaneously not observable during the execution of the whole navigation task. By choosing k, the user can trade off the number of landmarks against the robustness.

Our approach has several characteristics that make it especially useful in practice: The robustness against missing landmarks allows the application of smooth fallback procedures: If the robot does not observe a placed landmark for some time, it can send a message to the maintenance personnel and can still travel back safely to its parking position, leaving the workspace unobstructed for others. Similar to the approach from the previous chapter, we assume that the robot is steered by an LQR controller and linearize the model of the entire navigation cycle using the methods presented in Chapter 5. Due to this linearization, our approach is highly efficient and therefore can be utilized even in large-scale scenarios. To achieve the desired robustness in the landmark placement, our approach extends the conservative approximation of the landmark observability applied in the previous chapter, so that it only depends on the desired bound specified by the user. Furthermore, considering the trace of the covariance of the deviation instead of the maximum deviation allows us to utilize techniques from submodular function optimization, which come with formal approximation guarantees.

This chapter is organized as follows. In the next section, we formally state the problem definition. In Section 7.2, we introduce our landmark placement algorithm, whose theoretical properties we analyze in Section 7.3. Finally, we provide extensive experiments carried out both in simulation and with a real robot.

7.1 Problem Statement

We consider the problem of landmark placement for a mobile robot that repeatedly and autonomously travels along the same discretized trajectory $\mathcal{T} = \left(\mathbf{x}^\star_{0:T}, \mathbf{u}^\star_{1:T}\right)$, consisting of the desired states $\mathbf{x}^\star_t$ and the desired controls $\mathbf{u}^\star_t$ at each time step t. We assume that the navigation cycle of the robot can be described by the framework introduced in Section 5.1. To briefly recapitulate, for localization the robot has a map of the positions and the unique identities of the landmarks $\mathcal{A} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n\}$ in the area surrounding the trajectory and is equipped with a sensor to observe them. At every time step t, the robot takes a noisy observation $\mathbf{z}^{\mathcal{A}}_t$ of the landmarks inside its actual field of view, updates its state estimate in a localization algorithm, and executes a control command $\mathbf{u}_{t+1}$, propagating its state $\mathbf{x}_t$ according to a noisy motion model. It selects the control commands depending

on the difference between its localization estimate and its desired trajectory using a linear-quadratic regulator (LQR) feedback controller [13]. In this closed-loop system for autonomous navigation, we consider a discrete set of possible landmark locations $\mathcal{V}$, and aim to select the subset $\mathcal{A} \subseteq \mathcal{V}$ of landmarks for placement that are most beneficial for the navigation task.

In this chapter, we follow the idea of Bayesian A-optimal design [86], i.e., we aim at placing landmarks so that the trace of the covariance matrix $S_t^{\mathcal{A}}$ of the expected distribution $p(\mathbf{x}_t - \mathbf{x}_t^\star \mid \mathcal{A})$ stays below a user-defined threshold for every time step $t \in [0, T]$. Here and in the following, we denote the covariance matrix S_t of the expected distribution as $S_t^{\mathcal{A}}$ to make explicit its dependency on the positions of the landmarks in the environment. By bounding the covariance of the expected distribution, we effectively bound the deviation of the robot from its desired trajectory for all dimensions of the state space with high confidence.

To guarantee safe operation even if up to k landmarks are missing, we aim at finding

$$\mathcal{A}^\star = \underset{\mathcal{A} \subseteq \mathcal{V}}{\operatorname{argmin}} \, |\mathcal{A}| \tag{7.1}$$

subject to

$$\max_{\mathcal{B} \subseteq \mathcal{A}, |\mathcal{B}| \leq k} tr\left(S_t^{\mathcal{A} \setminus \mathcal{B}}\right) \leq \epsilon_t \quad \forall t \in [0, T]\,. \tag{7.2}$$

This is the smallest set of landmarks $\mathcal{A}^\star$ that ensures a bounded trace of the covariance of the expected distribution. In particular, we ensure that the trace of the covariance $S_t^{\mathcal{A}^\star \setminus \mathcal{B}}$ stays below ϵ_t for all $t \in [0, T]$, even if any subset $\mathcal{B} \subseteq \mathcal{A}^\star$ with $|\mathcal{B}| \leq k$ is not observable during operation. Here, ϵ_t is a user-defined bound that can be set for each part of the trajectory individually. This allows the user to specify lower ϵ_t values for critical parts of the trajectory in which a higher accuracy in navigation is required.

7.2 Efficient and Robust Landmark Placement

To efficiently place landmarks in the above-described framework, two important issues have to be addressed: handling the combinatorial structure of the problem stated in Equations (7.1) and (7.2) and estimating the covariance $S_t^{\mathcal{A}}$ of the expected distribution of the deviation of the robot. Our efficient solution to the combinatorial optimization problem is presented in Sections 7.2.2 and 7.2.3.

In general, $S_t^{\mathcal{A}}$ cannot be estimated in closed form, so one solution that is often applied is to approximate it via Monte-Carlo simulation. Monte-Carlo simulation can deal, for example, with non-linearities due to discontinuities in the observability of landmarks depending on the actual state of the robot, but it is computationally demanding. In contrast to that, we follow the approach taken in the previous chapter and introduce a

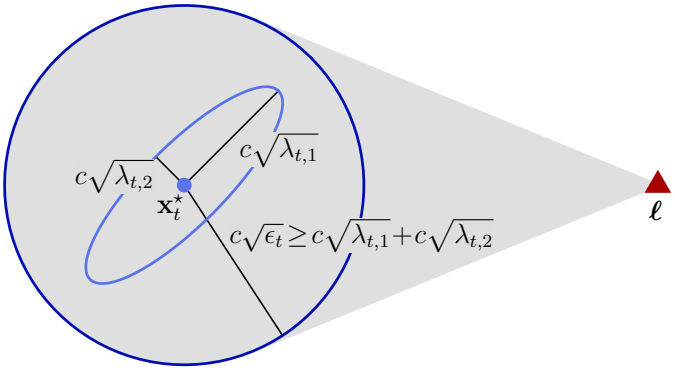

Figure 7.1: Desired robot state $\mathbf{x}_t^\star$ and landmark $\boldsymbol{\ell}$. The $(1-\delta)$ confidence regions of all the covariances of expected distributions that satisfy the bound in Equation (7.2) (one example shown in light blue) are inside the circular region with radius $c\sqrt{\epsilon_t}$ (dark blue). The landmark $\boldsymbol{\ell}$ is ϵ_t-observable at time t if it is inside the sensor range of every pose in the dark blue circle and if the shaded area is free of obstacles that would block the robot's line of sight to the landmark.

conservative approximation of the landmark observability. Using this approximation, we can estimate $S_t^{\mathcal{A}}$ efficiently by applying the linearization and recursive calculation of expected distributions described in Section 5.2. In the experimental section, we demonstrate the conservativeness of our approximation in practice and in Monte-Carlo simulations.

7.2.1 Observability Constraints

For most types of landmark detection sensors (e.g., cameras, RFID readers, and laser range finders), the ability to observe a landmark $\boldsymbol{\ell}$ changes with the state $\mathbf{x}_t$ of the robot due to a limited sensor range or obstacles concealing the landmark. At the time of landmark placement, the concrete robot state $\mathbf{x}_t$ at time t and therefore also the actual observability of $\boldsymbol{\ell}$ at time t are not yet known. To deal with this fact, we follow the basic idea of the approach from the previous chapter and conservatively approximate the observability of landmarks. However, the approximate observability at time t from the previous chapter recursively depends on the observed landmarks at all time steps previous to t and does not take into account that those landmarks could be obstructed from the view of the robot. Therefore, we cannot apply that approximate observability of landmarks, but introduce a new approximation of the observability of landmarks that depends only on the restriction of the covariance defined in Equation (7.2), which the final landmark set shall guarantee:

We consider a landmark as ϵ_t-*observable* at time t only if it is observable with a probability of at least $(1-\delta)$ according to every expected distribution for which the bound in Equation (7.2) holds, i.e., for which $tr(S_t^{\mathcal{A}}) \leq \epsilon_t$. Figure 7.1 visualizes the

concept of ϵ_t-observability in an example with a two dimensional robot state $\mathbf{x}_t = [x_t, y_t]$. Note that in the experiments, we consider a three dimensional robot state $[x_t, y_t, \theta_t]$.

For evaluating the ϵ_t-observability of a landmark $\boldsymbol{\ell}$, we use the following insights: As stated in Section 5.2, all estimated expected distributions are Gaussians. The $(1-\delta)$-confidence region of a Gaussian is an ellipsoid with principal axes of length $c\sqrt{\lambda_{t,i}}$. Here, $\lambda_{t,i}$ are the eigenvalues of the covariance $S_t^{\mathcal{A}}$ and $c = c(\delta, \dim(\mathbf{x}_t))$ is a constant that depends only on the probability δ and the dimensionality of the state space. If the bound in Equation (7.2) holds, it holds that $tr(S_t^{\mathcal{A}}) \leq \epsilon_t$ and therefore $\lambda_{t,i} \leq \epsilon_t$ for all eigenvalues $\lambda_{t,i}$ of $S_t^{\mathcal{A}}$. Consequently, at time t, at least $(1-\delta)$ of the probability mass of every Gaussian that satisfies Equation (7.2) lies inside the sphere K with radius $c\sqrt{\epsilon_t}$ and center $\mathbf{x}_t^\star$. If a landmark $\boldsymbol{\ell}$ is observable from within every state inside K, we define it to be ϵ_t-observable.

When estimating $S_t^{\mathcal{A}}$ for a given set $\mathcal{A}$ of landmarks, we consider only those landmarks as observable that are ϵ_t-observable and apply the efficient estimation scheme for expected distributions introduced in Section 5.2. Note that in the definition of the ϵ_t-observability we assume that the bound in (7.2) holds for $\mathcal{A}$. Hence, if the bound holds, then also the approximation of the observability, which was applied in the evaluation of the bound, is conservative.

7.2.2 Objective Function

Being able of evaluate Equation (7.2) for a given landmark set $\mathcal{A}$ makes it possible to run a brute force search on the power set $\mathcal{P}(\mathcal{V})$ of all possible landmark positions to find the optimal landmark set satisfying Equation (7.2). However, as $\mathcal{P}(\mathcal{V})$ grows exponentially with the number of possible landmark locations $|\mathcal{V}|$, we apply an efficient approximation instead. Building on techniques of Krause *et al.* [57], we now show how the overall problem defined in Equations (7.1) and (7.2) can be reformulated in a way that admits highly efficient approximation algorithms, which take into account that up to k landmarks can be hidden from the view of the robot.

As a first step, we define the reduction of the trace of the covariance of the expected distribution induced by the observations of the landmarks in $\mathcal{A}$ as

$$F_t(\mathcal{A}) = tr(S_t^{\varnothing}) - tr(S_t^{\mathcal{A}}) \tag{7.3}$$

for every time step t. We truncate this function at the target value $tr(S_t^{\varnothing}) - \epsilon_t$, leading to the function

$$F_t(\epsilon_t, \mathcal{A}) = \min\left(F_t(\mathcal{A}),\, tr(S_t^{\varnothing}) - \epsilon_t\right) . \tag{7.4}$$

Note that this function achieves its maximum value if and only if the target condition $tr(S_t^{\mathcal{A}}) \leq \epsilon_t$ is satisfied. We take into account the robustness against up to k missing

Algorithm 8 Iterative landmark placement robust to obstructions from view

Input: $\mathcal{V}$, k, $\epsilon_{0:T}$
Output: $\mathcal{A}$
  $\mathcal{A} \leftarrow \varnothing$
  while $F(k, \epsilon_{0:T}, \mathcal{A}) < c$ **do**
    $\boldsymbol{\ell}^\star \leftarrow \operatorname{argmax}_{\boldsymbol{\ell} \in \mathcal{V}} \; F(k, \epsilon_{0:T}, \mathcal{A} \cup \{\boldsymbol{\ell}\})$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{\ell}^\star\}$
  end while
  return $\mathcal{A}$

landmarks in the objective function by considering the average over $F_t(\epsilon_t, \mathcal{A} \setminus \mathcal{B})$ for all possible subsets of missing landmarks $\mathcal{B}$:

$$F_t(k, \epsilon_t, \mathcal{A}) = \frac{1}{\sum_{i=0}^{k} \binom{|\mathcal{A}|}{i}} \sum_{\mathcal{B} \subseteq \mathcal{A}, |\mathcal{B}| \leq k} F_t(\epsilon_t, \mathcal{A} \setminus \mathcal{B}) \,. \tag{7.5}$$

Similar to the function in Equation (7.4), this function achieves its maximum value $tr(S_t^{\varnothing}) - \epsilon_t$ if and only if the target condition under k-robustness $tr(\hat{S}_t^{\mathcal{A} \setminus \mathcal{B}}) \leq \epsilon_t$ is satisfied for all $\mathcal{B}$. We finally consider multiple time steps or a whole trajectory by using the same averaging procedure, leading to

$$F(k, \epsilon_{0:T}, \mathcal{A}) = \frac{1}{T+1} \sum_{t=0}^{T} F_t(k, \epsilon_t, \mathcal{A}) \,. \tag{7.6}$$

Due to its construction, this function takes on its maximum value

$$c = \frac{1}{T+1} \sum_{t=0}^{T} tr(S_t^{\varnothing}) - \epsilon_t \tag{7.7}$$

if and only if the condition defined in Equation (7.2) is satisfied. With this, we can re-formulate the problem definition stated in Equations (7.1) and (7.2) in terms of F as

$$\mathcal{A}^\star = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}} |\mathcal{A}| \;\; \text{s.t. } F(k, \epsilon_{0:T}, \mathcal{A}) = c \,. \tag{7.8}$$

7.2.3 Landmark Placement Algorithm

Since problems of the type defined in Equation (7.8) are typically **NP**-hard (see Section 7.3 for details), we apply a greedy iterative landmark placement algorithm that finds an approximate solution to (7.8). The procedure is stated in Algorithm 8. The computation of the argmax operator in the algorithm evaluates each landmark $\boldsymbol{\ell}$ individually, which makes it well-suited for parallel computing.

Note that due to the usage of the ϵ_t-observability in the evaluation of F, which is a conservative approximation of the observability only if $F(k, \epsilon_{0:T}, \mathcal{A}) = c$, stopping the algorithm before $F(k, \epsilon_{0:T}, \mathcal{A})$ reaches c leads to landmark sets that can perform arbitrarily badly. However, for the final output set $\mathcal{A}$, the observability, and therefore also the condition in Equation (7.2), is approximated conservatively.

7.2.4 Practical Considerations

Our algorithm can be used to guarantee a collision-free trajectory execution with high confidence. For that, we use the same insights as for the approximation of the observability. We choose the bound ϵ_t on the trace such that the nearest static obstacle is at least $c\sqrt{\epsilon_t}$ away from the desired state $\mathbf{x}_t^\star$ for every t. To avoid collisions with moving obstacles without breaking the bound on the trace, the robot needs to stop if its path is blocked and wait until the moving obstacle left the corridor with width $c\sqrt{\epsilon_t}$ around the desired trajectory. Note that this collision avoidance strategy conservatively approximates the deviation of the robot from its desired trajectory, which is the value that the approach from the previous chapter optimizes, with the trace of the covariance of the expected distribution, which we optimize in this chapter.

In some applications, it is necessary for the robot to operate continuously, executing the same trajectory repeatedly without being able to readjust its pose in between runs. If the desired final state $\mathbf{x}_T^\star$ of the robot equals the initial state $\mathbf{x}_0^\star$, our method can be adjusted similar to the method in the previous chapter to guarantee bounded traces even for a continuous operation of the robot. By setting ϵ_T to at most the minimum eigenvalue of $S_0^{\mathcal{A}}$, our algorithm produces a landmark set $\mathcal{A}$ that guarantees that $S_T^{\mathcal{A}}$ is governed by $S_0^{\mathcal{A}}$, which enables a continuous safe operation.

7.3 Approximation Bound

In this section, we provide a theoretical motivation for our approximation algorithm, which rests on the concept of submodularity, a natural diminishing returns property, which is presented in detail in Section 3.4 of the background chapter. For landmark sets, submodularity states that adding a landmark to an already large set of landmarks $\mathcal{C}$ results in a smaller increase in the objective function than adding the same landmark to a subset of $\mathcal{C}$. Concretely, a function F is called submodular if for all $\mathcal{A} \subseteq \mathcal{C} \subseteq \mathcal{V}$ and all landmarks $\boldsymbol{\ell} \in \mathcal{V} \setminus \mathcal{C}$

$$F(\mathcal{C} \cup \{\boldsymbol{\ell}\}) - F(\mathcal{C}) \leq F(\mathcal{A} \cup \{\boldsymbol{\ell}\}) - F(\mathcal{A})\,. \tag{7.9}$$

For submodular functions, problems of the type defined in Equation (7.8) are called *submodular set cover* problems, and are **NP**-hard in general [31, 109]. However, for

these problems, Wolsey [109] showed that for greedy solutions $\mathcal{A}_{\text{greedy}}$, such as those produced by Algorithm 8, it holds that

$$|\mathcal{A}_{\text{greedy}}| \leq |\mathcal{A}^\star| \Big(1 + \log \max_{\boldsymbol{\ell} \in \mathcal{V}} F(\{\boldsymbol{\ell}\}) \Big) , \tag{7.10}$$

and under natural complexity-theoretic assumptions, no efficient algorithm can provide better solutions (see Section 3.4 for details). Hence, such greedy solutions $\mathcal{A}_{\text{greedy}}$ are near-optimal for submodular set cover problems.

Therefore, the key question is whether (or under which conditions) our objective function $F(k, \epsilon_{0:T}, \mathcal{A})$ for landmark placement is monotonic and submodular. First, note that the steps with which we build $F(k, \epsilon_{0:T}, \mathcal{A})$ from $F_t(\mathcal{A})$ as described in Section 7.2.2 preserve monotonicity and submodularity. This is due to the fact that they only apply nonnegative linear combinations and truncation, which are operations that preserve these properties (see Fujito [34] and Krause *et al.* [57]). Therefore, if $F_t(\mathcal{A})$ would be monotonic and submodular, also $F(k, \epsilon_{0:T}, \mathcal{A})$ would have these properties. However, even though Das and Kempe [21] have shown that, under conditions slightly different from ours, variance reduction is monotonic and submodular, this result cannot be extended to $F_t(\mathcal{A})$. In Appendix 10 we present a detailed theoretical examination of the non-submodular behavior of $F_t(\mathcal{A})$. In practice, though, violations of the submodularity property of $F_t(\mathcal{A})$ appear only seldom, and when they appear, the property is only slightly violated. This insight stems from an experiment in which we randomly sampled one million three-dimensional robot poses and corresponding covariance matrices. For each robot pose, we randomly sampled a set $\mathcal{C}$ of two to eight observable landmarks, a non-empty strict subset $\mathcal{A} \subset \mathcal{C}$, and one additional observable landmark $\boldsymbol{\ell}$. In this experiment, only 0.56% of the one million samples did not satisfy the submodularity inequality stated in Equation (7.9). In the cases in which the submodularity property was not satisfied, the value by which the inequality was violated was 0.042% of the value of $F(\mathcal{A})$ on average and 1.4% in the maximum case.

7.4 Experimental Evaluation

We evaluated our approach in extensive experiments both with a simulated differential drive robot and with a real holonomic drive robot. For these robots, the state $\mathbf{x}_t$ of the robot can be described by its pose $[x_t, y_t, \theta_t]$ in the 2d-plane. We assume that the robot is equipped with a landmark detection sensor with a circular field of view and $5\,\mathrm{m}$ sensor range. In the different experiments, the sensor can observe either landmarks placed on the walls or landmarks placed on the ceiling of the environment, resulting in different sets $\mathcal{V}$ of possible landmark locations. In all experiments, we set the allowed maximum trace ϵ_t to 0.05 for all time steps t and the probability δ in the definition of the ϵ_t-observability to 1%.

This section does not contain an experimental comparison between this approach and other approaches, as we present a detailed comparison between our different approaches in Chapter 8.

7.4.1 Evaluation of Robustness

In the first set of simulation experiments, we evaluated the robustness of our landmark sets against obstructions of landmarks from the view of the robot. To this end, we considered the two trajectories shown in Figure 7.2, corresponding to a sweeping pattern in an obstacle-free environment and a surveillance task in an environment with obstacles. For the sweeping trajectory, our approach placed 10, 15, and 19 landmarks assuming at most zero, one, and two missing landmarks, respectively. For the surveillance task, our approach placed 12, 20, and 27 landmarks. To evaluate the effects of the linear approximation applied in the landmark placement method, we conducted Monte-Carlo simulations using the real, non-linear models. In the simulations, we estimated the traces $tr_{\text{MC}}(S_t^{\mathcal{A}})$ of the expected distributions using the empirical distributions gained from the deviations $\mathbf{x}_t - \mathbf{x}_t^\star$ observed in 1,000 simulated executions of the trajectory in each scenario. For all six landmark configurations and all possible combinations of k missing landmarks, the Monte-Carlo simulations resulted in traces that were below the bound $\epsilon_t = 0.05$ for all time steps t. The maximum value 0.0439 of the traces in simulation occurred in the surveillance scenario for $k = 0$.

7.4.2 Landmark Placement for Changing Bounds

In the second set of simulation experiments, we demonstrate the ability of our approach to place landmarks for values of ϵ_t which vary along the trajectory. We applied our approach on the pick-and-place trajectory shown in Figure 7.3. One of the shown placements results from specifying a higher demand for accuracy in the pick-up and deposit zones and another placement results from increasing the required robustness against missing landmarks. As can be seen in the figure, these two requirements in this case lead to the same number of landmarks, but to different locations.

7.4.3 Long Term Evaluation on a Real Robot

Finally, we evaluated the landmark sets placed by our approach also on the real robot shown in Figure 7.4. The robot is equipped with Mecanum wheels for omnidirectional motion and with two SICK S300 laser scanners mounted on opposite corners of the robot, providing a 360° field of view. The lasers can detect reflective markers, whose unique landmark IDs we calculated using a nearest neighbor heuristic. In a training run, we calibrated the motion noise and sensor noise of this specific robot, and used the calibrated

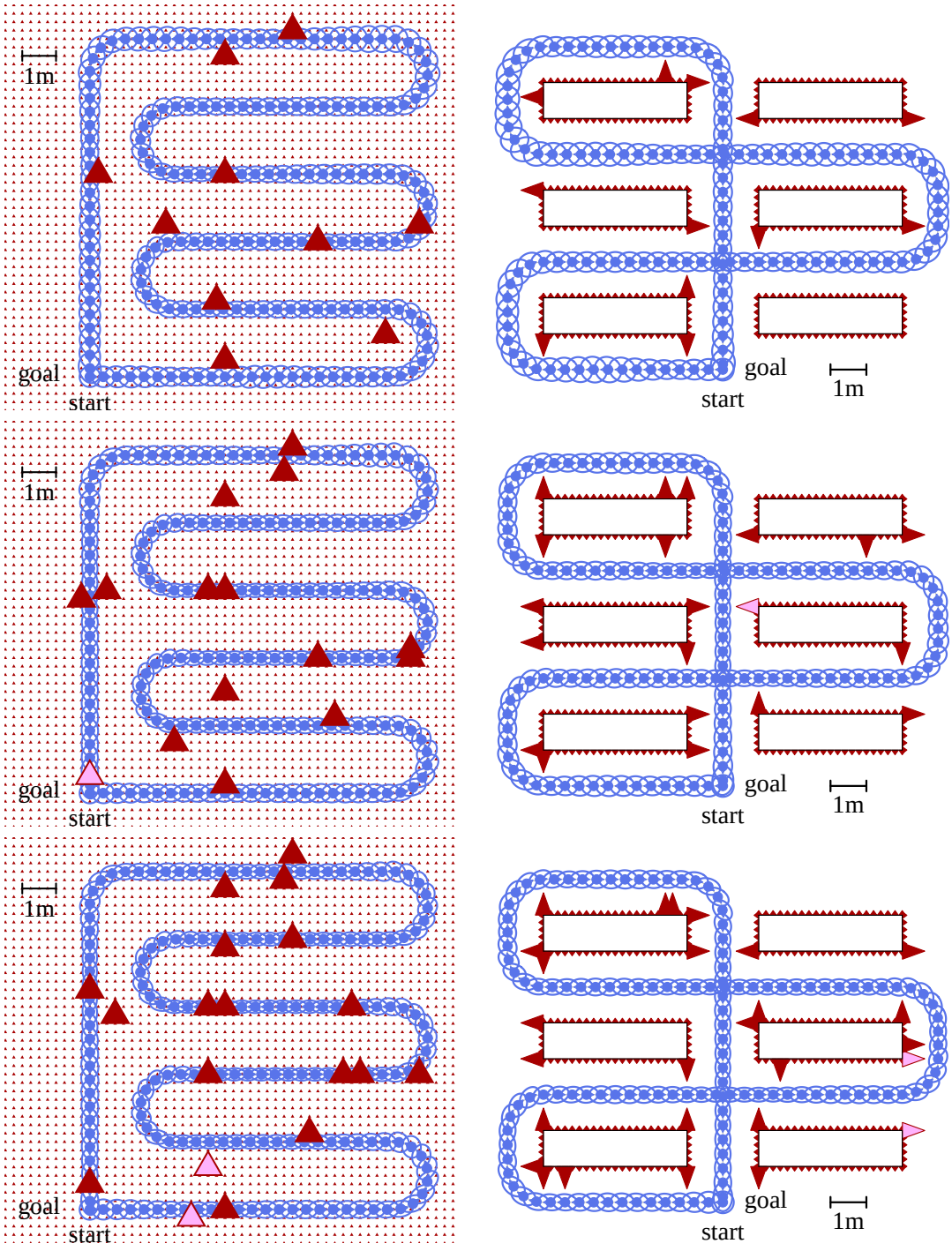

Figure 7.2: A sweeping task (left) and a surveillance task (right) with the 99% confidence ellipses of the expected distributions (blue) when observing all landmarks (red triangles) in the sets that our algorithm placed for at most $k = 0$ (top), 1 (middle), or 2 (bottom) missing landmarks. The k-subsets of placed landmarks whose absence resulted in the highest simulated maximum trace are shown in pink. The red dots indicate the sets of possible landmark locations $\mathcal{V}$.

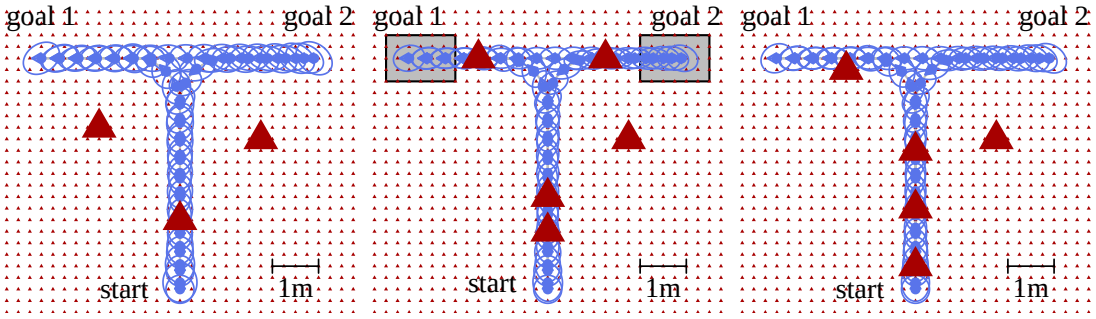

Figure 7.3: Pick-and-place trajectory and landmark sets placed by our approach for a constant ϵ_t of 0.05 for $k = 0$ (left) and $k = 1$ (right), and for $k = 0$ and $\epsilon_t = 0.03$ for the time steps in which the robot is inside the pick-up and deposit zones (gray areas) and 0.05 outside (middle). The trajectory goes from the start to goal 1, then to goal 2, and back to the start.

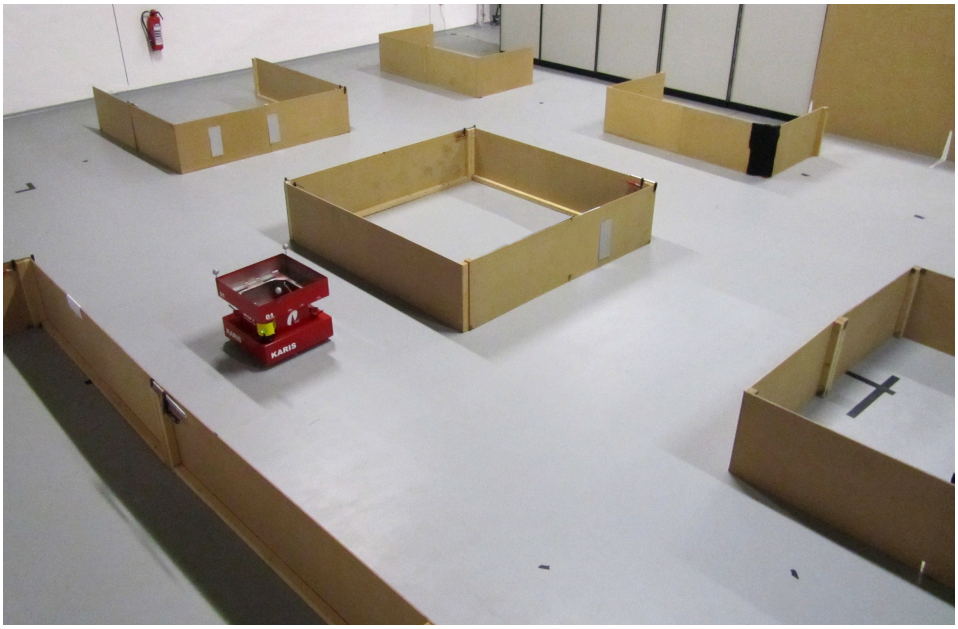

Figure 7.4: The KARIS robot in the experimental environment. The three stripes of reflective tape on the walls can be detected in the laser scans and are part of the robust landmark set placed by our approach. As can be seen in the picture, one landmark is currently hidden by a black cloth.

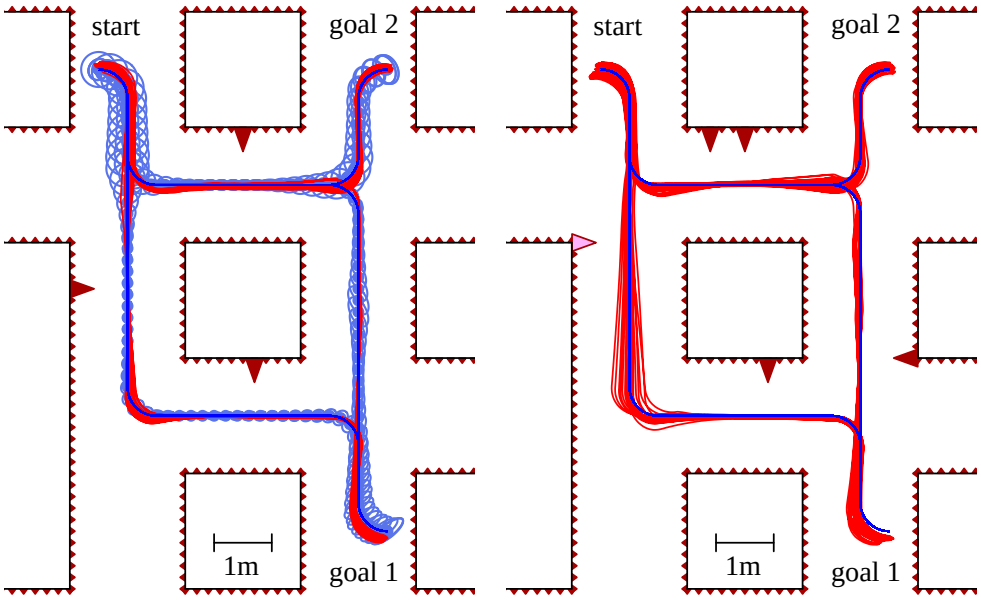

Figure 7.5: Landmark sets (red triangles), desired robot path (blue) and actual robot paths (red) for $k = 0$ (left), and $k = 1$ (right). For $k = 0$, also the 99% confidence regions of the expected distributions in the linearized system are shown (light blue). For $k = 1$, each landmark was hidden during 10 runs of the robot. The landmark whose absence resulted in the largest deviations is marked in pink. The trajectory goes from the start to goal 1, back to the start, then to goal 2 and again back to the start.

parameters in the linearized models for landmark placement. Note that compared to Section 6.5.3, in which we present experiments with the same robot, the calibrated motion noise for this experiment is considerably smaller, as here the floor on which the robot moves is smoother. This can be seen when comparing the images in Figures 6.7 and 7.4. The trajectory and the landmarks that our approach placed to ensure continuous long term operation (see Section 7.2.4) are shown in Figure 7.5. To evaluate the placed landmark sets, the robot continuously executed the trajectory several times. Observing only the landmarks placed for $k = 0$, the robot autonomously executed 20 runs of the trajectory, continuously operating for one hour.

We estimated the traces of the expected distributions from the empirical distributions of the deviations measured by a Motion Analysis motion capture system with ten digital Raptor-E cameras. The leftmost frequency plot in Figure 7.6 displays the estimated traces for $k = 0$. As can be seen in the figure, the estimated traces stayed considerably below 0.05, with a maximum of 0.0087 occurring close to goal 1. The measured maximum translational deviation of the robot from the desired trajectory was $0.36\,\mathrm{m}$. For $k = 1$, the robot executed 50 runs of the trajectory during 2.5 hours of continuous operation. During operation, each landmark was hidden from the robot during ten runs. The traces

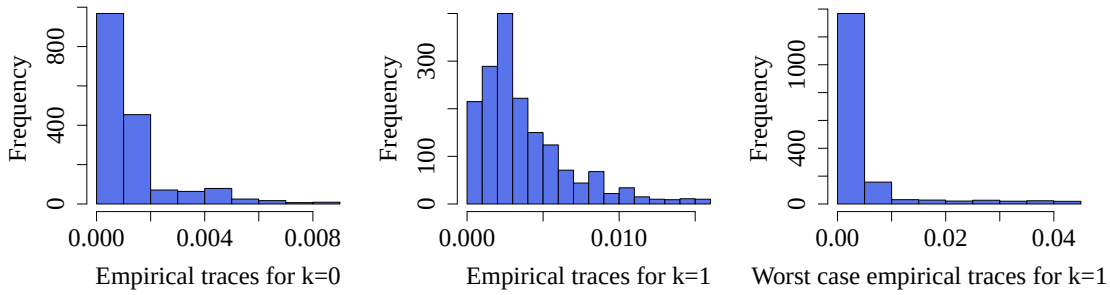

Figure 7.6: Frequency plots of the traces of the covariances of the empirical expected distributions calculated from long-term runs of the real robot. Displayed are the traces for $k = 0$ (left) and the traces for $k = 1$ estimated from all runs of the robot (middle) and from only those runs in which the worst-case landmark was hidden from the view of the robot (right).

estimated from the whole dataset and also the ones estimated from each block of ten runs in which one landmark was hidden stayed below 0.05, as can be seen in the middle and right plots in Figure 7.6. The maximum value, 0.0444, occurred close to the lower left corner of the trajectory when the landmark marked in Figure 7.5 was hidden. Throughout the whole experiment, the measured maximum translational deviation from the desired trajectory was $0.45\,\mathrm{m}$.

7.5 Discussion

In this chapter, we presented a novel method to trajectory-dependent landmark placement for mobile robot navigation, which is robust against missing landmarks. This method builds on and extends the method from the previous chapter. It keeps the traces of the covariances of the expected distributions of all robot states below a user-defined threshold, effectively bounding the uncertainty in all dimensions of the state space. The linearized objective function in our method takes into account the full specification of the navigation task. We evaluate this function efficiently by combining the estimation technique presented in Chapter 5 with techniques from submodular function optimization. Extensive experiments, also with a real robot, demonstrate that the robustness against missing landmarks resulting from our approach is guaranteed in practice, even in continuous long term operation.

Chapter 8

Comparison between Landmark Placement Methods

> In this chapter, we give an overview of the similarities and the differences between the three landmark placement approaches presented in the previous chapters. Furthermore, we compare the landmark sets resulting from our approaches in extensive simulation experiments.

In the previous chapters, we presented three different approaches to optimally placing artificial landmarks for mobile robots that repeatedly execute the same trajectory. In this chapter, we give a detailed comparison of these three approaches. The insights gained from our comparison can be used to decide which of our approaches is best suited for a specific task at hand. The discussion of benefits and drawbacks of the different individual components of our approaches can also be a valuable source of information when constructing new landmark placement methods.

8.1 Properties

In this section, we give an overview of the similarities and the differences between the three landmark placement approaches presented in Chapters 4, 6, and 7. To distinguish between the different approaches, we refer to them in the following with the names

- *localization* for the method presented in Chapter 4, which aims at optimizing the localization performance of the robot,
- *deviation* for the approach from Chapter 6, which aims at optimizing the navigation performance of the robot by bounding its deviation from the desired trajectory, and
- *robust* for the method presented in Chapter 7, which aims at optimizing the navigation performance of the robot in a way that robustly handles scenarios in which landmarks are unpredictably hidden from the view of the robot.

All three approaches consider landmark placement as a trajectory-dependent problem. Given a pre-defined trajectory that is intended to be executed many times by a mobile robot, all three methods aim at finding a set of landmark positions along this trajectory that are optimal for the robot during operation. When placing landmarks, all methods explicitly take into account the full specification of the applied robot, consisting of its motion model, its sensor model, and the controller that selects its control commands. All our approaches deal with the noise and errors that occur in the motion and sensor observations of real robots by using probabilistic formulations of the motion model and the sensor model of the robot.

However, there are also considerable differences between the approaches. The localization method aims at optimizing the expected localization performance of the robot. In contrast to that, the other two approaches assume that the robot uses its localization estimate to adjust its navigation behavior via a feedback controller and aim at optimizing the navigation behavior itself. Figure 8.1 illustrates the difference between the localiza-

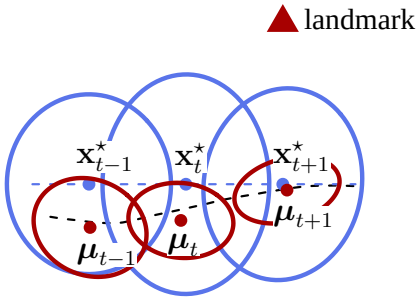

Figure 8.1: Detail from a navigation task with one landmark. Shown are the desired trajectory (dashed blue line) and the actually executed trajectory (dashed black line) in one simulated run of the robot. Additionally, the figure shows the means and 99% confidence ellipses of the posterior distributions (red), which capture the localization accuracy in the simulated run of the robot and the means and 99% confidence ellipses of the expected distributions (blue), which capture its navigation accuracy averaged over all possible runs.

tion performance and the navigation performance in terms of confidence ellipses. The red confidence ellipses visualize the uncertainty of the localization system of the robot during one individual execution of the trajectory. They encode the uncertainty about the difference

$$\mathbf{x}_t - \boldsymbol{\mu}_t \tag{8.1}$$

between the real robot state $\mathbf{x}_t$ and the localization estimate $\boldsymbol{\mu}_t$ at every time step t, taking into account the concrete values of all previously executed control commands $\mathbf{u}_{1:t}$ and sensor observations $\mathbf{z}_{1:t}$. These confidence ellipses stem from the so-called posterior distributions $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ of the states $\mathbf{x}_t$ of the robot. When optimizing the expected localization performance of the robot, we aim at minimizing the expected uncertainty

of these distributions with respect to the executed control commands and observations. Concretely, for a function U that measures the uncertainty of the distribution, we aim at minimizing

$$\int U\big(p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\big)\, p(\mathbf{u}_{1:t}, \mathbf{z}_{1:t})\, \mathrm{d}(\mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,. \tag{8.2}$$

The equivalent of this equation in our localization approach is Equation (4.41), in which we use the differential entropy h as a measure for the uncertainty.

The blue confidence ellipses, on the other hand, stem from the expected distributions of the states of the robot with respect to all possible sensor observations and to all possible resulting control commands selected by a feedback controller. In the robotic systems that we consider in the deviation and in the robust approach, they encode the uncertainty about the difference

$$\mathbf{x}_t - \mathbf{x}_t^\star \tag{8.3}$$

between the real robot state $\mathbf{x}_t$ and the desired state of the robot $\mathbf{x}_t^\star$ at every time step t (see Lemma 5.1) and are typically considerably larger than the red ones (see Corollary 5.2). When optimizing the navigation performance of the robot, we aim at minimizing the uncertainty U of these distributions, i.e.,

$$U\left(\int p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\, p(\mathbf{u}_{1:t}, \mathbf{z}_{1:t})\, \mathrm{d}(\mathbf{u}_{1:t}, \mathbf{z}_{1:t})\right) = U\big(p(\mathbf{x}_t \mid \mathcal{A})\big)\,. \tag{8.4}$$

In our robust approach, for example, we use the trace of the covariance of the expected distribution to describe the uncertainty U of the navigation performance.

Formally, the only difference between the above defined uncertainties in localization and in navigation is the order in which the function U and the integral operator are applied in Equations (8.2) and (8.4). Note that this order is not interchangeable, as the measures for the uncertainty that we apply are not linear functions. In fact, they typically depend on the covariances of the distributions, which are quadratic functions.

Another difference between the localization approach and the other two approaches is that the localization approach applies Monte Carlo simulation to evaluate the integral in Equation (8.2), whereas the other two approaches apply the linearization and the efficient recursive calculation scheme from Chapter 5 to evaluate the expected distributions, i.e., the integral in Equation (8.4). The Monte Carlo simulation does not need linearized motion models and sensor models and can deal with arbitrary types of robot controllers, but is computationally demanding. The recursive calculation in the linearized system is highly efficient, which makes it possible to apply the deviation and the robust approach even in large-scale scenarios. However, the linearized recursive calculation is only applicable if the robot is steered with an LQR feedback controller. Therefore, also the deviation and the robust approach can only be applied to robots steered with LQR controllers. Furthermore, the quality of the linearized approximation depends strongly on the considered motion

model and sensor model. To deal with the highly non-linear jump discontinuities that occur in the sensor model at the border of the field of view of the robot, both approaches apply conservative approximations of the observability of landmarks.

Above, we discussed the similarities and differences between the two landmark placement approaches for navigation and the landmark placement approach for localization. In the following, we discuss the similarities and differences between the two landmark placement methods for navigation themselves:

The deviation approach bounds the deviation of the robot from its desired trajectory with high confidence by iteratively placing landmarks along the trajectory. This iterative placement maximizes the trajectory length between two consecutively observed landmarks without considering the possibility that a landmark could be unpredictably obstructed from the view of the robot. As can be seen in the experiments in the next section, the resulting landmark sets lead to an unreliable navigation behavior in situations in which at least one landmark is obstructed from view.

The robust approach, on the other hand, explicitly takes into account unpredictably missing landmarks. In order to deal with missing landmarks, this approach applies a conservative approximation that underestimates the observability of landmarks even more than the approximation applied in the deviation approach.

Furthermore, in contrast to the deviation approach, the robust approach does not use the deviation of the position of the robot from its desired trajectory as objective function but it uses the trace of the covariance of the expected distribution of the state of the robot. This trace can be considered as a conservative approximation of the deviation of the robot (see Section 7.2.4). The deviation of the position of the robot itself is highly non-submodular due to the strong influences that the position components and the orientation component of the considered covariance matrices have on each other via the motion model of the robot. Switching to the trace instead of the deviation allows us tot apply techniques from submodular function optimization in the robust approach.

Summing up, the deviation approach cannot deal with scenarios in which it is likely that landmarks are unpredictably obstructed from the view of the robot. The robust approach explicitly considers this possibility, but therefore needs to make stronger approximations than the deviation approach.

8.2 Experimental Evaluation

We compared our landmark placement approaches not only in theory, but also in experiments with a simulated differential drive robot steered by an LQR controller. We describe the state $\mathbf{x}_t$ of the robot at time t by its pose $[x_t, y_t, \theta_t]$ in the 2d-plane, and assume that the robot is equipped with a landmark detection sensor with a circular field of view and $5\,\mathrm{m}$ sensor range. For landmark placement with the robust approach, we set the allowed

maximum trace ϵ_t to 0.05 and the allowed maximum number k of simultaneously missing landmarks to 0 and 1, respectively. In order to achieve comparable results, we adjusted the parameters of the other approaches so that all methods placed the same numbers of landmarks for each considered trajectory.

We compared the landmark sets placed by the deviation approach, the robust approach, the localization approach, and by two straightforward heuristics. The *grid* and *random* heuristics place a given number of landmarks in the area observable by the robot, grid in a regular grid pattern and random at randomly sampled locations. To get scenario-independent results, we considered ten randomly chosen trajectories, each connecting six randomly sampled goal points in an area of $15\,\mathrm{m} \times 15\,\mathrm{m}$. For all trajectories, we considered a set $\mathcal{V}$ of possible landmark locations consisting of 3600 positions on a regular grid covering the whole environment. The sampled trajectories, together with

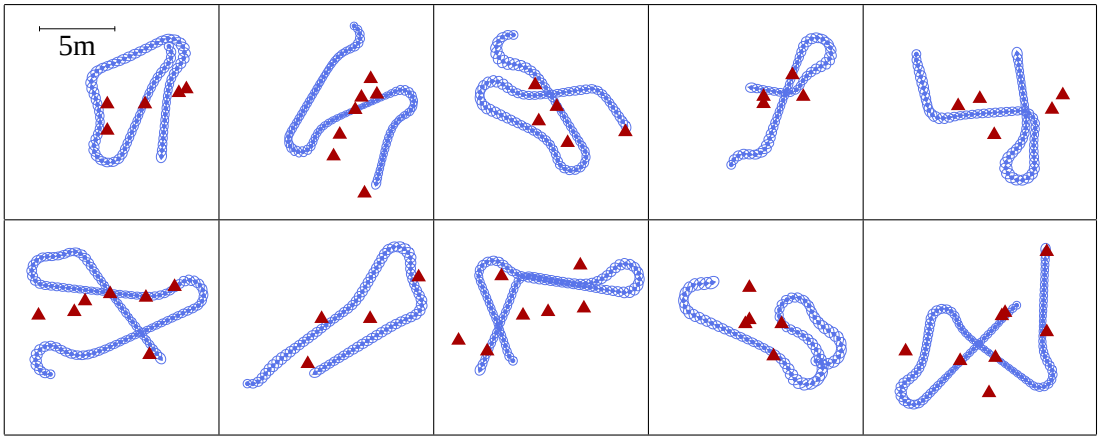

Figure 8.2: The ten randomly sampled trajectories used in the experiments and the corresponding 99% confidence ellipses of the expected distributions (blue) when the robot observes the landmarks placed by the robust approach for $k = 0$ obstructed landmarks (red triangles).

the landmarks placed by the robust approach for $k = 0$ can be seen in Figure 8.2. For clarity of presentation, the set $\mathcal{V}$ is not displayed in the figure. The placed landmark sets had average sizes of 5.7 and 9.3 for $k = 0$ and $k = 1$, respectively. We used Monte-Carlo simulations to evaluate the three quantities that our approaches optimize, namely the 99% quantile of the maximum translational deviation of the robot from its desired trajectory, the maximum of the traces of the covariances of the expected distributions of the states of the robot, and the expected information gain of the joint posterior distribution of $\mathbf{x}_{0:T}$ induced by the landmark observations. The results can be seen in Figure 8.3.

For $k = 0$ missing landmarks, the deviation and the robust approach both produce similar maximum traces of the covariances and deviations, but the deviation approach seems to behave unreliably in the case of a missing landmark. For $k = 1$, when the most influential landmark is hidden, it results in maximum traces and deviations that

	Deviation	Robust
Using our calculation scheme	20:04 min	17:25 min
Using van den Berg's method	27:22 min	28:27 min

Table 8.1: Runtime comparison of the deviation and the robust landmark placement approach when using our efficient method for calculating expected distributions and when using the calculation scheme by van den Berg *et al.* [11] instead.

are significantly higher than the ones produced by the robust approach and even the localization approach on a 5% level. In contrast to that, the robust approach, which explicitly takes missing landmarks into account, results in small maximum traces of the covariances and deviations also in the case when $k = 1$ landmark is missing.

For the maximum traces and deviations, which both describe the navigation performance of the robot, the localization approach resulted in worse values than the robust approach. On the other hand, for the expected information gain, which describes the localization performance of the robot, it yielded the best values.

The grid and random heuristics typically yielded the worst results. Only in the case in which one landmark is missing, the deviation method resulted in worse maximum traces and deviations than the grid heuristic.

Running multi-threaded on an Intel® Core™ i7 2.8GHz, the runtime of the localization approach for computing a single landmark set was 8:10 h on average. The runtimes for the deviation approach and the robust approach were significantly shorter due to the application of the linearization and of our efficient calculation scheme for expected distributions. The average runtimes for single-threaded implementations of these approaches are shown in Table 8.1. Additionally, the table shows the runtimes that the deviation and the robust landmark placement approach needed when applying the method by van den Berg *et al.* [11] for calculating expected distributions instead of our method from Chapter 5 inside the landmark placement algorithms. As can be seen from the table, the reduction of the runtime for calculating expected distributions resulting from our method also yields a reduction of the overall runtime of the landmark placement approaches.

8.3 Discussion

In this chapter, we gave a detailed comparison of the three approaches to landmark placement presented in the previous chapters. The discussion of the benefits and the drawbacks of the different approaches can be valuable when deciding which landmark placement method to use for a specific task at hand. We also compared the landmark sets resulting from our approaches in extensive simulation experiments. The results show that

the placed landmark sets yield a better localization and navigation performance of the robot than landmark sets placed by heuristics.

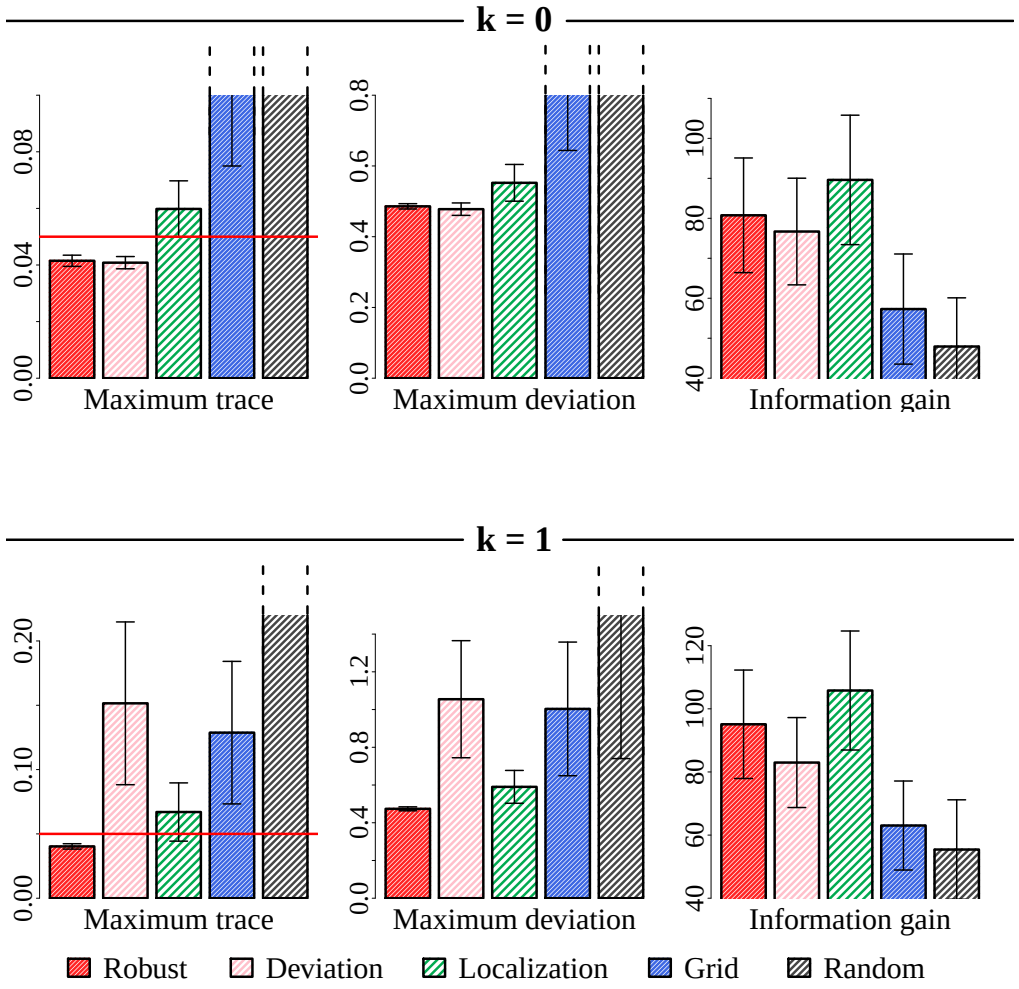

Figure 8.3: Means and 95% error bars resulting from simulations on ten randomly sampled trajectories. For $k = 1$, the results for the simulations in which the one most crucial landmark was missing are shown. The horizontal red line indicates the bound $\epsilon_t = 0.05$ that the robust approach guarantees with high confidence. The maximum deviation is stated in m.

Chapter 9

Landmark Deployment to Foster Data Association in SLAM

For mobile robots operating in previously unknown and unmapped environments, the landmark placement approaches from the previous chapters are not applicable and the robot is faced with the problem of simultaneous localization and mapping (SLAM). In this context, we consider the fundamental problem of data association, i.e., deciding if two observations stem from the same environment feature or from different ones. This problem is hard to solve correctly, especially in ambiguous environments. In this chapter, we consider a scenario where the robot can deploy a limited number of uniquely identifiable artificial landmarks along its path and use them afterwards as fixed anchors to ease the data association problem. Obviously, the choice of the positions for deploying the artificial landmarks is crucial as poor choices might prevent the robot from establishing accurate data associations. We present a novel approach for learning when to deploy landmarks to optimize the data association performance. We use Monte Carlo reinforcement learning for computing an optimal policy and apply a statistical convergence test to decide if the policy is converged and the learning process can be stopped. Extensive experiments in simulation and with a real robot demonstrate that our approach significantly outperforms baseline strategies.

In the previous chapters, we presented approaches for optimally placing artificial landmarks in a known environment. The goal of these methods was to improve the localization and navigation performance of a mobile robot that is equipped with a map of these landmarks. In this chapter, we consider a different scenario in which it is beneficial to deploy artificial landmarks. A mobile robot that is capable of autonomously deploying artificial

landmarks travels through a previously unknown and unmapped environment. During operation, the robot builds a map of the environment while localizing itself in this very map. This procedure is known as simultaneous localization and mapping (SLAM), and is one of the fundamental problems in mobile robotics. One of the key challenges in SLAM is that of data association, where the robot has to recognize previously observed places. In general, data association failures lead to inconsistent maps that cannot be used for navigation tasks. While highly effective methods for computing a map given the data associations have been developed in the past [37, 61, 80], the development of methods for robust data association is still an open research problem. In practice, data association quickly becomes intractable, particularly in ambiguous environments, as the complexity of the data association problem grows exponentially with the number of feature observations. One way of resolving ambiguities in the environment and supporting data association is to deploy artificial landmarks. This idea was already implemented in the well-known fairy tale Hansel and Gretel, in which the siblings deployed breadcrumbs to find their way back out of the forest. Robots, on the other hand, can drop radio-frequency identification (RFID) tags or similar uniquely identifiable landmarks [24, 50, 107]. Still there remains the question of deciding where and when to deploy the landmarks. This question becomes even more relevant when the robot can only carry a limited number of landmarks.

Our approach to learning a landmark deployment policy is designed to assist the SLAM system without interfering with the actual navigation task carried out by the robot. Consequently, the robot does not need to perform any detours for proper landmark deployment. Our method furthermore does not rely on information about the navigation task of the robot, such as its current target position. To compute the optimal policy, we apply actor-critic Monte Carlo reinforcement learning using the number of incorrectly estimated feature correspondences as performance measure. In an offline learning phase, we employ simulated episodes of robot navigation tasks. In order to make the resulting policies generalize well to different environments, our approach relies on general features like the remaining battery life time, the number of landmarks left on board, the distance to the closest deployed landmark, and a feature capturing the abstract local structure of the environment. To reduce the number of episodes required for learning, we employ a statistical convergence test to decide if the learned policy is converged and the learning process can be stopped. As a result, the robot can efficiently learn a policy for placing artificial landmarks so that the data association errors are greatly reduced.

This chapter is organized as follows. In the next section, we show how to integrate deployed landmarks into the SLAM posterior and how data association performance can be measured. In Section 9.2, we introduce our approach for learning artificial landmark deployment policies. Finally, we provide extensive experiments that demonstrate the effectiveness of the learned policies both in simulation and on a real robot.

9.1 Simultaneous Localization and Mapping with Deployed Landmarks

As described in Section 3.5, simultaneous localization and mapping (SLAM) refers to the problem of estimating the joint posterior distribution

$$p(\mathbf{x}_{0:T}, \mathbf{m}_{1:n}, c_{1:T} \mid \mathbf{u}_{1:T}, \mathbf{z}_{1:T}) \tag{9.1}$$

of the robot's poses $\mathbf{x}_{0:T}$ and the map $\mathbf{m}$, which consists of n features $\mathbf{m}_{1:n}$, given a set of robot motion commands $\mathbf{u}_{1:T}$ and a set of feature observations $\mathbf{z}_{1:T}$. Here, $c_{1:T}$ are the estimated data associations, i. e., the estimated identities of the map features perceived in the observations $\mathbf{z}_{1:T}$. Estimating these correspondences $c_{1:T}$ is an integral part of any solution to the SLAM problem. If the estimated correspondence c_t identifies the observation $\mathbf{z}_t$ as an observation of the map feature $\mathbf{m}_i$, we denote this as

$$c_t(\mathbf{z}_t) = \mathbf{m}_i \,. \tag{9.2}$$

For an overview of the existing estimation methods for $c_{1:T}$, see Section 3.5.1.

To facilitate the data association in SLAM, we consider a robotic system that can autonomously deploy a limited number k of uniquely identifiable landmarks. Given these additional landmarks, the SLAM posterior turns into

$$p(\mathbf{x}_{0:T}, \mathbf{m}_{1:n}, c_{1:T}, \boldsymbol{\ell}_{1:k} \mid \mathbf{u}_{1:T}, \mathbf{z}_{1:T}, \mathbf{z}^{\boldsymbol{\ell}}_{1:T}, c^{\boldsymbol{\ell}}_{1:T}) \,, \tag{9.3}$$

where $\boldsymbol{\ell}_{1:k}$ are the positions of the artificial landmarks, and $c^{\boldsymbol{\ell}}_{1:T}$ are the known identities of these landmarks perceived in the observations $\mathbf{z}^{\boldsymbol{\ell}}_{1:T}$. The observations $\mathbf{z}^{\boldsymbol{\ell}}_{1:T}$ of the deployed artificial landmarks and, in particular, the correspondences $c^{\boldsymbol{\ell}}_{1:T}$ refine the SLAM posterior, potentially making the data association problem more tractable by resolving ambiguities in the environment.

9.1.1 Measuring the Performance of Data Association

When deploying artificial landmarks, we aim at maximizing the performance of data association. To measure this performance, we count the number of incorrectly estimated map feature correspondences. It is not obvious how to define this number, because if the estimated feature correspondences of two observations originating from different environment features point to the same estimated map feature, one cannot decide which of the correspondences is wrong and which is right. To distinguish between the n estimated features that the robot marks in its map and the n' real features existing in the environment, we denote the environment features as $\mathbf{m}^{\star}_{1:n'}$ in contrast to the map features $\mathbf{m}_{1:n}$. Let $c^{\star}_t$ be the true data association that indicates that observation $\mathbf{z}_t$ stems from environment

feature $\mathbf{m}_i^\star$. In contrast to that, c_t is the correspondence of observation $\mathbf{z}_t$ to a map feature $\mathbf{m}_j$ as estimated by the data association method. For every environment feature $\mathbf{m}_i^\star$, we count the number $N(\mathbf{m}_i^\star)$ of map features $\mathbf{m}_j$ that the data association method associated at least once with $\mathbf{m}_i^\star$. More formally, we define this number as

$$N(\mathbf{m}_i^\star) = \left|\{\mathbf{m}_j \in \mathbf{m}_{1:n} \mid \exists t \in [1,T] : c_t^\star(\mathbf{z}_t) = \mathbf{m}_i^\star \wedge c_t(\mathbf{z}_t) = \mathbf{m}_j\}\right|. \tag{9.4}$$

Since the features correspondences link the observations of every environment feature to at least one map feature, it holds that $N(\mathbf{m}_i^\star) \geq 1$. If the feature correspondences are correctly estimated, they link all observations of the same environment feature to the same map feature, leading to $N(\mathbf{m}_i^\star) = 1$. Accordingly, the total number of incorrectly estimated feature correspondences is given by

$$E(c_{1:T}^\star, c_{1:T}) = \sum_{i=1}^{n'} \left(N(\mathbf{m}_i^\star) - 1\right). \tag{9.5}$$

Our approach aims at placing the artificial landmarks such that the number of incorrectly estimated feature correspondences E is minimized.

Note that the value E counts all errors that occur if the data association method links at least two observations of the same environment feature to different map features. Another type of error occurs if the data association method links every single observation of two different environment features to the same map feature. Even though the number of incorrectly estimated feature correspondences E does not encode this second type of error, the experimental results in this chapter show that artificial landmarks deployed in order to optimize E result in significantly more accurate pose estimates of the robot compared to landmarks deployed with other approaches.

9.2 Reinforcement Learning for Improving Data Association

Extensive experiments (see Section 9.3) revealed that heuristics for deciding when to deploy landmarks perform badly or need to be hand-tuned for specific scenarios. Therefore, we apply actor-critic Monte Carlo reinforcement learning with a softmax policy as described in Section 3.6 to estimate a landmark deployment policy. We compute a policy that allows the robot to deploy a set of artificial landmarks at the locations that minimize the risk of wrong data associations in terms of the error E defined in Equation (9.5). To compute this error during the learning phase, we need access to the ground truth data associations $c_{1:T}^\star$. Therefore, we perform the learning phase in simulation. As our approach is designed to be decoupled from the actual navigation framework of the robot, we let the robot execute a randomly sampled navigation task in each simulated episode. The robot

thereby applies a SLAM approach and deploys its artificial landmarks according to the currently estimated policy. In each of the episodes, the robot receives the rewards

$$r(s_t, a_t) = \begin{cases} 0 & \text{if } t < T, \\ -E(c^{\star}_{1:T}, c_{1:T}) & \text{if } t = T \end{cases} \tag{9.6}$$

for executing the action a_t when it is in the state s_t. In our reinforcement learning framework, the the individual state-action pairs (s_t, a_t) are weighted with the returns $R(s_t, a_t) = \sum_{t'=t+1}^{T} r(s_{t'}, a_{t'})$, as described in Section 3.6. The rewards defined in Equation (9.6) yield the returns

$$R(s_t, a_t) = -E(c^{\star}_{1:T}, c_{1:T}) \tag{9.7}$$

for all state-action pairs (except the last) occurring during the same episode.

9.2.1 Action and State Representation

During the navigation tasks, the robot decides at every time step t according to a policy $\pi(s, a)$ whether to drop one of the artificial landmarks in the current state s. Hence, the space of the actions a in the reinforcement learning problem is given by $\mathfrak{A} = \{\textit{drop}, \textit{keep}\}$.

To learn policies that generalize well to different environments, we describe the state of the robot and the environment in terms of general state features. We use the remaining battery life time in percent, the number of artificial landmarks left on board, and the distance to the artificial landmark that has been deployed closest to the robot. This distance is estimated from the current posterior distribution (see Equation (9.3)) of the robot's SLAM system. In addition to that, we make use of a feature that captures the abstract spatial structure of the environment based on a classification of the current position of the robot in terms of the categories room, doorway, corridor, and junction. There exist several robust techniques to compute this spatial feature for robots equipped with laser scanners [33, 68] or vision systems [66, 85]. For an efficient representation of the state-action pairs in the learning procedure, we divide the space of state-action pairs into bins.

9.2.2 Statistical Convergence Test

As mentioned in Section 3.6, in actor-critic reinforcement learning an actor generates episodes following the policy π resulting from its Q-function, while a critic observes the episodes and updates its own Q-function accordingly. Critic and actor switch roles when the critic has observed enough episodes to learn the Q-function under the policy π followed by the current actor. To test whether the critic is already confident of its estimated Q-function, our approach applies a statistical convergence test after each episode.

As stated in Section 3.6, the estimated Q-function is defined as

$$\hat{Q}^{\pi}(s,a) = \frac{1}{n_{\mathcal{F}}} \sum_{e \in \mathcal{F}(s,a)} R^{e}_{\text{first}}(s,a)\,, \tag{9.8}$$

where $\mathcal{F}(s,a) = \{e \mid (s,a) \in e\}$, $n_{\mathcal{F}} = |\mathcal{F}(s,a)|$, and $R^{e}_{\text{first}}(s,a)$ is the return at the first occurrence of (s,a) in episode e. Since the policy π observed by one critic is not changed in between the episodes, the estimated Q-function is computed using independent and identically distributed (i.i.d.) samples from the same policy. Therefore, given the definition of the estimator $\hat{Q}^{\pi}(s,a)$ in Equation (9.8), its variance can be estimated as

$$S^2 = \frac{\sum_e \left(R^{e}_{\text{first}}(s,a) - \hat{Q}^{\pi}(s,a) \right)^2}{n_{\mathcal{F}} - 1}. \tag{9.9}$$

Student's t-distribution has been proven to describe the mean of a number of i.i.d. samples well in practice, even if the estimated quantity does not necessarily follow a normal distribution (see,e. g.,[75]). Under this assumption, we know that the value $Q^{\pi}(s,a)$ that we estimate lies in the interval

$$\left[\hat{Q}^{\pi}(s,a) - \sqrt{\frac{S^2}{n_{\mathcal{F}}}} t_{n_{\mathcal{F}}-1,\frac{\alpha}{2}}, \hat{Q}^{\pi}(s,a) + \sqrt{\frac{S^2}{n_{\mathcal{F}}}} t_{n_{\mathcal{F}}-1,\frac{\alpha}{2}} \right] \tag{9.10}$$

with confidence $1-\alpha$. Here, $t_{n_{\mathcal{F}}-1,\frac{\alpha}{2}}$ is the $(1-\frac{\alpha}{2})$-quantile of Student's t-distribution with $n_{\mathcal{F}}-1$ degrees of freedom. Once the confidence interval of the critic's estimate indicates convergence for all observed state-action pairs (s,a), the critic becomes actor, and a new critic is initialized.

9.3 Experimental Evaluation

We evaluated the performance of our approach both in simulation and on a real robot. In the experiments, we considered a robot that is equipped with a device for deploying five artificial landmarks, a noisy odometer, and a noisy landmark detection sensor. The landmark detection sensor receives noisy readings of the relative positions of the (completely indistinguishable) environment features and of the deployed uniquely identifiable landmarks. In the simulated learning phase, we initialized the robot in each episode at a random pose and let it perform randomly sampled navigation tasks until its battery was empty. We applied a graph-based approach to SLAM using the framework proposed in [61] and a nearest neighbor filter to compute the data associations.

9.3.1 Data Association Using the Learned Policies

In the first set of simulation experiments, we evaluated the data association performance of the policies learned by our approach. In this and the following experiments, we simulated

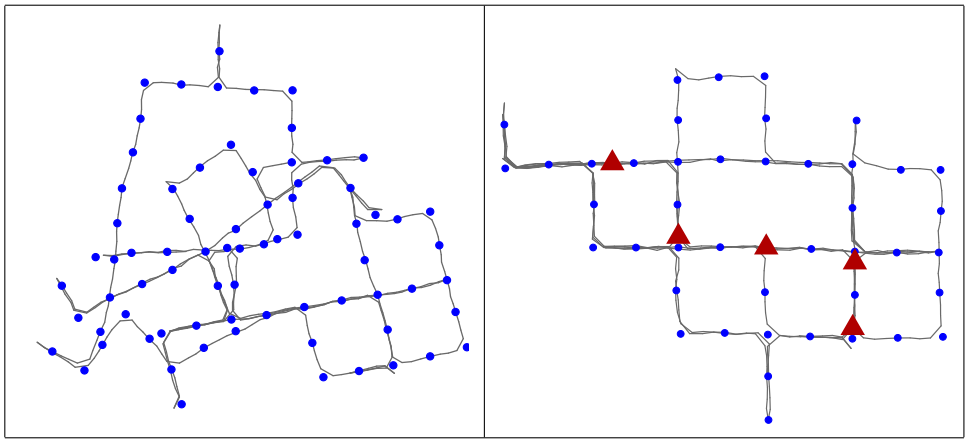

Figure 9.1: Estimated landmark positions (blue circles) and robot path (gray line) for a simulated robot traveling through a Manhattan-like world with data associations calculated without the use of deployed markers (left) and with the uniquely identifiable markers (red triangles) deployed by our approach (right).

the robot's landmark detection sensor with a circular field of view with radius $2\,\mathrm{m}$ and applied a deterministic spatial feature detection. We compared our learned policies to four naive approaches, namely *equidistant*, which deploys the artificial landmarks equidistantly in time, *random*, which deploys the markers at random time steps, *always*, which deploys landmarks at every time step until all markers are deployed, and *never*, which never deploys any landmarks, and to a heuristic in the sense of Kleiner *et al.* [50], named *density*. This heuristic computes the obstacle density to the left and to the right of the robot from a simulated laser scan by applying kernel density estimation. Likewise, it computes the density of the already deployed landmarks. Based on these densities, it decides whether to drop a landmark. We used scenario-specific hand tuned parameters to optimize the performance of this heuristic.

We evaluated every approach in 100 randomly sampled simulated runs in a 6×6-row Manhattan-like environment with 96 non-distinguishable environment features. One sample run of the simulated robot through this environment can be seen in Figure 9.1. In this scenario, the simulated robot is able to discern the spatial features "corridor" and "junction". For evaluating the quality of the different approaches, we calculated the data association error E introduced in Section 9.1.1 and the relative translational and rotational errors of the pose estimates of the SLAM approach according to the framework for evaluating SLAM described by Burgard *et al.* [18]. Figure 9.2 shows the errors for the evaluated approaches. We additionally performed two-sided t-tests, which revealed that our approach significantly outperforms all other approaches in all errors on a 95% confidence level.

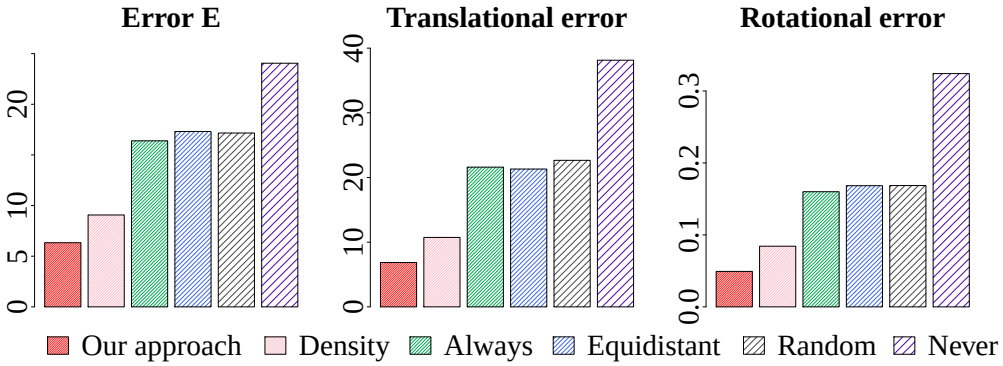

Figure 9.2: The incorrectly estimated feature correspondences E and the translational and rotational pose estimation errors (in m and rad, respectively), averaged over 100 sample runs in a Manhattan-like environment.

	Intel A	Intel B	FR079 A	FR079 B	FR106	Average
Our approach	4.30	0.45	2.62	2.39	5.29	3.01
Density	4.95	0.53	3.03	2.74	7.99	3.85
Always	6.18	1.36	5.75	4.87	10.30	5.69
Equidistant	7.09	1.80	7.12	5.84	10.59	6.49
Random	6.65	4.93	7.68	7.34	13.07	7.93
Never	14.32	10.24	15.58	14.64	26.67	16.29

Table 9.1: Values of the error E in the cross-validation in five environments

9.3.2 Generalization to New Environments

We performed a five-fold cross validation in simulation to evaluate how well the policies computed by our approach generalize to environments that the robot has not seen previously. To do so, we considered five environments: FR079 A and FR079 B, depicted in Figure 9.5, and Intel A, Intel B, and FR 106, depicted in Figure 9.3. In the simulation, we placed environment features at randomly sampled locations in the maps. In each fold of the cross validation, we learned a policy in four of the five environments and then evaluated its performance on the excluded one.

The resulting values of E are given in the first row of Table 9.1. In the other rows of the table, the E values for the heuristics described in Section 9.3.1 are stated for comparison. As can be seen in the table, the policies learned by our approach yield the lowest error values on average and for every single environment. This suggests that the policies computed by our approach generalize well to new environments.

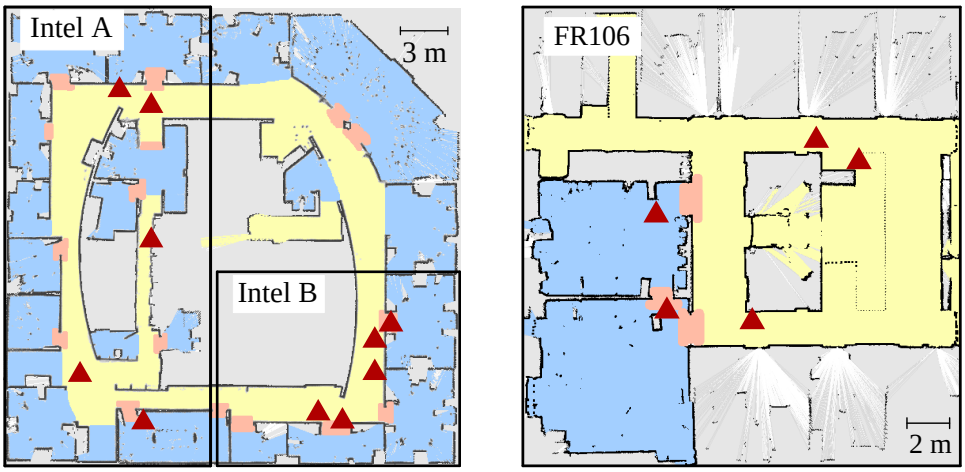

Figure 9.3: Three of the environments used for cross-validation. Intel Research Lab (left) and FR106 dataset (right). The deployed landmarks (red triangles) in the plots correspond to sample executions of the learned policies. The environments are labeled with spatial features used for learning: corridors (yellow), doorways (orange) and rooms (blue).

9.3.3 Adaptation to the Sensor Range

In this section, we evaluate how the policies computed by our learning approach adapt to the range of the landmark detector. We learned landmark deployment policies for three simulated robots with the sensor ranges $2\,\mathrm{m}$, $1\,\mathrm{m}$, and $0.5\,\mathrm{m}$ in the FR106 environment also used in the previous experiments. Figure 9.4 presents intensity plots of the resulting policies. As can be seen in the figure, the robot with sensor range $0.5\,\mathrm{m}$ strongly prefers deploying landmarks in doorways, probably because landmarks deployed in narrow passages are more likely to be observed later on, even with the small sensor range. The figure also shows that with an increasing sensor range, the decision to deploy a landmark is stronger influenced by the distance to the nearest landmark and less by the spatial feature.

9.3.4 Experiments with a Real Robot

To evaluate the performance of our approach in practice, we applied a policy learned by our approach on the robot depicted in Figure 9.5, executing randomly sampled paths in the environment shown on the right hand side of the figure. The learning phase was done in simulation and required $2,800$ episodes to converge, which took 37.48 minutes in our multi-threaded implementation on an Intel® Core™ i7 2.8GHz.

The robot is equipped with a SICK RFI641 RFID reader with a circular field of view with radius $0.9\,\mathrm{m}$ mounted at the front and a custom made device for dropping RFID tags mounted in the back. To model the RFID sensor readings, we used a two-

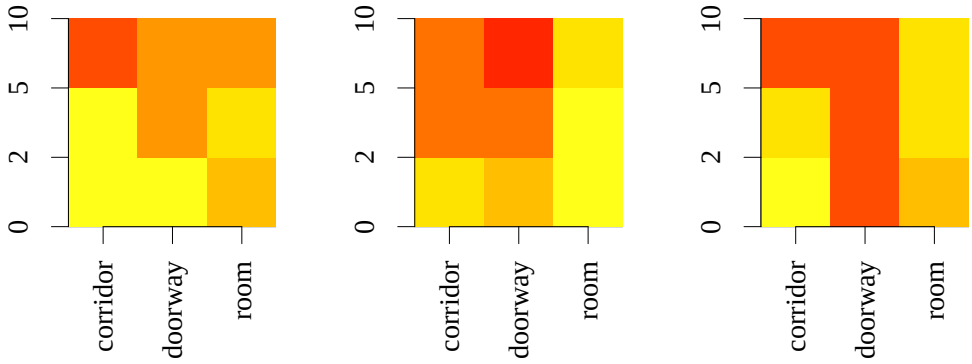

Figure 9.4: The policies learned by our approach when using sensor ranges of $2\,\mathrm{m}$ (left), $1\,\mathrm{m}$ (middle), and $0.5\,\mathrm{m}$ (right), where red corresponds to $p(\mathit{drop} \mid s) = 1$ and white corresponds to $p(\mathit{drop} \mid s) = 0$. The ordinate is the distance to the nearest deployed landmark and the abscissa is the spatial feature. The values are averaged over the battery level and the number of remaining landmarks. The probability in the lower right corner cell is not converged due to the seldom occurrences of this situation.

	Error E	Translational error	Rotational error
Our approach	19.30	$0.88\,\mathrm{m}$	$0.09\,\mathrm{rad}$
Never	23.10	$3.68\,\mathrm{m}$	$0.24\,\mathrm{rad}$

Table 9.2: Quantitative results of the real-world experiments

dimensional Gaussian distribution that we cut off at $0.9\,\mathrm{m}$ and whose covariance we calibrated according to the actual measurements of the device. Additionally, the robot is equipped with a SICK S300 laser range finder with a field of view of $270°$, which we used for computing the spatial features. To do so, we applied a straightforward heuristic: it considers local minima in the scans for extracting door posts and long parallel lines for finding corridor walls. Note that applying a more sophisticated classification technique [33, 68] or a more sophisticated sensor model [43] would possibly even further improve our results. In the experiments, we used 70 RFID tags placed at randomly selected positions as environment features. The uniquely identifiable IDs of these tags, which the robot's SLAM system did not use, make it possible to precisely evaluate the data association error E. Furthermore, we evaluated the same relative translational and rotational errors of the SLAM system that we considered in the first set of simulation experiments.

Table 9.2 shows the results averaged over ten runs of the robot. It compares the performance of the estimation of the SLAM graph considering the deployed landmarks against considering only the environment features. As can be seen in the table, the moderate reduction of the error E results in a large improvement of the pose estimation errors. The results of this experiment show that our approach is applicable in practice and

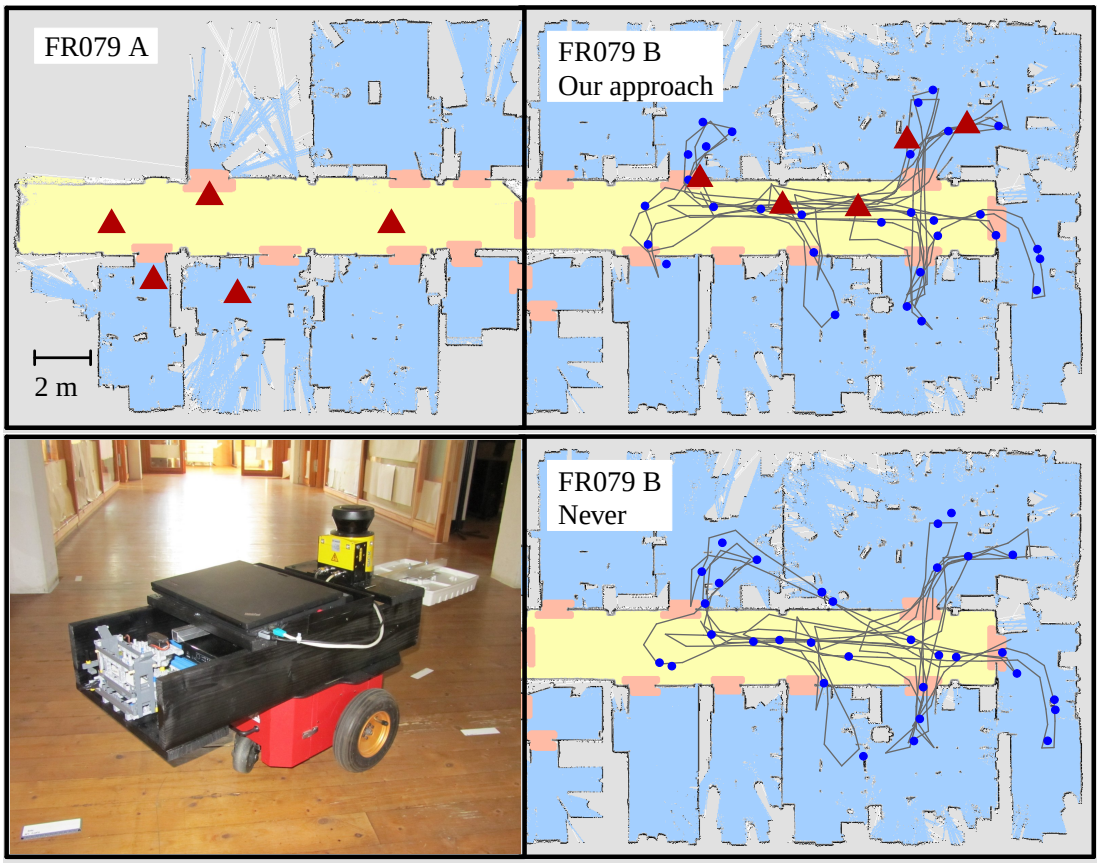

Figure 9.5: The environment used for the experiments with the real robot and for cross-validation. Upper left: Deployed landmarks (red triangles) after one sample execution of our policy in the cross-validation experiment. Upper right: One of the runs of the real robot. Depicted are the estimated path (gray lines), the estimated positions of the environment features (blue dots), and the estimated positions of the deployed landmarks (red triangles). Lower right: Estimation for the same run without integrating the observations of the deployed landmarks. The environment is labeled with the spatial features used for learning: corridors (yellow), doorways (orange) and rooms (blue). The picture shows the Pioneer P3-DX robot used in the experiments.

that for the very noisy distance readings of the RFID sensor, the landmarks deployed by our approach especially help reducing the pose estimation errors.

9.4 Discussion

In this chapter, we presented an approach to learn a policy that allows a mobile robot to effectively deploy uniquely identifiable artificial landmarks in order to minimize data association errors in SLAM. Our approach is based on actor-critic Monte Carlo reinforcement learning and uses features that support transferring the learned policies to environments not seen before. The method is designed so that it can be seamlessly integrated into any robotic system without interfering with the actual navigation task of the robot. Extensive experiments, both in simulation and on a real robot, demonstrate that our deployment approach results in significantly more accurate pose estimates than those obtained with different heuristics.

Chapter 10

Discussion

Mobile robots that can observe uniquely identifiable landmarks in their environment can use these observations, depending on their tasks, to improve their localization estimate, their navigation behavior, or their performance when building a map of the environment.

In this thesis, we introduced several novel approaches for placing minimum sets of uniquely identifiable artificial landmarks in the environment of a mobile robot in order to achieve a certain quality in the task execution of the robot.

We introduced a landmark placement approach for optimizing the expected localization performance of a mobile robot that repeatedly travels along the same pre-defined trajectory. In this context, we formulated landmark placement as the optimization problem of maximizing the expected mutual information between the states of the robot and the observations of the landmarks. We proved that this problem is NP-hard. Using the concept of submodularity, we derived a tight constant-factor bound on the error of our polynomial time approximation algorithm. For estimating the required mutual information values, our algorithm uses Monte Carlo simulations, which can deal with arbitrary system dynamics and control modes of the robot.

Furthermore, we presented a novel efficient method for estimating the expected navigation performance for a specific type of robotic system. This method is significantly faster than the Monte Carlo simulation applied in the above-mentioned approach. Our approach assumes that the robot selects its control commands with an LQR controller. It linearizes the model of the whole navigation cycle, including control, motion, and observation, which makes it possible to recursively calculate the expected distributions of the robot's deviation from its desired trajectory. Our method exploits the structure of the stochastic dependencies in the navigation framework, which allows us to reduce the dimensionality of the matrix multiplications in the calculation of the expected distributions by half, compared to the state of the art.

We applied this efficient calculation scheme in a second approach to trajectory-dependent landmark placement. Consequently, this approach also assumes that the robot is steered by an LQR controller. Instead of optimizing the localization performance of the robot, it directly optimizes the robot's navigation performance. Concretely, this

method aims at placing a minimum configuration of landmarks that guarantees a bound on the maximum deviation of the robot from its planned trajectory with high confidence. It uses an incremental landmark placement algorithm that maximizes the distance between two consecutively observed landmarks. The guaranteed bound on the maximum deviation of the robot that comes with the landmark sets placed by this approach is of high practical importance, for example, for the safety approval of mobile robots. Furthermore, the efficiency resulting from the efficient recursive calculation of the expected navigation performance makes it possible to compute landmark placements even for large-scale scenarios in reasonable time.

Some types of landmarks, like colored marks painted on walls, can wear out over time, and in many practical applications, the robot needs to share its environment with other vehicles that can block its line of sight to a placed landmark. For robustly handling such situations, we introduced another novel method for landmark placement, which builds on and extends the above-mentioned approach. This method evaluates the quality of a landmark position independently of the observability of all other placed landmarks by conservatively approximating the landmark observability. This allows us to apply an optimization procedure that explicitly takes into account that a given number of the placed landmarks can be missing. In the construction of this efficient optimization procedure, we use techniques from submodular function optimization.

In contrast to the above-described methods, which aim at preparing a known environment with artificial landmarks, landmark placement for robots traveling through unknown and unmapped environments requires different approaches. In this situation, we consider a mobile robot that is equipped with a device to deploy artificial landmarks itself. We presented an approach to learning a policy that allows the robot to effectively deploy a limited number of uniquely identifiable artificial landmarks in order to minimize data association errors in SLAM. Our approach is based on actor-critic Monte Carlo reinforcement learning and uses features that support transferring the learned policies to previously not observed environments. The method is designed so that it can be seamlessly integrated into any robotic system without interfering with the actual navigation task of the robot.

In summary, the key contributions of this thesis are approaches to placing artificial landmarks along the desired trajectory of a mobile robot and an approach to learning a policy for deciding when to autonomously deploy artificial landmarks with a mobile robot. In the landmark placement approaches, we presented solutions for handling the combinatorial structure of the placement problems, for efficiently evaluating the expected probability distributions of the states of the robot, and for robustly dealing with unpredictably missing landmarks. All our approaches are customizable to different types of robots and explicitly take into account the specifications of the applied robots including probabilistic models of the errors occurring in the motion execution and the sensor observations of the robots. As a result, our landmark placements and landmark

deployment policies are especially suited for the considered specific robots and are robust to the noise and errors occurring during operation of the robots. We evaluated all presented methods in extensive experiments, both in simulation and with real robots. The experimental results demonstrate that our approaches outperform baseline methods and that they work well on real robots. We believe that the presented landmark placement methods are a useful tool for guaranteeing a safe and reliable operation of mobile robots in practice, especially in industrial settings.

Future Work

Even though the approaches presented in this thesis cover the topic of landmark placement from several perspectives and result in a demonstrated reliable operation of mobile robots, there are still extensions that could be addressed in future research.

For example, our approaches for landmark placement optimize the configuration of landmarks along a given desired trajectory. However, in some applications, the trajectories that the robot shall execute might change often or they might even be unknown at the time of landmark placement. For these applications, it would be beneficial to have a landmark placement method that guarantees a reliable localization of the robot at any arbitrary position in the environment. Since the localization quality of the robot at a certain moment depends not only on the landmarks that the robot observes in this moment, but also on the landmarks that the robot has observed earlier, it would be necessary to make approximations or assumptions in order to extend our methods to such an application.

In the autonomous landmark deployment method, one could use the spatial features that the robot observes not only during the reinforcement learning process but also directly for data association. We believe that this is quite helpful as observations made at positions with the same spatial feature are more likely to correspond to the same landmark than observations made at positions with different spatial features.

Our methods to landmark placement are also applicable to the related problem of placing a sensor network in the environment of a vehicle for tracking it. For example, the approach for improving the localization performance of a mobile robot could also be applied to the problem of low cost surveillance of dangerous street crossings. In this scenario, our method could be adjusted to select a minimum number of positions in the vicinity of the street crossing at which it would be necessary to install a sensor in order to accurately track vehicles passing the street crossing on all trajectories that are allowed by traffic regulations.

Appendix

On the Properties of Covariance Reduction

In this appendix, we present a theoretical evaluation of covariance reduction, a function that we use in our robust landmark placement approach presented in Chapter 7 to describe the quality of landmarks. Concretely, we consider the function

$$F_t(\mathcal{A}) = tr(S_t^{\varnothing}) - tr(S_t^{\mathcal{A}})\,, \tag{A.1}$$

which describes the reduction of the trace of the covariance of the expected distribution of the robot's deviation from its desired trajectory resulting from observing the landmark set $\mathcal{A}$. Here, $S_t^{\mathcal{A}}$ is the covariance at time t in the linearized system considered in Chapter 7 assuming that the robot can observe the landmarks in $\mathcal{A}$ and $S_t^{\varnothing}$ is the same covariance, but for making no landmark observations at all.

In Section 7.3, we show that our landmark placement algorithm is guaranteed to yield near-optimal solutions if F_t is submodular, monotonically increasing, and $F_t(\varnothing) = 0$. However, the experiments presented in that section produce counter-examples disproving the submodularity of F_t. In our experiments, F_t violated the submodularity property only in a few examples and then only slightly. Therefore, it seems reasonable that F_t could be close to submodular in terms of the so-called submodularity ratio introduced by Das and Kempe [22].

Like us, Das and Kempe [21, 22] investigated covariance reduction in linear systems. In [21], they showed that for a special kind of linear system, covariance reduction is submodular. In [22], they considered covariance reduction for a broader class of linear systems. They introduced the submodularity ratio, which describes how close to submodular the behavior of a set function is. Furthermore, they relaxed the known near-optimality bounds for submodular functions to receive weaker near-optimality bounds for functions that are only close to submodular in terms of the submodularity ratio.

However, the proof in [22] of the weaker near-optimality bounds for functions with a good submodularity ratio essentially relies on the monotonicity of the considered function, and our objective function F_t is in general not monotonically increasing. Even though

this means that the landmark placement algorithm from Chapter 7 lacks a theoretical guarantee for the quality of its approximation, the experiments in Sections 7.4 and 8.2 show that it works well in practice and outperforms baseline algorithms. In the following, we prove that F_t is in general not monotonically increasing and that there are no reasonable restrictions of the landmark placement problem that would lead to a guaranteed monotonicity of F_t. Since the following proofs apply several results from Chapter 5 concerning the linearized system, we refer the reader to that chapter for the introduction of the general notation that we use here.

Let $\mathcal{A}_{t-1}$ be a set of landmark locations observable only from poses that the robot reaches at time $t-1$ or later. Throughout the rest of this appendix, we only consider observations of this landmark set and use the shorthand notations

$$S_t := S_t^{\mathcal{A}_{t-1}} \quad \text{and} \quad \overline{S}_t := \overline{S}_t^{\mathcal{A}_{t-1}} \tag{A.2}$$

for the covariance and the predicted covariance of the expected distribution resulting from placing landmarks in $\mathcal{A}_{t-1}$. Also for the other Matrices Σ_t, M_t, H_t, and K_t that depend on the set of considered landmarks, we always implicitly consider the landmark set $\mathcal{A}_{t-1}$. Only when referring to the covariance matrices resulting from making no landmark observations at all, we use the notations $S_t^{\varnothing}$, $\Sigma_t^{\varnothing}$, and $M_t^{\varnothing}$.

Lemma A.1. *For the landmark set $\mathcal{A}_{t-1}$, the values of the objective function from Equation (A.1) for the time steps $t-1$ and t are*

$$F_{t-1}(\mathcal{A}_{t-1}) = 0 \tag{A.3}$$

and

$$\begin{aligned} &F_t(\mathcal{A}_{t-1}) \\ &= tr\left(A_t K_{t-1} H_{t-1} \overline{\Sigma}_{t-1} A_t^T - (A_t + B_t L_{t-1}) K_{t-1} H_{t-1} \overline{\Sigma}_{t-1} (A_t^T + L_{t-1}^T B_t^T)\right). \end{aligned} \tag{A.4}$$

Equation (A.3) states that the landmark observations at time $t-1$ do not reduce the trace of the expected covariance S_{t-1} of the deviation of the robot at that time step. As can be seen from the following proof, this is due to the fact that the reduction of the posterior covariance Σ_{t-1} of $\mathbf{x}_{t-1}$ resulting from the landmark observations is canceled out by the increase of the expected covariance M_{t-1} of $\boldsymbol{\mu}_{t-1}$. This is an intuitive result, as making an observation reduces the localization uncertainty of the robot, but also makes the localization estimate $\boldsymbol{\mu}_{t-1}$ jump. This jump results in an increased uncertainty of the expected distribution of the localization estimate $\boldsymbol{\mu}_{t-1}$, which itself can be considered as a random variable. In the next time step, however, the robot executes a control command based on the refined localization estimate, which effects S_t as shown in Equation (A.4).

Proof of Lemma A.1. Since $t-1$ is by definition the first time step at which landmarks from $\mathcal{A}_{t-1}$ are observable, it holds that

$$S_{t-1}^{\varnothing} = \overline{S}_{t-1}\,. \tag{A.5}$$

Theorem 5.1 from Chapter 5 states that $S_t = \Sigma_t + M_t$. Applying this theorem and Equation (A.5) on the definition of F_{t-1} and then replacing Σ_{t-1} and M_{t-1} according to their recursive update rules stated in Equations (5.18) to (5.22) and Lemma 5.3, respectively, yields

$$\begin{aligned}
&F_{t-1}(\mathcal{A}_{t-1}) && \text{(A.6)}\\
&= tr\left(\overline{\Sigma}_{t-1} + \overline{M}_{t-1} - (\Sigma_{t-1} + M_{t-1})\right) && \text{(A.7)}\\
&= tr\left(\overline{\Sigma}_{t-1} + \overline{M}_{t-1} - (\overline{\Sigma}_{t-1} - K_{t-1}H_{t-1}\overline{\Sigma}_{t-1} + \overline{M}_{t-1} + K_{t-1}H_{t-1}\overline{\Sigma}_{t-1})\right) && \text{(A.8)}\\
&= 0\,, && \text{(A.9)}
\end{aligned}$$

which proves Equation (A.3).

For proving Equation (A.4), we consider the two parts S_t and $S_t^{\varnothing}$ of $F_t(\mathcal{A}_{t-1})$ individually. With the same argument as the one used in Equation (A.5), we know that $S_t^{\varnothing}$ equals a version of $\overline{S}_{t-1}$ that is once updated according to the motion model of the robot, leaving out the update steps for the observations at times $t-1$ and t. Using Theorem 5.1 and the same recursive update rules for $\Sigma_t^{\varnothing}$ and $M_t^{\varnothing}$ as in the above equations leads to

$$\begin{aligned}
S_t^{\varnothing} &= \Sigma_t^{\varnothing} + M_t^{\varnothing} && \text{(A.10)}\\
&= \overline{\Sigma}_t^{\varnothing} + \overline{M}_t^{\varnothing} && \text{(A.11)}\\
&= A_t\Sigma_{t-1}^{\varnothing}A_t^T + V_tQ_tV_t^T + (A_t + B_tL_{t-1})M_{t-1}^{\varnothing}(A_t^T + L_{t-1}^TB_t^T) && \text{(A.12)}\\
&= A_t\overline{\Sigma}_{t-1}A_t^T + V_tQ_tV_t^T + (A_t + B_tL_{t-1})\overline{M}_{t-1}(A_t^T + L_{t-1}^TB_t^T)\,. && \text{(A.13)}
\end{aligned}$$

Next, we express S_t in terms of the covariance matrices at time $t-1$, using the same update rules as above, but this time also with the observation updates for the time steps $t-1$ and t. As the steps of this derivation are similar to the equations above, we just state the final result:

$$\begin{aligned}
S_t &= A_t(\overline{\Sigma}_{t-1} - K_{t-1}H_{t-1}\overline{\Sigma}_{t-1})A_t^T + V_tQ_tV_t^T && \text{(A.14)}\\
&\quad + (A_t + B_tL_{t-1})(\overline{M}_{t-1} + K_{t-1}H_{t-1}\overline{\Sigma}_{t-1})(A_t^T + L_{t-1}^TB_t^T)\,. && \text{(A.15)}
\end{aligned}$$

Plugging Equations (A.13) and (A.15) into the definition of F_t from Equation (A.1) finishes the proof:

$$\begin{aligned}
&F_t(\mathcal{A}_{t-1}) && \text{(A.16)}\\
&= tr\left(A_t\overline{\Sigma}_{t-1}A_t^T + V_tQ_tV_t^T + (A_t + B_tL_{t-1})\overline{M}_{t-1}(A_t^T + L_{t-1}^TB_t^T)\right) && \text{(A.17)}\\
&\quad - tr\left(A_t(\overline{\Sigma}_{t-1} - K_{t-1}H_{t-1}\overline{\Sigma}_{t-1})A_t^T + V_tQ_tV_t^T\right. && \text{(A.18)}\\
&\quad \left. + (A_t + B_tL_{t-1})(\overline{M}_{t-1} + K_{t-1}H_{t-1}\overline{\Sigma}_{t-1})(A_t^T + L_{t-1}^TB_t^T)\right) && \text{(A.19)}\\
&= tr\left(A_tK_{t-1}H_{t-1}\overline{\Sigma}_{t-1}A_t^T - (A_t + B_tL_{t-1})K_{t-1}H_{t-1}\overline{\Sigma}_{t-1}(A_t^T + L_{t-1}^TB_t^T)\right)\,. && \text{(A.20)}
\end{aligned}$$

□

Using this lemma, we can show that F_t is in general not monotonically increasing. This makes it infeasible to adapt the proof of Theorem 3.2 in [22] to prove that optimizing F_t greedily results in near-optimal landmark sets.

Remark A.1. *In general, F_t is not monotonically increasing. There are no reasonable restrictions of the landmark placement problem that would lead to a guaranteed monotonicity of F_t.*

Derivation of Remark A.1. From the definition of F_t in Equation (A.1) we know that

$$F_t(\varnothing) = 0\,. \tag{A.21}$$

In the following, we show that in general, $F_t(\mathcal{A}_{t-1}) \not\geq 0$, and discuss the restrictions of the structure of the occurring matrices that would be necessary to guarantee that $F_t(\mathcal{A}_{t-1}) \geq 0$. Because of Equation (A.21), this is equivalent to the statement in Remark A.1.

For the simplest instance of an LQR controller with look-ahead 1 ($t' = t$ in Equation (5.2)), the feedback matrix L_{t-1} can be computed via Equation (5.26) as

$$L_{t-1} = -(B_t^T C B_t + D)^{-1} B_t^T C A_t\,, \tag{A.22}$$

where C and D are the constant weight matrices of the LQR controller. With this equation, we have

$$\begin{aligned}
&A_t + B_t L_{t-1} && \text{(A.23)}\\
&= A_t - B_t (B_t^T C B_t + D)^{-1} B_t^T C A_t && \text{(A.24)}\\
&= \big(I - \underbrace{B_t (B_t^T C B_t + D)^{-1} B_t^T C}_{=:X}\big)\, A_t && \text{(A.25)}\\
&= \big(I - X\big)\, A_t\,, && \text{(A.26)}
\end{aligned}$$

where I is the identity matrix. Plugging this into Equation A.4 yields

$$\begin{aligned}
&F_t(\mathcal{A}_{t-1}) && \text{(A.27)}\\
&= tr(\underbrace{A_t K_{t-1} H_{t-1} \overline{\Sigma}_{t-1} A_t^T}_{=:Y} - (I - X)\, \underbrace{A_t K_{t-1} H_{t-1} \overline{\Sigma}_{t-1} A_t^T}_{=:Y} (I - X^T)) && \text{(A.28)}\\
&= tr(Y - (I - X) Y (I - X^T))\,. && \text{(A.29)}
\end{aligned}$$

As discussed in the beginning of this proof, $F_t(\mathcal{A}_{t-1}) \geq 0$ would be a necessary (but not a sufficient) precondition for F_t to be monotonically increasing. In the following, we discuss the restrictions on the matrices in Equation (A.29) that would be necessary to guarantee that $F_t(\mathcal{A}_{t-1}) \geq 0$.

Depending on the initial covariance Σ_0 and the other matrices from which Y is recursively created, Y can be any symmetric positive definite matrix. In particular, Y does not need to be diagonally dominant. Therefore, the only possibility to guarantee that for all possible Y, the trace of $(I - X)Y(I - X^T)$ is smaller than the trace of Y itself would be to guarantee that X is a diagonal matrix whose diagonal elements are all greater or equal zero. Now consider the simplest case, in which C and D are diagonal matrices whose diagonal entries are all the same (which allows for them to commute with every other matrix) and B_t is an invertible square matrix. Even in this case, for X to be diagonal, it would be necessary that $B_t B_t^T$ or $B_t^T B_t$ is a diagonal matrix, which is not true for almost all reasonable motion models. The other possibility to ensure that X is diagonal would be to choose C and D differently for every time step, to cancel out the different kinds of non-diagonal B_t matrices. This is obviously also not a reasonable restriction. □

In Chapter 4, we showed that the objective function considered in that chapter is not submodular in the case of autonomous controls. Note that the argument from Chapter 4 is not valid for the objective function from Chapter 7, which we discussed in this appendix. For the objective function from Chapter 4, the submodularity property in the autonomous control case does not hold due to the dependency of the observability of a landmark on the observations of other landmarks. We give a detailed description of this fact in Section 4.3.2. In contrast to that, in the above discussion and in Chapter 7, the observability of landmarks poses no problem due to the conservative approximation presented in Section 7.2.1. Still, the reason that F_t is not generally monotonically increasing lies in the structure of the autonomous control mode. As described in the above derivation, the reason is the structure of the Jacobian B_t of the motion model that appears in the definition of the feedback matrix L_{t-1} of the autonomous LQR controller (Equation (A.22)).

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, 1964.

[2] Y. Bar-Shalom, T. Kirubarajan, and X. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2002.

[3] M. Batalin and G. Sukhatme. Efficient exploration without localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[4] M. Beinhofer and W. Burgard. Efficient estimation of expected distributions for mobile robot navigation. In *Proc. of the Austrian Robotics Workshop (ARW)*, 2014.

[5] M. Beinhofer, J. Müller, and W. Burgard. Landmark placement for accurate mobile robot navigation. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2011.

[6] M. Beinhofer, J. Müller, and W. Burgard. Near-optimal landmark selection for mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[7] M. Beinhofer, H. Kretzschmar, and W. Burgard. Deploying artificial landmarks to foster data association in simultaneous localization and mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[8] M. Beinhofer, J. Müller, and W. Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics and Autonomous Systems (RAS)*, 61(10):1060 – 1069, 2013.

[9] M. Beinhofer, J. Müller, A. Krause, and W. Burgard. Robust landmark selection for mobile robot navigation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[10] M. Bender, A. Fernãdez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176(1): 1–21, 2002.

[11] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[12] J. van den Berg, S. Patil, R. Alterovitz, P. Abbeel, and K. Goldberg. LQG-based planning, sensing, and control of steerable needles. In *Proc. of the Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2010.

[13] D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition, 2005.

[14] I. Białynicki-Birula and J. Mycielski. Uncertainty relations for information entropy in wave mechanics. *Communications in Mathematical Physics*, 44(2):129–132, 1975.

[15] M. Billinghurst and H. Kato. Collaborative augmented reality. *Communications of the ACM*, 45(7):64–70, 2002.

[16] C. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

[17] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[18] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. 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.

[19] G. Casella and R. Berger. *Statistical Inference*. Duxbury Resource Center, 2001.

[20] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

[21] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, 2008.

[22] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2011.

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

[24] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Map validation and robot self-location in a graph-like world. *Robotics and Autonomous Systems*, 22(2):159–178, 1997.

[25] H. Durrant-Whyte, D. Pagac, B. Rogers, M. Stevens, and G. Nelmes. Field and service applications – an autonomous straddle carrier for movement of shipping containers. *IEEE Robotics & Automation Magazine*, 14:14–23, 2007.

[26] W. Emara and M. Kantardzic. Efficient approximate semi-supervised support vector machines through submodular optimization. In *Proc. of the Int. Conf. on Machine Learning and Applications (ICMLA)*, 2011.

[27] A. Ercan, D. Yang, A. El Gamal, and L. Guibas. Optimal placement and selection of camera network nodes for target localization. In *Proc. of the Int. Conf. on Distributed Computing in Sensor Systems (DCSS)*, 2006.

[28] L. Erickson and S. LaValle. An art gallery approach to ensuring that landmarks are distinguishable. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[29] L. Erickson and S. LaValle. Navigation among visually connected sets of partially distinguishable landmarks. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[30] C. Estrada, J. Neira, and J. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. on Robotics*, 21(4):588–596, 2005.

[31] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[32] M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of approximations for maximizing submodular set functions – II. In *Polyhedral combinatorics*, pages 73–87. Springer Verlag, 1978.

[33] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[34] T. Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Trans. on Information & Systems*, 83(3):480–487, 2000.

[35] A. Gelb. *Applied optimal estimation*. MIT Press, 1974.

[36] G. Golub and C. van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

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

[38] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Trans. on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[39] J. Hess, M. Beinhofer, D. Kuhner, P. Ruchti, and W. Burgard. Poisson-driven dirt maps for efficient robot cleaning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[40] J. Hess, M. Beinhofer, and W. Burgard. A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

[41] G. Hollinger, B. Englot, F. Hover, U. Mitra, and G. Sukhatme. Uncertainty-driven view planning for underwater inspection. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[42] Z. Jiang, G. Zhang, and Larry S. Davis. Submodular dictionary learning for sparse coding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[43] D. Joho, C. Plagemann, and W. Burgard. Modeling RFID signal strength and tag detection for localization and mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[44] D. Jourdan and N. Roy. Optimal sensor placement for agent localization. *ACM Trans. on Sensor Networks*, 4(3):1–40, 2008.

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

[46] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, 2008.

[47] O. Kallenberg. *Foundations of modern probability*. Springer Verlag, 2002.

[48] R. Kalman. A new approach to linear filtering and prediction problems. *Trans. of the ASME - Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[49] G. Kim, E. Xing, L. Fei-Fei, and T. Kanade. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2011.

[50] A. Kleiner, J. Prediger, and B. Nebel. RFID technology-based exploration and SLAM for search and rescue. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[51] A. Kleiner, D. Sun, and D. Meyer-Delius. ARMO – adaptive road map optimization for large robot teams. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[52] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

[53] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[54] A. Krause and C. Guestrin. Optimal nonmyopic value of information in graphical models – efficient algorithms and theoretical limits. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2005.

[55] A. Krause and C. Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 35:557–591, 2009.

[56] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[57] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research (JMLR)*, 9:2761–2801, 2008.

[58] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, 2008.

[59] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin. Simultaneous placement and scheduling of sensors. In *Proc. of the ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2009.

[60] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Trans. on Sensor Networks (TOSN)*, 7(4):31:1–31:33, 2011.

[61] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[62] Qiao L., R. Negi, and M. Ilić. Phasor measurement units placement for power system state estimation: A greedy approach. In *Proc. of the IEEE Power and Energy Society General Meeting*, 2011.

[63] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation*, 7(3):376–382, 1991.

[64] J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1991.

[65] R. Lerner, E. Rivlin, and I. Shimshoni. Landmark selection for task-oriented navigation. *IEEE Trans. on Robotics*, 23(3):494–505, 2007.

[66] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[67] Ning M., M. Bouchard, and R. Goubran. Speech enhancement using a masking threshold constrained kalman filter and its heuristic implementations. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(1):19–32, 2006.

[68] O. Martinez-Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[69] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria. How the location of the range sensor affects EKF-based localization. *Journal of Intelligent & Robotic Systems*, 68(2):121–145, 2012.

[70] R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, 2004.

[71] 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 Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[72] 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 Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[73] D. Meyer-Delius, M. Beinhofer, and W. Burgard. Occupancy grid models for robot mapping in changing environments. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, 2012.

[74] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proc. of the Conf. on Autonomous Robot Systems and Competitions*, 2009.

[75] W. Navidi. *Statistics for Engineers And Scientists*. McGraw-Hill Higher Education, 2007.

[76] J. Neira and J. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. on Robotics and Automation*, 17(6):890–897, 2001.

[77] G. Nemhauser and L. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[78] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14: 265–294, 1978.

[79] E. Olson. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems (RAS)*, 57(12):1157–1172, 2009.

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

[81] R. Parker, F. Doyle, and N. Peppas. A model-based algorithm for blood glucose control in type I diabetic patients. *IEEE Trans. on Biomedical Engineering*, 46(2): 148–157, 1999.

[82] S. Patil, J. van den Berg, and R. Alterovitz. Motion planning under uncertainty in highly deformable environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[83] M. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[84] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Int. Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

[85] A. Pronobis, B. Caputo, P. Jensfelt, and H. Christensen. A discriminative approach to robust visual place recognition. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[86] F. Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2006.

[87] R. Qian, M. Sezan, and K. Matthews. A robust real-time face tracking algorithm. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP)*, 1998.

[88] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 24(6):843–854, 1979.

[89] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation: Mobile robot navigation with uncertainty in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[90] T. Rupp and P. Levi. Optimized landmark arrangement for absolute localization – a practical approach. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[91] P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson. Landmark selection for vision-based navigation. *IEEE Trans. on Robotics and Automation*, 22(2):334–349, 2006.

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

[93] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Grasping and fixturing as submodular coverage problems. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2011.

[94] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[95] S. Shmakov. A universal method of solving quartic equations. *Int. Journal of Pure and Applied Mathematics (IJPAM)*, 71(2):251–259, 2011.

[96] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

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

[98] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

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

[100] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

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

[102] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.

[103] P. Tokekar and V. Isler. Sensor placement and selection for bearing sensors with bounded uncertainty. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[104] P. Vernaza, B. Taskar, and D. Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[105] M. Vitus and C. Tomlin. Sensor placement for improved robotic navigation. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[106] M. Vitus and C. Tomlin. Closed-loop belief space planning for linear, gaussian systems. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[107] H. Wang, M. Jenkin, and P. Dymond. The relative power of immovable markers in topological mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[108] V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, 2012.

[109] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

[110] H. Wünsche. *Bewegungssteuerung durch Rechnersehen: Ein Verfahren zur Erfassung und Steuerung räumlicher Bewegungsvorgänge in Echtzeit*. Fachberichte Messen, Steuern, Regeln. Springer Verlag, 1988.

[111] P. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19, 2008.