State Estimation and Optimization for Mobile Robot Navigation

Rainer Kümmerle

Technische Fakultät
Albert-Ludwigs-Universität Freiburg im Breisgau

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

April, 2013

State Estimation and Optimization for Mobile Robot Navigation

Rainer Kümmerle

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

Dekan:	Prof. Dr. Yiannos Manoli
Erstgutachter:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	Prof. Dr.-Ing. Thomas Brox
Tag der Disputation:	08.04.2013

Abstract

Robust autonomous navigation is a key feature of a mobile robot realizing services such as transportation, cleaning, search and rescue, and surveillance. In addition to that, navigation is a building block for a robot assisting humans in potentially dangerous situations, such as search-and-rescue scenarios. Hence, navigation is one of the major research topics in the robotics community.

To realize the above mentioned applications, we need to fulfill certain requirements, so that a robot is regarded as useful. For example, a robot which performs pick-and-place tasks or offers guidance in city centers needs to be aware of its own position in the environment and it needs to have an accurate model of the environment for planning an appropriate path. A robot which should guide a human to a certain place or has to deliver goods is only regarded as helping hand, if the location is reliably reached within the expected time frame.

Particularly, estimating the state which describes the current situation of the navigation system is complex. In this thesis, we focus on efficient and accurate state estimation techniques which apply probabilistic algorithms. An example for such a state estimation task is the Simultaneous Localization and Mapping (SLAM) problem, in which a robot has to address both aspects. First, it needs to estimate what the environment looks like. This is the mapping part which deals with integrating the information obtained by the sensors of the robot into an appropriate representation. Second, the localization component has to estimate the position of the robot with respect to the model of the environment.

In the first part of this thesis, we present efficient approaches to estimate the state of the robot while performing SLAM. Our approach allows a robot to accurately estimate the model of the environment in an online setting and also in situations when provided with a poor initial guess. Additionally, we provide an empirical evaluation which demonstrates the advantages of our approach compared to other state-of-the-art methods. Subsequently, we extend our state estimation approach to also include the unknown calibration parameters, which might change during the lifetime of the robot, to incorporate prior information about the structure of the environment, and to improve the fine-grained details of the estimated models.

In the second part of this thesis, we demonstrate two challenging applications which we realized by building upon and extending the algorithms presented in the first part. In detail, we discuss an approach which allows a car to autonomously park in a complex multi-level parking garage. As second application we present a robotic pedestrian assistant which is able to navigate in densely populated pedestrian zones.

All techniques presented in this thesis have been implemented and tested using both real-world data collected with mobile robots and simulated data. To support our claims, we performed an extensive collection of experiments, in which we compared the performance of our approaches with the state-of-the-art. We believe that the proposed approaches will allow us in the future to build systems that can assist humans in their homes and at their workplaces.

Zusammenfassung

Die Fähigkeit sich in einer Umgebung zurechtzufinden ist eine der grundlegenden Vorbedingungen, um wirklich selbstständige Roboter zu realisieren. Solche autonomen Roboter können eine Reihe von sinnvollen Dienstleistungen anbieten. Als Beispiel seien hier der Transport von Waren zu einem gewünschten Ziel, die effiziente und zuverlässige Säuberung eines Gebietes oder die Überwachung eines bestimmten Bereiches genannt. Des Weiteren ist die autonome Navigation ein wichtiger Bestandteil eines Roboters, der Menschen in potentiell gefährlichen Situationen unterstützen soll, wie z.B. bei Such- und Rettungseinsätzen. Aufgrund dieser vielfältigen Möglichkeiten ist die autonome Navigation eines der zentralen Forschungsgebiete im Bereich der Robotik.

Um die angesprochenen Anwendungen zu realisieren, müssen wir jedoch eine Menge von Bedingungen erfüllen, so dass der Roboter als nützlich erachtet wird. So muss beispielsweise ein Roboter, der Hol- und Bringdienste oder Hilfe zur Orientierung in einer Stadt anbietet, seine eigene Position möglichst präzise kennen. Des Weiteren benötigt der Roboter eine genaue Karte der Umgebung, um einen geeigneten Pfad zum Ziel zu planen. Ein Roboter, der eine Person zu einem gewünschten Ziel führt oder Waren liefert, wird nur dann als nützlich erachtet, wenn das Ziel zuverlässig innerhalb der erwarteten Zeitspanne erreicht wird.

Wenn ein Roboter in einem unbekannten und unvorhersehbaren Gebiet navigieren soll, muss er ein schwieriges Problem lösen. Die Schwierigkeit bei der Schätzung des wahrscheinlichsten Zustands ergibt sich hierbei aus der Tatsache, dass der Roboter sich in der Umgebung lokalisieren muss, während er gleichzeitig ein Modell der Umgebung schätzt. Die simultane Lokalisierung und Kartierung ist in der Robotik als SLAM-Problem bekannt. Lösungsansätze für SLAM müssen beide Aspekte behandeln. Der erste Teil befaßt sich mit der Kartierung der Umgebung. Anhand der Sensormessungen des Roboters wird ein geeignetes Modell der Umgebung erstellt. Dieses Modell wiederum wird von der zweiten Komponente, der Lokalisierung, verwendet, um die Position des Roboters zu ermitteln. Jeder der beiden Teile kann, wenn das entsprechende Gegenstück gegeben ist, auf unkomplizierte Art und Weise gelöst werden. Die Schwierigkeit ensteht durch die Notwendigkeit, beide Teile gleichzeitig zu lösen. In diesem Fall beeinflusst das Wissen über die Position des Roboters das Resultat der Kartierung, wohingegen das Modell der Umgebung sich auf die Schätzung der Position auswirkt. Dementsprechend handelt es sich bei SLAM um eine komplexe Zustandschätzung, insbesondere da der Zustand hochdimensional ist.

Eine mögliche Repräsentation für das SLAM-Problem ist ein Graph bestehend aus Knoten und Kanten. Die Positionen des Roboter werden hierbei durch die Knoten modelliert und die Kanten des Graph bilden die Beziehung der Knoten zueinander gemäß der Sensormessungen ab. Die Lösung besteht darin die Konfiguration der Knoten zu finden, die die Wahrscheinlichkeit aller Messungen maximiert.

Während der Roboter seine Umgebung exploriert besteht die Notwendigkeit, das zum aktuellen Zeitpunkt wahrscheinlichste Modell der Umgebung und die Position des Roboters zu bestimmen. Der Roboter benötigt diese Information unter anderem, um seine nächste Aktion

zu planen. Daher sollte unser Verfahren in der Lage sein, eine solche Zwischenlösung schnell zu berechnen. Im Fall von langwierigen Berechnungen müsste der Roboter gegebenenfalls häufig auf das Ergebnis warten. Des Weiteren kann der aktuell wahrscheinlichste Zustand dem Roboter bei der Interpretation der Sensordaten helfen und somit unterstützend zur Lösung der komplexen Datenassoziation beitragen.

Außerdem benötigt der Roboter Wissen über Parameter, um die Daten, die von den Sensoren geliefert werden, zu verarbeiten. Ein Sensor misst im Allgemeinen eine physikalische Größe, wie Spannungsunterschiede, Phasenverschiebung, oder die Laufzeit eines Signals. Um diese Messungen in die Zustandsschätzung einfließen zu lassen, wird üblicherweise eine Umrechnung in abstraktere Werte durchgeführt. So wird zum Beispiel die Verschiebung der Phase in die Distanz zum Objekt, das den Lichtstrahl reflektiert hat, transformiert. Diese Operation benötigt jedoch Wissen über eine Menge von Parametern, die zum einen konstant sein können (wie die Wellenlänge des Lichtstrahls) oder zum anderen sich im Laufe der Zeit ändern. Für die Berechnung der Odometrie des Roboters etwa muss neben anderen Parametern der Durchmesser der Räder bekannt sein. Einerseits wirkt sich die Abnahme des Reifendrucks in aufgepumpten Reifen auf deren Größe aus und andererseits beeinflusst das Gesamtgewicht des Roboters den Durchmesser der Räder, so dass sich dieser im Laufe der Zeit verändern kann. Die Mehrheit der Ansätze zur Lösung von SLAM, die dem Stand der Technik entsprechen, nehmen die Parameter als gegeben und als konstant an. Eine nicht modellierte Veränderung der Parameter kann zu einem systematischen Fehler innerhalb der Zustandsschätzung führen, was sich letztendlich in einer verminderten Qualität der Umgebungsmodelle äußern kann.

Während die oben angesprochene zeitlich schritthaltende Kalibrierung zu einer Verbesserung der Qualität führen kann, können wir durch weitere Messungen ebenfalls die Genauigkeit der Umgebungsmodelle steigern. Im Internet steht zum heutigen Zeitpunkt eine Fülle an präzisem Kartenmaterial frei zur Verfügung. Wäre der Roboter in der Lage dieses Material für die Lokalisierung zu nutzen, so könnte dies zu einer verbesserten Schätzung des Zustands führen. Die meisten bisher vorgeschlagenen Ansätze für SLAM sind jedoch nicht in der Lage, dieses Vorwissen miteinzubeziehen. Bei der Einbindung des bestehenden Kartenmaterials müssen wir jedoch beachten, dass das Vorwissen eventuell nicht alle Bereiche der Umgebung abdeckt.

Wie wir oben angedeutet haben besteht ein Ansatz für SLAM aus zwei Teilen, der Lokalisierung und der Kartierung. Eine mögliche Variante zur Kartierung mithilfe von Abstandssensoren ist ein sogenanntes Belegungsgitter. In diesem Gitter wird für jede Zelle unabhängig die Wahrscheinlichkeit belegt zu sein mittels eines bayesschen Filter geschätzt. Andere Arten das Modell der Umgebung zu schätzen – wie Punktwolken – sind ebenfalls geeignet, falls sie sich zur Lokalisierung des Roboters anwenden lassen. Bei der Erstellung des Modells muss der Roboter die verbleibende Unsicherheit in der geschätzten Position und den Abstandsmessungen beachten. Das Verfahren zur Berechnung eines Belegungsgitter nimmt die Positionen des Roboters als bekannt an und modelliert lediglich das Rauschen in den Abstandsmessungen. Wenn unser Verfahren zur Berechnung eines Umgebungsmodells in der Lage wäre auch die verbleibende Unsicherheit in der Position des Roboters zu beachten, könnten wir die Genauigkeit der Kartierung steigern.

Die von uns in der vorliegenden Arbeit vorgestellten Verfahren zur Zustandsschätzung verfügen über ein breites Anwendungsspektrum. So ermöglicht ein präzises Umgebungsmodell unter anderem die Berechnung eines Pfads zwischen einer gegebenen Startposition und einem gewünschten Ziel. Des Weiteren bieten Autos heutzutage eine Reihe von Assistenzsystemen (z.B. Spurhalteunterstützung und Notbremssysteme) an, die sich als Teilbereiche der autonomen Navigation einordnen lassen. Auch das komplett selbstständige Einparken unter gewissen Vorbedingungen, wie die räumliche Nähe zur Parklücke, ist mit modernen Fahrzeugen möglich.

Hinzu kommen aktuelle Forschungsfahrzeuge, die in der Lage sind am normalen Straßenverkehr teilzunehmen. Diese Fahrzeuge benötigen hierfür jedoch üblicherweise GPS. Die in der vorliegenden Arbeit vorgestellten Verfahren erlauben uns, diese strikten Annahmen abzuschwächen. Durch die Realisierung von sicherem Fahren und mehr Komfort in lästigen Situationen, wie der Suche nach einem Parkplatz in einem engen Parkhaus, ergibt sich ein Markt für die von uns im Bereich der Robotik entwickelten Technologien.

Zusammenfassend werden wir in dieser Arbeit die folgenden Fragestellungen untersuchen:

- Wie kann der Roboter den Zustand, der seine Position und die Karte beinhaltet, effizient und robust schätzen?
- Wie können die Parameter, die zur korrekten Interpretation der Sensordaten benötigt werden, geschätzt werden?
- Wie kann bestehendes Kartenmaterial ausgenutzt werden, um die Qualität der von Robotern erstellten Karten zu erhöhen?
- Wie kann ein präzises Modell der Umgebung bestimmt werden, das die Unsicherheit in den Messdaten und in der Position des Roboters betrachtet?
- Wie können wir Roboter realisieren, die selbstständig in komplexen urbanen Umgebungen navigieren können.

Die vorliegende Arbeit ist wie folgt strukturiert. Im ersten Kapitel wird zunächst das Thema dieser Arbeit eingeführt und die wissenschaftlichen Fragestellung herausgearbeitet. Es folgt der erste Teil dieser Arbeit bestehend aus den Kapiteln 2 bis 7, in dem wir zuerst das probabilistische Modell für die selbstständige Kartierung einer Umgebung durch einen mobilen Roboter erläutern. Dies führt uns zu einem Optimierungsproblem, für das wir eine effiziente allgemeine Lösungsmethode vorstellen.

Aufbauend auf dieser Methode entwickeln wir in Kapitel 3 einen neuartigen Ansatz, der es dem Roboter ermöglicht zeitlich schritthaltend und ausgehend von einem schlechten initialen Zustand robust und effizient eine Lösung zu bestimmen. Dieses Verfahren wendet das bekannte Prinzip „Teile und Herrsche“ an, indem es das gesamte Optimierungsproblem in kleinere Teilprobleme zerlegt, die unabhängig voneinander gelöst werden und dann zu einer Lösung für das ursprüngliche Problem kombiniert werden.

Nach der Vorstellung unserer Verfahren für die Erstellung eines Umgebungsmodells mit mobilen Robotern betrachten wir in Kapitel 4 eine Metrik, mit deren Hilfe wir in der Lage sind, unsere Ergebnisse objektiv mit dem Stand der Technik zu vergleichen. Die Evaluation auf Grundlage einer Menge von häufig genutzten Datensätzen zeigt die hohe Genauigkeit unserer Methode im Vergleich zum Stand der Technik.

Anschließend stellen wir in Kapitel 5 einen neuen Ansatz vor, der in der Lage ist, die Kalibrierung von Parametern durchzuführen, während der Roboter gleichzeitig ein Umgebungsmodell lernt. Hierbei beinhaltet die Menge der Parameter unter anderem die Parameter der Kinematik des Roboters. Diese können sich im Laufe der Zeit ändern, werden jedoch von den meisten Ansätzen als bekannte Konstanten angenommen. In Situationen, in denen der Roboter Güter transportiert, ist der Durchmesser der Räder abhängig vom Gesamtgewicht der Plattform. Mit unserem Ansatz können wir diese Veränderungen präzise und effizient berücksichtigen.

Von unserem Ansatz zur Optimierung ausgehend entwickeln wir in Kapitel 6 eine neue Methode, die es erlaubt Informationen über die Struktur einer Umgebung, die durch bestehendes Kartenmaterial gegeben ist, in die Schätzung des Roboters einfließen zu lassen. Unser Ansatz

verwendet hierzu Luftaufnahmen, anhand derer der Roboter seine Position in der Welt mit Hilfe seiner drei-dimensionalen Sensormessungen bestimmt. Dies führt zu einer gesteigerten Genauigkeit der Karten.

Des Weiteren befassen wir uns in Kapitel 7 mit einem neuen Ansatz die Qualität der geschätzten Umgebungsmodelle zu verbessern. In diesem Fall modellieren wir ausdrücklich die Messfehler in den Messungen eines Laser-Abstandssensors. Unser Modell ist durch die physikalischen Prinzipien des Sensors motiviert. Zu diesem Zweck erstellen wir ein gemeinsames Optimierungsproblem, das sowohl die Abstandsmessungen als auch die Positionen des Roboters beinhaltet. Im Vergleich zum Stand der Technik kann unser Ansatz die verbleibende Unsicherheit des geschätzten Modells reduzieren.

Im zweiten Teil dieser Arbeit, bestehend aus den Kapiteln 8 und 9, stellen wir zwei mögliche Applikationen für die von uns entwickelten Ansätze vor. Zunächst diskutieren wir in Kapitel 8 einen Ansatz, der es einem autonomen Fahrzeug ermöglicht in einem mehrstöckigen Parkhaus zu navigieren und schließlich zu parken. Dieses System hebt die Vorzüge unserer Verfahren hervor, da es ohne die genaue Karte des Parkhauses nicht möglich wäre einen Pfad zu planen oder eine Wegvorgabe abzufahren.

Anschließend beschreiben wir in Kapitel 9 einen Roboter, der als Assistent für Fußgänger eingesetzt werden kann. Dieser mobile Roboter ist in der Lage sowohl in dicht bevölkerten Fußgängerzonen zu navigieren als auch lange Strecken im Stadtgebiet zurückzulegen. Dabei heben wir neben der Beschreibung des Gesamtsystems besonders unser Verfahren zur Kartierung von großflächigen Umgebungen und unser effizientes Planungsverfahren hervor.

Zum Abschluss fassen wir in Kapitel 10 die Ergebnisse dieser Arbeit zusammen und liefern zusätzlich einen Ausblick auf mögliche weitere wissenschaftliche Fragestellungen. Zusammengefasst werden in dieser Arbeit innovative Techniken vorgeschlagen, die es einem Roboter ermöglichen eine Zustandsschätzung durchzuführen. Insbesondere entwickeln wir Ansätze, die es einem Roboter erlauben ein präzises Modell seiner Umgebung zu schätzen. Alle von uns in dieser Arbeit vorgestellten Verfahren wurden sowohl mit echten Robotern als auch mit simulierten Daten getestet. Neben den Verbesserungen gegenüber dem bisherigen Stand der Technik verdeutlichen unsere Experimente mit echten Robotern, dass unsere Algorithmen in der Praxis angewendet werden können. Wir sind der Ansicht, dass unsere Arbeit dadurch einen Beitrag hin zu nützlichen Robotern im täglichen Arbeitsablauf und im Alltagsleben leistet.

Acknowledgements

It is my immense pleasure to thank all the people that contributed to this work.

First of all, I would like to thank my advisor Wolfram Burgard. This thesis would not have been possible without his continuous support. He provided me encouragement, inspiration, opportunities, and freedom to pursue own ideas. I learned a lot of things while at the same time having plenty of fun. I would also like to thank Thomas Brox for reviewing this thesis.

My deepest thanks to my co-advisor Giorgio Grisetti, who supported me with valuable ideas and good advice. I enjoyed the countless fruitful discussions over the years and it was a pleasure to work with him.

I would like to thank all colleagues and co-authors for the collaborations and enlightening discussions: Dmitri Dolgov, Christian Dornhege, Udo Frese, Giorgio Grisetti, Dirk Hähnel, Christoph Hertzberg, Dominik Joho, Alexander Kleiner, Kurt Konolige, Henrik Kretzschmar, Benson Limketkai, Kai Ni, Patrick Pfaff, Michael Ruhnke, Cyrill Stachniss, Bastian Steder, Hauke Strasdat, Juan D. Tardós, Sebastian Thrun, Rudolph Triebel, Regis Vincent, and Kai M. Wurm. Without their support many projects could not have been realized.

Many thanks to Dirk Hähnel, Dmitri Dolgov, and Sebastian Thrun for making my visit to Stanford possible, for giving me the opportunity to work on their research car, and for their hospitality.

Furthermore, I thank Slawomir Grzonka, Michael Ruhnke, and Bastian Steder for the pleasant working atmosphere and their friendship. I would also like to thank all colleagues at the AIS lab for the great and friendly atmosphere.

For their valuable feedback on earlier versions of this document I thank Giorgio Grisetti, Slawomir Grzonka, Henrik Kretzschmar, Michael Ruhnke, Cyrill Stachniss, Christoph Sprunk, and Bastian Steder.

I thank all the members of the EUROPA project for the collaboration and our joint efforts to realize a great robotic system.

I would also like to thank Susanne Bourjaillat, Kristine Haberer, Michael Keser, Bettina Schug, and Daniela Wack for their administrative and technical support.

I thank all people who made their software and their data sets publicly available: Mike Bosse, Dirk Hähnel, Michael Kaess, John Leonard, Eduardo Nebot, Edwin Olson, and Cyrill Stachniss.

My deepest gratitude goes to my family for their support and love they gave me during all these years.

This work has partly been supported by the European Commission under contract number FP7-231888-EUROPA and ERC-AG-PE7-267686-LifeNav as well as by the German Research Foundation (DFG) under contract number SFB/TR-8.

Contents

Chapter 1

Introduction

Being able to navigate is commonly regarded as one of the core prerequisites for providing truly autonomous robots. This ability enables the robot to offer a large set of services to the potential customer. Efficient navigation, for example, allows the robot to pick-up goods and deliver them to a certain location, to reliably clean an area, or to offer surveillance of an environment. In addition to that, autonomous navigation is a key component of a robot assisting humans in potentially dangerous situations, such as search-and-rescue scenarios. Hence, navigation is one of the major research topics in the robotics community.

To realize the envisioned applications, which are mentioned above, we need to fulfill certain requirements, so that a robot is regarded as useful. For example, a robot which performs pick-and-place tasks or offers guidance in city centers needs to be aware of its own position in the environment and it needs to have an accurate model of the environment for planning an appropriate path. A robot, which should guide a human to a certain place or has to deliver goods, is only regarded as helping hand if the location is reliably reached within the expected time frame.

As a robot that operates in unknown or unpredictable environments needs to localize itself in the world and to map the environment at the same time, we have to tackle a complex state estimation problem, which is commonly referred to as Simultaneous Localization and Mapping (SLAM). Solutions to SLAM need to address both aspects. First, they need to estimate what the environment looks like. This is the mapping part which deals with integrating the information obtained by the sensors of the robot into an appropriate representation. Second, the localization component has to estimate the position of the robot with respect to the model of the environment. Each task is straightforward to realize on its own if the counterpart is given. Solving both tasks at the same time, however, is difficult, as the knowledge about the pose impacts mapping and vice versa. Hence, the combination of both tasks results in a difficult state estimation problem which may involve thousands of variables, for instance, the pose of the robot at any time and all observations acquired by the robot.

A promising representation for addressing the SLAM problem is a so-called pose-graph. In such a graph, the poses of the robot are encoded as nodes and the noisy sensor information is modeled as edges or soft constraints between the nodes. The solution is then obtained by determining the configuration of the robot poses that best satisfies all soft constraints.

While exploring the environment the robot may, however, require intermediate solutions to the SLAM problem, e.g., it needs to know what the environment looks like for planning the next action. Hence, the solution has to be provided online and in a fast manner. Otherwise, the robot might have to stop frequently before being able to continue executing its task. Furthermore, the intermediate solution might help the robot to interpret the sensor information correctly. For

example, to recognize a previously visited location and thus solve the complex data association problem that arises in the context of SLAM.

Additionally, the robot needs to be aware of parameters to interpret the sensor data. A sensor measures physical quantities, such as the voltage difference, the phase shift, or the time-of-flight. To incorporate these measurements into the state estimation process, they are typically mapped into a more abstract space, such as transforming the phase-shift into the distance to the object causing the reflection of the beam. The mapping, however, requires knowledge about a set of parameters, which might be constant (e.g., the wavelength of the laser beam) or they might change over time. For instance, estimating the odometry position of the robot by counting the rotation of the wheels needs to be aware of the wheel diameters, which are subject to change over time for inflated tires and are affected by the overall weight of the platform. Currently, most state-of-the-art approaches for SLAM assume these parameters as given and ignore the fact that the parameters might change over time. If the variation of the parameters is not appropriately modeled, it leads to systematic errors in the state estimation that in turn may affect the quality of the result.

While the above mentioned online calibration has the potential to increase the accuracy of the result, we may also take into account external sources of information for improving the estimate. Consider, for example, the large variety of map information which is publicly available on the Internet. The robot may exploit this information by relating its sensor measurements to the prior map. The majority of SLAM approaches, however, estimates the model of the environment from scratch. Yet, exploiting the prior information may lead to a better estimate if we augment the state estimation with the soft constraints derived from the prior. We have to consider that the prior might only be partially available, though.

As we have mentioned above, a SLAM algorithm consists of a mapping and a localization part. To realize the mapping part, the robot may, for example, consider the well-known occupancy grid map algorithm or other appropriate representations which allow the robot to localize itself, such as a point cloud. While computing the model the robot has to cope with the remaining uncertainty over its position and the noise in the sensor data. The aforementioned occupancy grid map algorithm, for example, assumes the poses as known and only models the uncertainty in the range data. If our algorithm for estimating the model, however, also appropriately models these uncertainties, the robot might be able to increase the quality of the model of the environment.

Such a model may serve as the basis for autonomous navigation, as it allows the robot to plan an appropriate path. The majority of existing solutions for autonomous navigation, however, enforces a certain set of prerequisites. For example, most approaches assume that the work space of the robot is indoors. Thus, it is sufficient to employ 2D data structures. One goal of this thesis is to realize navigation systems for mobile robots that allow the robot to operate in an environment without enforcing conditions such as modifications to the environment. For instance, the robot should be able to operate in urban structures without depending on artificial beacons to facilitate localization. Furthermore, we have to deal with a potentially complex and large-scale environment. This imposes challenges to the whole navigation system and the underlying state estimation technique in particular.

The methods presented in this thesis address a large variety of potential applications. Consider, for example, the capabilities of nowadays off-the-shelf cars. Such vehicles offer a subset of autonomous navigation features, such as lane keeping or safety braking. Recently, cars are also able to park autonomously if the human drives the car spatially close to the parking spot before starting the autonomous maneuver. Additionally, research cars implement long term navigation on the street, but they typically rely on GPS and a known network of streets. Relax-

ing these strict assumptions has the potential to immensely expand the market of mobile robots by offering safe driving and more comfort in tedious tasks, such as parking the car in a narrow parking garage, where GPS is not available.

In summary, we have identified the following key problems for estimating the state while learning a model of an environment or while utilizing such a model for navigation:

- How to efficiently and robustly estimate the state including the position of the robot and the map?
- How to estimate the underlying parameters which are required to correctly interpret the sensor information?
- How to incorporate prior information about the structure of the environment to improve the quality of the map?
- How to estimate an accurate model of the environment that deals with the noise in the sensor information and the uncertainty in the positions of the robot?
- How to build real-world systems which are able to navigate autonomously through complex environments?

While addressing the key problems raised above we always have to cope with the fact that all information perceived by the robot through its sensors is imperfect. In other words, the sensor information is affected by noise. Thus, our approaches have to model this adequately. We account for the noise in the real-world by implementing sound probabilistic approaches which explicitly model those noisy measurements under some reasonable assumptions.

Throughout this thesis, we provide innovative probabilistic approaches that enable a mobile robot

- to determine the maximum likelihood estimate by phrasing and solving a large least squares problem,
- to estimate the calibration parameters online without interfering with the mission of the robot,
- to integrate prior information about the geometry of the environment derived from publicly available sources,
- to jointly optimize the poses of the robot as well as the sensor measurements to appropriately deal with the uncertainty in the data, and
- to navigate through urban environments without prerequisites.

This thesis is organized in two parts. In Part I “Graph-Based Optimization for SLAM”, which includes Chapter 2 to 7, we present our innovative approaches for addressing the SLAM problem, a comparison with other state-of-the-art approaches for SLAM, and our novel extensions to graph-based SLAM. In detail, in Chapter 2, we explain the probabilistic formulation that allows us to model the SLAM problem. Subsequently, we show that this leads us to a least squares estimation problem, which can efficiently be solved by numerical approaches that exploit the typical properties of SLAM. This technique is commonly known as graph-based SLAM since it employs a graph structure. Whereas our approach efficiently addresses the SLAM problem, it is by no means restricted to SLAM. In fact, it is also applicable to other related problems, such as calibration, Structure from Motion, or model fitting.

In Chapter 3, we present a novel approach which applies the divide-and-conquer principle to solve the least squares problem arising in SLAM. The advantages of our approach over the standard methods are twofold. In addition to improving the convergence properties, it also yields an efficient algorithm for approximating the intermediate solution during online mapping, i.e., in situations where the robot is still collecting data and the currently best estimate is required, for instance, to plan the next action.

In Chapter 4, we propose a metric which enables us to compare the results of our graph-based SLAM technique with other state-of-the-art approaches. We show that graph-based SLAM yields results which are typically superior to other approaches to SLAM.

Subsequently, in Chapter 5, we discuss a novel approach that allows the robot to simultaneously estimate the underlying parameters of the platform while performing SLAM. Such parameters might include the kinematic parameters of the robot which are subject to change during the life-time of the robot. Existing approaches to SLAM typically assume these parameters as given and static, whereas our approach includes them into the state estimation process. For instance, in situations where the robot has to carry a load, the diameter of the wheels is affected by the total weight of the platform. Our approach enables us to accurately and efficiently account for those changes while the robot carries out its mission.

Afterwards, in Chapter 6, we introduce an innovative approach for integrating prior information about the structure of the environment into the graph-based SLAM formulation. Our approach derives this information by considering aerial images, which are publicly available from various sources. We access the prior information by localizing the robot in the aerial image by matching its 3D sensor measurements to the aerial image. This allows the robot to improve the accuracy of the maps.

In Chapter 7, we describe a novel approach which further improves the quality of the maps obtained by a SLAM algorithm. Particularly, we explicitly model the noise in the range measurements by taking into account the physical principles of the sensors. To this end, we model the range data as samples whose positions are refined in a joint optimization problem, which also includes the poses of the robot, for estimating the maximum likelihood state. Compared to standard techniques, our approach is able to reduce the remaining uncertainty in the estimated model of the environment.

In Part II "Applications", which includes Chapter 8 and 9, we present two real-world robotic systems, which apply the techniques developed in this thesis to challenging scenarios. In Chapter 8, we develop an approach that enables a car to park autonomously in a complex multi-level parking garage starting from outside the parking garage. This system demonstrates the benefits of the approaches presented in this thesis as without the map, which has been estimated by our SLAM technique, the car would be unable to plan a path or to follow a given track.

In Chapter 9, we describe a system for a robotic pedestrian assistant which is able to navigate in crowded pedestrian zones as well as traveling long distances through urban environments. In particular, we present our techniques for mapping such a large-scale environment and our approach for efficient planning. Additionally, we give an overview over the complete navigation system. Finally, we summarize our results in Chapter 10 and we furthermore discuss possible future work.

1.1 Key Contributions

The key contributions of this thesis are novel approaches for estimating an accurate model of an environment and also for state estimation in general. Furthermore, we show real-world systems which build upon the techniques developed in this thesis.

To summarize, we propose:

- an approach for addressing the optimization of factor graphs in a general manner with a particular emphasis on SLAM (Chapter 2),
- an approach for optimizing a factor graph online based on a hierarchy for efficient and robust estimation (Chapter 3),
- a metric for evaluating the accuracy of different SLAM algorithms (Chapter 4),
- a technique for estimating the calibration parameters of the robot online while mapping the environment (Chapter 5),
- an approach for integrating prior information into graph-based SLAM (Chapter 6),
- a method for estimating highly accurate maps (Chapter 7), and
- navigation systems for a car and for a robotic pedestrian assistant, which demonstrate the real-world applicability of the methods developed in this thesis (Chapters 8 and 9).

1.2 Open-Source Software

Some of the algorithms presented in this thesis have been released as open-source. This allows other researchers to build their own approaches on top of ours, to compare their approaches against ours on several data sets, to verify our results, and to adapt our software to their specific needs. In detail, we released the following software:

- g^2o (http://openslam.org/g2o) implements a general graph optimization framework which allows us to optimize graph-embeddable functions, such as the solution to a graph-based SLAM instance. It has been published under the BSD-license to facilitate the usage in commercial applications and other open-source software. g^2o was developed in collaboration with Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. See Chapter 2 for a description of the approach.
- HOG-Man (http://openslam.org/hog-man) is an approach for graph-based SLAM. It provides a highly efficient error minimization procedure which considers that the underlying space is a manifold and not a Euclidean space. It furthermore generates a hierarchy of pose-graphs which is used perform the operations during online mapping in a highly efficient way. The approach works in 2D and 3D. HOG-Man was developed in collaboration with Giorgio Grisetti and Cyrill Stachniss. See Chapter 3 for a description of the approach.
- The SLAM Evaluation toolbox (http://ais.informatik.uni-freiburg.de/slamevaluation) allows us to compare the accuracy of SLAM algorithms given a set of manually verified relations. We also offer the relations for data sets that are frequently considered in the research community. The whole framework is a joint effort with Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, Alexander Kleiner, and Wolfram Burgard. See Chapter 4 for a description of the approach.

- The framework to simultaneously estimate the parameters of the kinematics of the robot and the position of the on-board sensor is released as extension to the g^{2}o framework. The approach was developed in collaboration with Giorgio Grisetti and Wolfram Burgard. See Chapter 5 for a description of the approach.

- SSA2D (http://openslam.org/ssa2d) is an extension of 2D graph-based SLAM. It further optimizes the maps obtained with graph-based SLAM by iteratively refining the poses and the range measurements in a joint optimization problem. In contrast to standard approaches, the range scans are not treated as rigid. This allows us to obtain highly accurate maps of an environment. SSA2D solves the underlying optimization problem with g^{2}o. SSA2D is a joint effort with Michael Ruhnke, Giorgio Grisetti, and Wolfram Burgard. See Chapter 7 for a description of the approach.

- SPA2D (http://www.ros.org/browse/details.php?name=vslam) is an efficient implementation for optimizing pose-graphs while mapping the environment. It applies a variant of the Levenberg-Marquardt algorithm. SPA2D was developed in collaboration with Kurt Konolige, Giorgio Grisetti, Wolfram Burgard, Benson Limketkai and Regis Vincent. We refer to our joint publication [110] for the details.

1.3 Publications

Parts of this thesis have been published in international journals and conference proceedings. The following list gives an overview about the individual publications in chronological order.

Journal Articles

- R. Kümmerle, G. Grisetti, and W. Burgard. Simultaneous parameter calibration, localization, and mapping. *Advanced Robotics*, 26(17):2021–2041, 2012.

- R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.

- G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

- R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

Conference Proceedings and Workshops

- R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A navigation system for robots operating in crowded urban environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013. Accepted for Publication.

- G. Grisetti, R. Kümmerle, and K. Ni. Robust optimization of factor graphs by using condensed measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

- M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate 3D surface models by sparse surface adjustment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

- R. Kümmerle, G. Grisetti, and W. Burgard. Simultaneous calibration, localization, and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

- R. Kümmerle, G. Grisetti, C. Stachniss, and W. Burgard. Simultaneous parameter calibration, localization, and mapping for robust service robotics. In *Proc. of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2011.

- M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Range sensor based model construction by sparse surface adjustment. In *Proc. of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2011.

- R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

- M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

- G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

- W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós. A comparison of SLAM algorithms based on a graph of relations. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

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

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

This thesis does not report on the following publications, which were written during the time as a research assistant

- K. M. Wurm, H. Kretzschmar, R. Kümmerle, C. Stachniss, and W. Burgard. Identifying vegetation from laser data in structured outdoor environments. *Robotics and Autonomous Systems*, 2012. In Press.

- K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

- K. M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

- R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics*, 25:346–359, 2008.

- P. Pfaff, R. Kümmerle, D. Joho, C. Stachniss, R. Triebel, and W. Burgard. Navigation in combined outdoor and indoor environments using multi-level surface maps. In *Workshop on Safe Navigation in Open and Dynamic Environments*, 2007.

- R. Kümmerle, P. Pfaff, R. Triebel, and W. Burgard. Active Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Fachgespräche Autonome Mobile Systeme (AMS)*, 2007.

- R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, 2007.

1.4 Collaborations

Parts of this thesis are the outcome of collaborations with others. Particularly, the hierarchy of pose-graphs (HOG-Man) presented in Chapter 3 was jointly developed with Giorgio Grisetti, Cyrill Stachniss, Christoph Hertzberg, and Udo Frese. Subsequently, the approach has been extended to arbitrary graphs in collaboration with Giorgio Grisetti and Kai Ni. The metric for evaluating SLAM discussed in Chapter 4 is the result of fruitful discussions with all the authors of [26, 123]. The approach for including prior information (Chapter 6) was jointly investigated with Bastian Steder, whose main contribution is the pre-processing of the sensor measurements discussed in Sections 6.1.2 and 6.1.3. In Chapter 7, we present our approach for obtaining highly accurate laser-based maps which is joint work with Michael Ruhnke, whereas the author of this thesis was mainly involved in constructing the dedicated optimization approach. The approach presented in Chapter 8 builds on top of the software of the Stanford Racing Team. In particular, the author collaborated with Dirk Hähnel, Dmitri Dolgov, and Sebastian Thrun, who provided their software and expertise about autonomous navigation with a car. The main contribution of the author of this thesis are the approaches to 3D SLAM and 3D localization as well as the global planner operating on the map. The navigation system for a robotic pedestrian assistant which is described in Chapter 9 is the result of a joint effort with Michael Ruhnke, Bastian Steder, and Cyrill Stachniss, whereas the main contributions of the author of this thesis are in the mapping algorithm and the planning components.

1.5 Notation

The following table summarizes the symbols used throughout this work.

Symbol	Meaning
$a, b, x, \ldots$	scalar value
$\mathbf{x}, \mathbf{y}, \ldots$	vector; if not stated otherwise, a column vector
$A, H, \ldots$	matrix
$\mathbf{x}^\top, A^\top$	transpose of a vector or a matrix
$\|\mathbf{x}\|$	norm (length) of a vector
$\|A\|$	determinant of a matrix
$\mathrm{diag}(\mathbf{x})$	matrix with $\mathbf{x}$ on the main diagonal
$(\ldots)$	vector with the given scalar values
$\{\ldots\}, \mathscr{C}, \mathscr{S}, \ldots$	set
$\|\mathscr{S}\|$	number of elements in the set
$\langle\ldots\rangle$	tuple, an ordered set
$p(a)$	probability distribution of a random variable a
$p(a \mid b)$	conditional probability of a given b
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ

Part I

Graph-Based Optimization for SLAM

Chapter 2

A General Framework for Graph Optimization

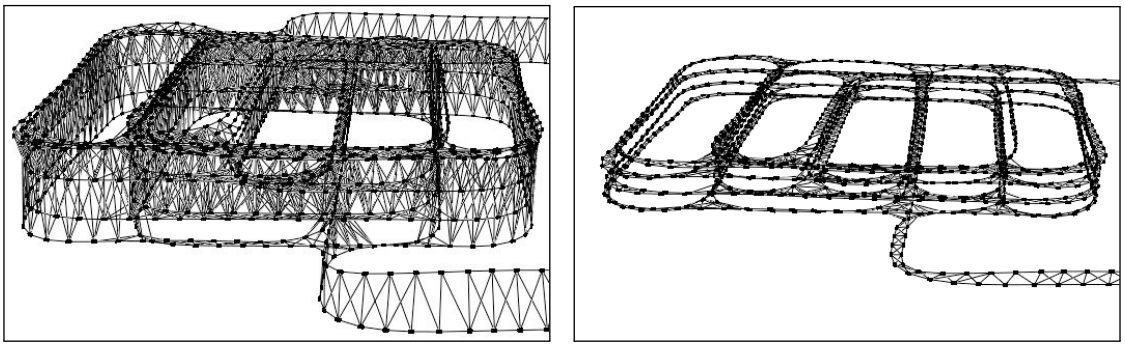
Building a map of an environment is a key task of autonomous robots. As we will see, simultaneous localization and mapping (SLAM) can be phrased as least squares optimization, which can be represented by a graph. In this chapter, we describe both the probabilistic formulation for SLAM and the general structure of such problems. Furthermore, we present g^{2}o, our open-source framework for optimizing graph-based nonlinear error functions. Our system has been designed to be easily extensible to a wide range of problems and a new problem typically can be specified with minimal effort. Currently, we provide solutions to the optimization problems arising in several variants of SLAM and Structure from Motion (SfM). We performed an evaluation on a wide range of real-world and simulated data sets. The results demonstrate that while being general g^{2}o offers a performance comparable to implementations of state-of-the-art approaches for the specific problems.

• • • • • • • • • • • •

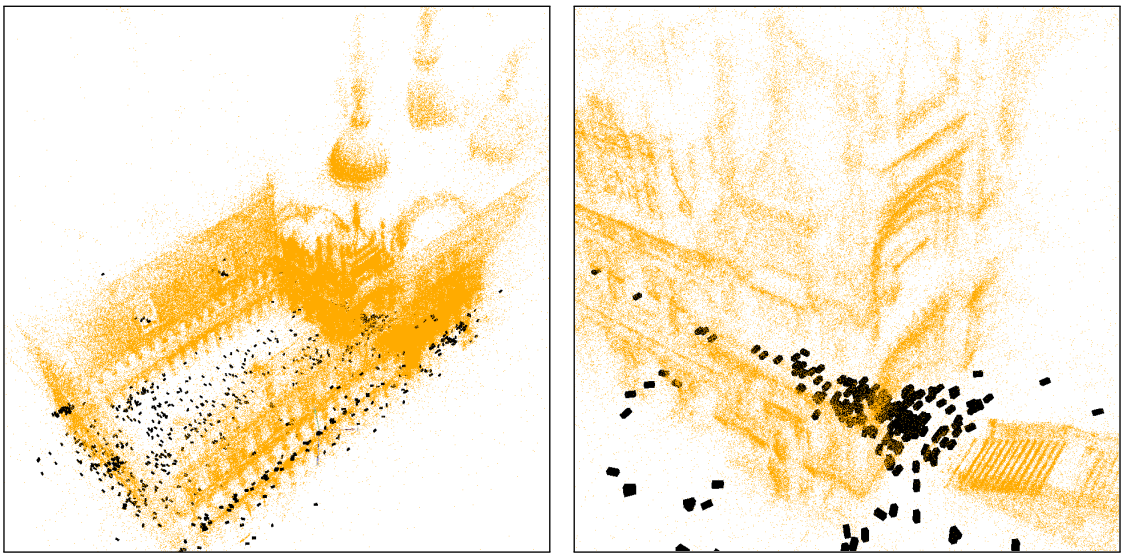
As we have discussed in the first chapter of this thesis, the ability to acquire accurate models of the environment is commonly regarded as one of the fundamental preconditions for truly autonomous robots. Within the robotics community the problem of estimating a model of the environment with a mobile robot is known as simultaneous localization and mapping (SLAM). In the context of mobile robots, these models typically are maps of the environment that support different tasks including localization and path planning.

Within this chapter we explain the probabilistic formulation of the SLAM problem, which then leads us to nonlinear least squares optimization whose solution addresses SLAM. Such a least squares optimization furthermore allows us to address a wide range of problems in robotics as well as in computer-vision, as they involve the minimization of a nonlinear error function that can be represented as a graph. Typical instances are as already mentioned SLAM [43, 78, 110, 143, 164, 201] or Structure from Motion (SfM) [107, 140, 210], as well as calibration [190]. The overall goal in all these is to find the configuration of parameters or state variables that yield the best explanation for the set of measurements affected by Gaussian noise.

In the graph representation each node indicates a state variable to optimize, each edge connects to nodes and models an observation between a subset of the nodes. For instance, in graph-based SLAM each state variable can be either the position of the robot in the environment or the



(a) SLAM (Parking Garage)



(b) Bundle Adjustment (Venice)

Figure 2.1: Real-world data sets processed with our system. The images in (a) depict a 3D pose graph of a multi-level parking garage. Whereas the image on the left shows the initial states, the image on the right depicts the result of the optimization process. (b) Full and zoomed view of the Venice Bundle Adjustment data set after being optimized by our system. The data set consists of 871 camera poses and 2,838,740 projections.

location of the landmarks in the map that can be observed with the robot's sensors. One goal of this chapter is to provide the basic algorithms for addressing nonlinear least squares. For example, we will explain well-known methods like Gauss-Newton and Levenberg-Marquardt with an additional emphasis on how to apply them in a SLAM setting. Particularly, we exploit our knowledge about the characteristic properties of SLAM, which results in an efficient implementation of the aforementioned methods.

In this chapter, we present a general framework for performing the optimization of nonlinear least squares problems that can be represented as a graph. We call this framework g^{2}o (short for "General Graph Optimization"). Figure 2.1 shows two examples out of the large variety of problems that can be solved by using g^{2}o as an optimization back-end. The proposed system achieves a performance that is comparable or even better than the performance of implementations of state-of-the-art methods, while being able to accept general forms of nonlinear measurements.

We achieve efficiency by utilizing algorithms that

- exploit the sparse connectivity of the graph,
- take advantage of the special structures of the graph that often occur in the problems mentioned above,
- use advanced methods from numerics to solve sparse linear systems,
- and utilize the features of modern processors like SIMD instructions and optimize the cache usage.

Despite its efficiency, g^{2}o is highly general and extensible: a 2D optimization back-end algorithm can be implemented in less than 30 lines of code. The user only has to specify the error function and its parameters.

The remainder of this chapter is organized as follows. We first derive the probabilistic formulation of SLAM in Section 2.1 which leads us to a nonlinear least squares optimization problem. In Section 2.2 we illustrate a typical setup of a complete SLAM system, which allows us to construct the least squares problem, whereas we elaborate least squares in Section 2.3. Subsequently, in Section 2.4 we discuss the features provided by our approach — the g^{2}o framework. After presenting an extensive experimental evaluation of g^{2}o in Section 2.5, we discuss the related work with a particular emphasis on solutions to the optimization problems arising in SLAM and Structure from Motion in Section 2.6.

2.1 Probabilistic Formulation of SLAM

In the following, we will characterize SLAM by its probabilistic formulation [205]. To this end, we will first introduce some notation which we will us throughout this thesis.

The pose of the robot is described by $\mathbf{x}_t$, whereas the index t denotes the position at time step t. For a robot operating on a two-dimensional plane, the pose consists of $\mathbf{x} = (x, y, \theta)^\top$ describing the current location and orientation of the robot relative to a given global reference frame. Dropping the flat ground assumption leads to a six-dimensional vector. Here, the pose $\mathbf{x}$ teams the location, which is a three-dimensional vector, with the attitude in space that also requires three parameters. A possible choice for representing the rotation are the Euler angles $(\phi, \vartheta, \psi)^\top$, such that in 3D SLAM the pose is given by $(x, y, z, \phi, \vartheta, \psi)^\top$. The whole trajectory taken by a robot up to time-step T is given by

$$\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}, \tag{2.1}$$

which describes the path taken by the robot as a sequence of poses. Here, T denotes the end of the sequence, it might span over the whole life-time of the robot, though.

While the robot drives along such a path its odometers allow the robot to measure the motion. We will denote such a measurement by $\mathbf{u}_t$, which stands for the movement between $t-1$ and t. Again, a sequence of odometry measurements is given by

$$\mathbf{u}_{1:T} = \{\mathbf{u}_1, \ldots, \mathbf{u}_T\}. \tag{2.2}$$

If the odometer would be perfect, chaining up the odometry measurements would lead to the true path taken by the robot. The measurements provided by odometry, however, are — as all measurements — affected by errors (such as a wheel slipping on the ground) and thus we only

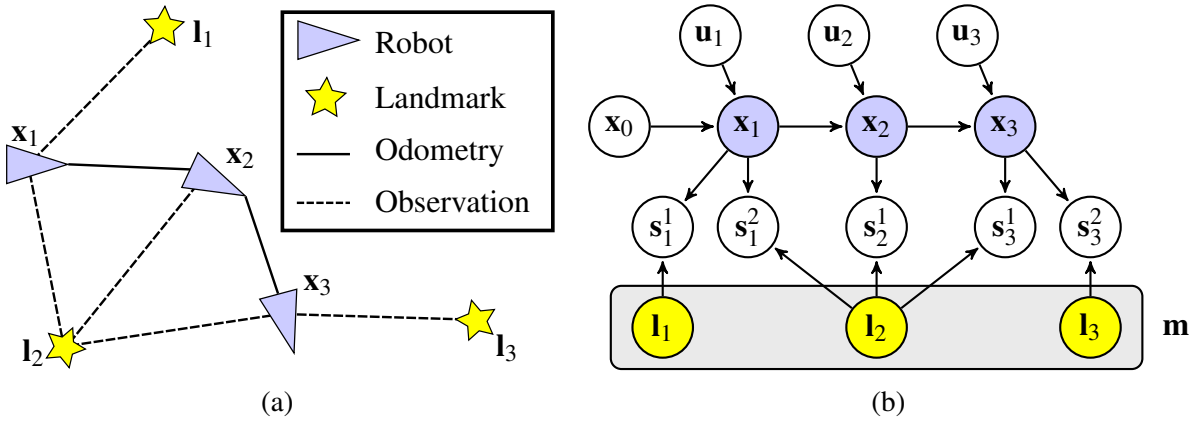


Figure 2.2: A small example of a SLAM instance and its corresponding Bayesian network. (a) The robot, indicated by a blue triangle, drives around in the environment and detects landmarks which are depicted as yellow stars by its sensors. (b) The Bayesian network of a SLAM problem.

obtain noisy measurements. We assume that the noise of all our measurements is white, i.e., it can be described by a Gaussian or normal distribution with zero mean. This assumption is commonly made in the robotics community [205].

Formally, an n-dimensional random variable $\mathbf{x}$ is normally distributed with mean μ and covariance Σ if the density $p(\mathbf{x})$ has the following form

$$p(\mathbf{x}) \sim \mathcal{N}(\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{2.3}$$

where $|\Sigma|$ corresponds to the determinant of Σ. In general, however, the noise is not white. We can account for such systematic noise by performing an accurate calibration, as we will see in Chapter 5.

In addition to the odometer, the robot is typically equipped with a sensor which provides the robot with measurements about the environment. Let us denote the map of the environment by $\mathbf{m}$ and we call the sensor measurements $\mathbf{s}_t$. Throughout this work, we assume the environment to be static. As above, the sequence of measurements obtained by the robot is denoted by

$$\mathbf{s}_{1:T} = \{\mathbf{s}_1, \ldots, \mathbf{s}_T\}. \tag{2.4}$$

Such a measurement might, for example, be the observation of a certain feature or a landmark that is present in the environment.

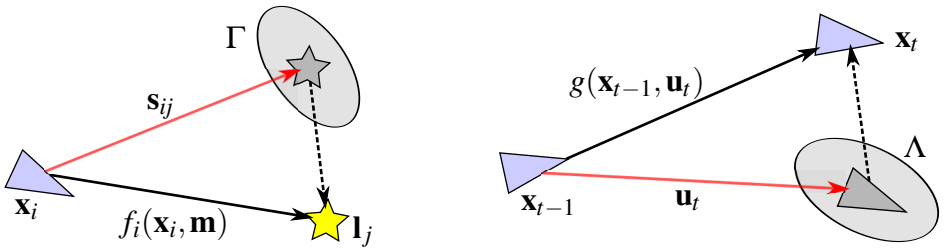
Given this notation, we are able to describe the SLAM problem as the process which estimates the posterior

$$p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{s}_{1:T}, \mathbf{u}_{1:T}). \tag{2.5}$$

Applying Bayes' rule allows us to factor the posterior at time step t:

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta\, p(\mathbf{s}_t \mid \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}), \tag{2.6}$$

where η is a normalizer accounting for the omitted denominator on the right hand side. Note that we reuse η in different equations throughout this section where η each times takes different values to fulfill the equation sign. In addition to the Gaussian assumption, we assume that the



(a) error of a landmark (b) error of a pose constraint

Figure 2.3: Illustration of the sensor and the odometry model. (a) A robot observes a landmark. By using the sensor model $f_i(\cdot)$, we can compute an expected measurement and analyze the difference between the measurement and the expected measurement. (b) Likewise, in case of a pose-pose constraint, the measurement function is employed to compute the difference between the current and the expected location of the observed pose.

state $\mathbf{x}_t$ encodes all relevant information up to time t. This is known as the Markov assumption and it enables us to simplify the first factor in Eq. 2.6 as

$$p(\mathbf{s}_t \mid \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) \overset{\text{Markov}}{=} p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}). \tag{2.7}$$

Furthermore, we can decompose the second term in Eq. 2.6 by splitting $\mathbf{x}_{0:t}$ into $\mathbf{x}_{0:t-1}$ and $\mathbf{x}_t$. This yields

$$\begin{aligned} p(\mathbf{x}_{0:t}.\mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) &= p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{m}, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) && (2.8) \\ &\overset{\text{Markov}}{=} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t-1}). && (2.9) \end{aligned}$$

By inserting Eq. 2.7 and Eq. 2.9 into the original form of the posterior given in Eq. 2.6, we obtain

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{0:t-1}, \mathbf{m} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t-1}), \tag{2.10}$$

which is a recursive definition, whereas the first pose is specified by the prior $p(\mathbf{x}_0)$. Such a prior might be the position of the robot in a global reference frame. We, however, are able to express the posterior in closed form as

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{x}_0) \prod_t p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t). \tag{2.11}$$

Figure 2.2a gives an example of a SLAM problem. The robot drives through an environment and perceives landmarks by its sensor. In this example, the map $\mathbf{m} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3)$ consists of three landmarks detectable by the sensor of the robot. Furthermore, odometry measurements provide information about the motion of the vehicle from one time step to the subsequent one. In Figure 2.2b we visualize the Bayes network corresponding to the situation depicted in Figure 2.2a. For indicating the individual sensor measurements and their relation to the map, we applied upper indices to the sensor measurement $\mathbf{s}$. For example, at time $t = 1$ the robot senses $\mathbf{s}_1 = (\mathbf{s}_1^1, \mathbf{s}_1^2)$ corresponding to two different landmarks in the environment.

By looking at Eq. 2.11 and the resulting Bayes network shown in Figure 2.2b, we identify two probability functions which we have to specify, namely the sensor model $p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{m})$ and the motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$. Following the Gaussian assumption as stated above, we assume that the measurement error is normally distributed according to $\mathcal{N}(f_t(\mathbf{x}_t, \mathbf{m}), \Gamma)$, where $f_t(\cdot)$ is the measurement function or sensor model generating an expected measurement at a

certain position and Γ is the covariance of the measurement error. Intuitively, $f_t(\cdot)$ computes the displacement of the landmarks as they would have been observed from a given location of the robot. Note that we assume the feature correspondence in each time-step is known in the sensor model $f_t(\cdot)$, which we denote by the subscript t. Likewise, $\mathcal{N}(g(\mathbf{x}_{t-1}, \mathbf{u}_t), \Lambda)$ models the motion of the robot, where $g(\cdot)$ is a deterministic function computing how the robot moves if it executes the command $\mathbf{u}_t$, and Λ is the covariance of the motion error. Figure 2.3 shows an illustration of the involved quantities. Hence, we obtain

$$p(\mathbf{s}_t \mid \mathbf{x}_t) = \eta \exp\left(-\frac{1}{2}\left(f_t(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t\right)^\top \Gamma^{-1} \left(f_t(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t\right)\right) \tag{2.12}$$

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) = \eta \exp\left(-\frac{1}{2}\left(g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\right)^\top \Lambda_t^{-1} \left(g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\right)\right). \tag{2.13}$$

Putting Eq. 2.12 and Eq. 2.13 into Eq. 2.11 and taking the negative logarithm, we get

$$\begin{aligned} -\log p(\mathbf{x}_{0:t} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) = c + \frac{1}{2}\Bigg[& \mathbf{x}_0^\top \Omega_0 \mathbf{x}_0 + \sum_t (f_t(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t)^\top \Gamma_t^{-1} (f_t(\mathbf{x}_t, \mathbf{m}) - \mathbf{s}_t) \\ & + (g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t)^\top \Lambda_t^{-1} (g(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t)\Bigg], \end{aligned} \tag{2.14}$$

where c is a constant reflecting the various normalizers used in the equations above and $\mathbf{x}_0^\top \Omega_0 \mathbf{x}_0$ represents the log-likelihood of $p(\mathbf{x}_0)$. Here, we anchor the first pose at $\mathbf{0}$. Different anchor locations, such as a known offset to a reference frame, can be achieved by computing a difference between $\mathbf{x}_0$ and the desired location instead.

We can obtain the optimal estimate $\mathbf{x}^+$ by the maximum a posteriori (MAP) estimate

$$\mathbf{x}^+ = \operatorname*{argmax}_{\mathbf{x}} p(\mathbf{x}_{0:t} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) \tag{2.15}$$

$$= \operatorname*{argmin}_{\mathbf{x}} -\log p(\mathbf{x}_{0:t} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}). \tag{2.16}$$

To simplify the notation, let us collect the odometry $\mathbf{u}$ and the sensor measurements $\mathbf{s}$ into a single variable denoted by $\mathbf{z}$. Furthermore, let us introduce a function $\mathbf{e}(\mathbf{x}, \mathbf{z})$ which either calls the above defined functions $g(\cdot)$ and $f_t(\cdot)$ to evaluate the difference as given in Eq. 2.14, or calculates the prior $p(\mathbf{x}_0)$, depending on the passed arguments. Likewise, Ω_k either reflects Γ_t^{-1}, Λ_t^{-1}, or Ω_0. If we then as a last step combine all pairs of indices for which a measurement is available into the set $\mathcal{G}$, we are able to rewrite Eq. 2.16 as

$$\mathbf{x}^+ = \operatorname*{argmin}_{\mathbf{x}} \sum_{k \in \mathcal{G}} \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k)^\top \Omega_k \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k). \tag{2.17}$$

The equation above is known as a least squares estimation problem. We will show in the next section how we can efficiently solve such problems and how to obtain a solution for SLAM while taking advantage of the characteristics of SLAM.

In the context of least squares, the measurements $\mathbf{z}$ are also denoted as constraints or factors and a factor graph is a commonly used representation for it. Such a graph describes the relations between the state variables. In detail, a factor graph is a bipartite graph having two kinds of nodes. A node either represents a variable $\mathbf{x}_i$ or it corresponds to one factor F_k, where

$$F_k = \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k)^\top \Omega_k \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k) \tag{2.18}$$

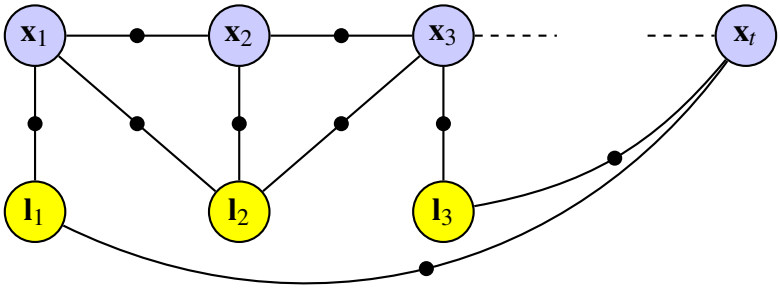


Figure 2.4: A factor graph representation of a SLAM problem. The odometry of the robot constraints the motion of the robot between subsequent time steps and the robot additionally observes landmarks in the environment.

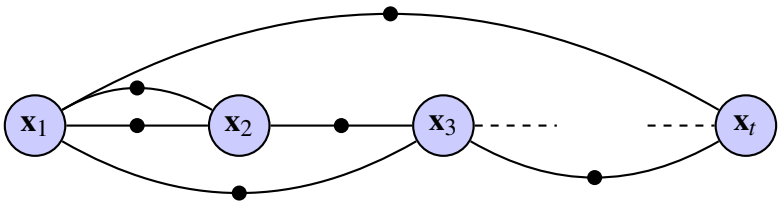


Figure 2.5: The pose-graph representation of the SLAM problem depicted in Figure 2.4. We can obtain such a representation, for example, by directly matching the raw sensor observation.

connects a subset of variables $\mathbf{x}_k$ which are related to each other by the observation $\mathbf{z}_k$. Figure 2.4 depicts an example for a factor graph. Here, the constraints either affect subsequent poses of the robot by the odometry measurement or relate the robot and a landmark by sensor measurements. In contrast to this, we can obtain a slightly different graph by directly relating the sensor observation to each other, for example, by determining the best transformation between the two observations. For range scans this is commonly referred to as scan-matching [19, 30, 188] and the estimated transformation is a so-called *virtual* measurement. This leads to a factor graph which only contains poses of the robot, see Figure 2.5. We refer to such a graph as pose-graph. Note that such a pose-graph could also be obtained by marginalizing out the landmarks.

So far, we have addressed the so-called full SLAM problem, as given in Eq. 2.11, which estimates the whole trajectory of the robot. In contrast to this, the online SLAM formulation only considers the most recent location of the robot

$$p\left(\mathbf{x}_t, \mathbf{m} \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}\right). \tag{2.19}$$

The typical solution to this problem is obtained by applying filtering techniques for recursively estimating the density. Strasdat *et al.* [200], however, show that even for short time frames filtering yields suboptimal results given nowadays computing power in terms of accuracy when compared to solving the full SLAM problem by a least squares optimization. In short time frames the linearization errors of filtering approaches are typically not the key restricting factor and we expect to obtain the best filtering performance. These errors, however, may become more evident for larger time intervals. Thus, we will focus only on the full SLAM problem instead, which allows us to re-linearize when needed, for determining a solution spanning over large time intervals.

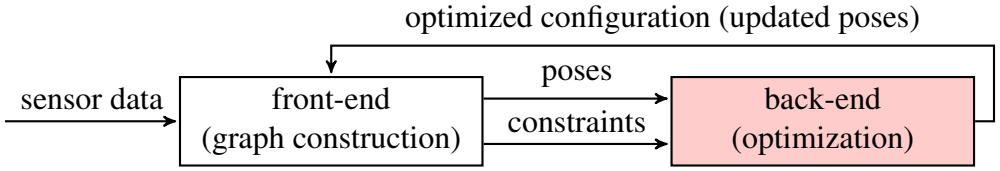


Figure 2.6: A typical setup for SLAM. A front-end processes the sensor data and passes the generated constraints along with the current robot poses to the optimization back-end. The back-end returns the updated robot poses to the front-end.

2.2 Front-End / Back-End

As we have seen, given the measurements and the data association, we are able to provide a maximum likelihood estimate by constructing a least squares problem. In this section, we will describe the essential components of such a SLAM system. Figure 2.6 depicts a typical setup.

For completeness, we briefly describe an algorithm for obtaining the constraints required for representing the SLAM problem by a factor graph. A full system addressing SLAM in such a way can be decoupled into two parts interacting with each other. The back-end calculates the maximum a-posteriori estimate given a set of constraints and an initial configuration — the poses of the robot — as input. In turn, the constraints are generated by the front-end. Particularly, the front-end has to solve the data association problem by identifying landmarks given the observations. To this end, the front-end needs to recognize places if the robots revisits them. This is known as loop-closing in the literature. The front-end here benefits from the poses updated by the back-end, as we can typically restrict the search area for possible loop closures to an area proportional to the uncertainty ellipsoid of the current robot location. Hence, the back-end needs to be efficient due to the demand for a continuously updated estimate of the model of the environment. Note that the decisions taken in the front-end are by no means final, e.g., the front-end may revoke a certain data association or introduce new correspondences at any time.

In our setup, the front-end generating the constraints, which serve as input for our back-end, is an own implementation of the approach proposed by Olson [163]. For example, when operating on 2D range data, it applies a correlative scan-matcher to estimate the motion of the robot between successive time steps along with a covariance representing the uncertainty of the estimated transformation. Furthermore, it obtains loop closures by matching the current scan against all scans which are within the ellipsoid covering 95 % of the Gaussian representing the uncertainty about the current pose of the robot. An approach based on spectral clustering filters false-positives. We can also apply the same principle to vision data whenever a robot is equipped with a camera. Running such a front-end along with an efficient back-end, we are able to build accurate maps of an environment as we will illustrate in the experiments in Section 2.5.

2.3 Nonlinear Least Squares

As outlined in Section 2.1, we can provide a solution to SLAM for a given data association by solving a least squares estimation problem, see the derivation leading to Eq. 2.17. Additionally, many other problems in robotics or in computer vision, for example, calibration or Structure from Motion, can be modeled by a least squares problem. Thus, we are in general interested in

finding the minimum of a function $F : \mathbb{R}^n \to \mathbb{R}$ of the form

$$F(\mathbf{x}) = \sum_{k \in \mathcal{G}} \underbrace{\mathbf{e}(\mathbf{x}_k, \mathbf{z}_k)^\top \Omega_k \mathbf{e}(\mathbf{x}_k, \mathbf{z}_k)}_{F_k}. \tag{2.20}$$

Within Eq. 2.20 we find the following entities:

- $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is the state vector where each $\mathbf{x}_i$ represents one generic state variable. Typically, we are able to divide our state vector into smaller entities which naturally belong together. For instance, in 2D SLAM these variables can model positions of the robot or the location of point features, which either integrate three or two scalar values into one compound.

- $\mathbf{z}_k$ is a measurement that depends on the state variables in $\mathbf{x}_k$. Given a configuration of $\mathbf{x}_k$, an expected value $\hat{\mathbf{z}}_k = \mathbf{h}_k(\mathbf{x}_k)$ for the measurement can be computed by a sensor model $\mathbf{h}_k(\mathbf{x}_k)$. We assume that the error in the measurement is Gaussian. Hence, the uncertainty of the measurement $\mathbf{z}_k$ is modeled by its information matrix $\Omega_k^{\mathbf{z}}$ which is the inverse of the covariance of the Gaussian distribution: $\Sigma_k^{\mathbf{z}} = \left(\Omega_k^{\mathbf{z}}\right)^{-1}$.

- $\mathbf{e}_k(\mathbf{x}_k, \mathbf{z}_k)$ is an error function that computes the difference between the expected measurement $\hat{\mathbf{z}}_k$ and a real measurement $\mathbf{z}_k$. This error is $\mathbf{0}$ when the prediction obtained by mapping the state $\mathbf{x}_k$ to the measurements is equal to the real measurement: $\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k)$. A straightforward error function is the vector difference between the prediction and the measurement: $\mathbf{e}(\mathbf{x}_k, \mathbf{z}_k) = \mathbf{h}_k(\mathbf{x}_k) - \mathbf{z}_k$, but as we will see later other choices are possible.

- Ω_k is the information matrix. Again, we assume a Gaussian distribution for the error and the covariance is $\Sigma_k = \Omega_k^{-1}$ that models the uncertainty of the *error*. The uncertainty of the error depends on the measurement function and on the measurement uncertainty represented by $\Omega_k^{\mathbf{z}}$. For example, if our error function is the vector difference between prediction and measurement, then $\Omega_k = \Omega_k^{\mathbf{z}}$.

- F_k is a factor that models a measurement depending on a subset $\mathbf{x}_k = (\mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_q}) \subset (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ of the state variables.

For simplicity of notation, in the rest of this chapter we will encode the measurement in the indices of the error function as

$$\mathbf{e}(\mathbf{x}_k, \mathbf{z}_k) \overset{\text{def.}}{=} \mathbf{e}_k(\mathbf{x}) \overset{\text{def.}}{=} \mathbf{e}_k. \tag{2.21}$$

The overall goal of least squares minimization is to determine the state vector $\mathbf{x}^+$ that best fits our set of observations, i.e., we seek for the solution of

$$\mathbf{x}^+ = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}). \tag{2.22}$$

Obtaining the solution $\mathbf{x}^+$ for Eq. 2.22 is in general hard. Hence, we restrict ourselves to determine a local minimizer $\mathbf{x}^*$ which satisfies the condition

$$\forall \mathbf{x} : \|\mathbf{x}^* - \mathbf{x}\| < \varepsilon \Rightarrow F(\mathbf{x}^*) \leq F(\mathbf{x}). \tag{2.23}$$

Useful for this endeavor is the gradient of F

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = 2 \sum_{k \in \mathcal{G}} J_k^\top \Omega_k \mathbf{e}_k, \tag{2.24}$$

where $J_k = \frac{\partial \mathbf{e}_k(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of the error function. Gradient descent methods [78, 164] determine a step in the opposite direction of the gradient — the steepest descent — such that the function value is reduced.

Within the literature, we can often find a different equation, namely

$$\acute{F}(\mathbf{x}) = \frac{1}{2}\sum_k \|\acute{\mathbf{e}}_k\|^2 \tag{2.25}$$

$$= \frac{1}{2}\sum_k \acute{\mathbf{e}}_k(\mathbf{x})^\top \acute{\mathbf{e}}_k(\mathbf{x}). \tag{2.26}$$

In fact, the formulations in Eq. 2.20 and Eq. 2.26 are equivalent. First, we note that the constant factor of $\frac{1}{2}$ does not affect the value of $\mathbf{x}^+$. Since $\Omega_k = \Sigma_k^{-1}$ is the inverse of a covariance matrix, Ω_k is also positive definite and symmetric [67]. Consequently, Ω_k can be factored into a matrix C_k such that

$$\Omega_k = C_k^\top C_k. \tag{2.27}$$

The decomposition given in Eq. 2.27 is known as Cholesky decomposition and C_k is upper triangular. Setting $\acute{\mathbf{e}}_k(\mathbf{x}) = C_k \mathbf{e}_k(\mathbf{x})$ proves the equivalence of the two formulations with respect to the optimal value $\mathbf{x}^+$. We prefer the formulation given in Eq. 2.20 over the variant given in Eq. 2.26 because it highlights the probabilistic nature of the underlying estimation.

2.3.1 Least-Squares Optimization

Given an initial value $\breve{\mathbf{x}}$ for the parameters, we are able to rewrite the sensor model by its Taylor expansion

$$\mathbf{h}_k(\breve{\mathbf{x}}_k + \Delta\mathbf{x}) = \mathbf{h}_k(\breve{\mathbf{x}}_k) + J_k\Delta\mathbf{x} + \mathscr{O}\left(\|\Delta\mathbf{x}\|^2\right), \tag{2.28}$$

where

$$J_k = \left.\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}} \tag{2.29}$$

is the Jacobian of the sensor model used to compute the expected measurement. If we assume that either the Jacobian is a sufficient approximation of the sensor model or that $\Delta\mathbf{x}$ is typically small, we can neglect the higher order terms from Eq. 2.28, which yields

$$\mathbf{h}_k(\breve{\mathbf{x}}_k + \Delta\mathbf{x}) \approx \mathbf{h}_k(\breve{\mathbf{x}}_k) + J_k\Delta\mathbf{x}. \tag{2.30}$$

Inserting this Taylor approximation into Eq. 2.20 leads to a linear approximation of the error function, namely

$$\mathbf{e}_k(\breve{\mathbf{x}} + \Delta\mathbf{x}) = \mathbf{h}_k(\breve{\mathbf{x}} + \Delta\mathbf{x}) - \mathbf{z}_k \tag{2.31}$$

$$\approx \mathbf{h}_k(\breve{\mathbf{x}}) + J_k\Delta\mathbf{x} - \mathbf{z}_k \tag{2.32}$$

$$= \mathbf{e}_k + J_k\Delta\mathbf{x}. \tag{2.33}$$

Substituting Eq. 2.33 in the error terms F_k of Eq. 2.20, we obtain

$$F_k(\breve{\mathbf{x}} + \Delta\mathbf{x}) = \mathbf{e}_k(\breve{\mathbf{x}} + \Delta\mathbf{x})^\top \Omega_k \mathbf{e}_k(\breve{\mathbf{x}} + \Delta\mathbf{x}) \tag{2.34}$$

$$\overset{\text{Eq. 2.33}}{\approx} (\mathbf{e}_k + J_k\Delta\mathbf{x})^\top \Omega_k (\mathbf{e}_k + J_k\Delta\mathbf{x}) \tag{2.35}$$

$$= \underbrace{\mathbf{e}_k^\top \Omega_k \mathbf{e}_k}_{\mathrm{c}_k} + 2\underbrace{\mathbf{e}_k^\top \Omega_k J_k}_{\mathbf{b}_k^\top}\Delta\mathbf{x} + \Delta\mathbf{x}^\top \underbrace{J_k^\top \Omega_k J_k}_{H_k}\Delta\mathbf{x} \tag{2.36}$$

$$= \mathrm{c}_k + 2\mathbf{b}_k^\top\Delta\mathbf{x} + \Delta\mathbf{x}^\top H_k\Delta\mathbf{x}. \tag{2.37}$$

With this linear approximation, we can rewrite the function $F(\mathbf{x})$ given in Eq. 2.20 as

$$
\begin{aligned}
F(\check{\mathbf{x}}+\Delta\mathbf{x}) &= \sum_{k\in\mathcal{G}} F_k(\check{\mathbf{x}}+\Delta\mathbf{x}) && (2.38)\\
&\approx \sum_{k\in\mathcal{G}} \left(\mathrm{c}_k + 2\mathbf{b}_k^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H_k \Delta\mathbf{x}\right) && (2.39)\\
&= \underbrace{\sum_{k\in\mathcal{G}} \mathrm{c}_k}_{\mathrm{c}} + 2\underbrace{\sum_{k\in\mathcal{G}} \mathbf{b}_k^\top}_{\mathbf{b}^\top}\Delta\mathbf{x} + \Delta\mathbf{x}^\top \underbrace{\sum_{k\in\mathcal{G}} H_k}_{H}\Delta\mathbf{x} && (2.40)\\
&= \mathrm{c} + 2\mathbf{b}^\top\Delta\mathbf{x} + \Delta\mathbf{x}^\top H\Delta\mathbf{x} && (2.41)\\
&=: G(\Delta\mathbf{x}). && (2.42)
\end{aligned}
$$

In Eq. 2.41 we see that this local approximation leads to a function $G(\Delta\mathbf{x})$, which is quadratic in $\Delta\mathbf{x}$. Furthermore, $G(\Delta\mathbf{x})$ is the linear approximation of $F(\check{\mathbf{x}}+\Delta\mathbf{x})$. Hence, we need to determine $\Delta\mathbf{x}^*$ such that

$$\Delta\mathbf{x}^* = \operatorname*{argmin}_{\Delta\mathbf{x}} G(\Delta\mathbf{x}). \tag{2.43}$$

To determine $\Delta\mathbf{x}^*$ let us look at the first and second derivative of $G(\Delta\mathbf{x})$:

$$
\begin{aligned}
G'(\Delta\mathbf{x}) &= 2H\Delta\mathbf{x} + 2\mathbf{b} && (2.44)\\
G''(\Delta\mathbf{x}) &= 2H. && (2.45)
\end{aligned}
$$

As we can see, the second derivative does not depend on $\Delta\mathbf{x}$. Additionally, H is symmetric and positive definite (see Section 2.3.3). Therefore, we are able to determine $\Delta\mathbf{x}^*$ by setting the first derivative to zero, i.e., find $\Delta\mathbf{x}^*$ such that $G'(\Delta\mathbf{x}^*) = 0$. This leads to the linear system

$$H\,\Delta\mathbf{x}^* = -\mathbf{b}. \tag{2.46}$$

Solving Eq. 2.46 yields the increments $\Delta\mathbf{x}^*$, which are added to the initial value

$$\mathbf{x}^* = \check{\mathbf{x}} + \Delta\mathbf{x}^*. \tag{2.47}$$

In general, however, due to our linear approximation of $F(\mathbf{x})$, which we introduced for computing the increments $\Delta\mathbf{x}^*$, we have not yet reached a local minimum. Our solution $\Delta\mathbf{x}^*$ is the best increment given the linearization of the nonlinear function $F(\mathbf{x})$. The approximation errors result from restricting the Taylor series of $F(\mathbf{x})$ to only include the first term (see Eq. 2.30). To overcome this issue, the typical solution is to iterate the linearization in Eq. 2.41, the solution in Eq. 2.46, and the update step in Eq. 2.47 until some given criterion is matched. Clearly, if $F(\mathbf{x})$ is a linear function, one iteration is sufficient. In every iteration, the previous solution is used as the linearization point. In the following, we will describe common algorithms that follow the outlined schema.

The Gauss-Newton algorithm [178, §15.5] is a popular method, which iterates the individual steps outlined above, see Algorithm 1 for the details. In every iteration, the previous solution serves as the linearization point and the initial guess. The iterations are stopped when a given termination criterion is matched. There is in general no guarantee that the algorithm converges, though. This depends on a number of factors:

- How far is our initial guess $\check{\mathbf{x}}$ from the optimal value $\mathbf{x}^*$?

Algorithm 1 Gauss-Newton minimization algorithm

Input: $\breve{\mathbf{x}}$: initial guess. $\mathcal{G} = \{\langle \mathbf{z}_k, \Omega_k \rangle\}$: measurements
Output: $\mathbf{x}^*$: new solution

1: $i \leftarrow 0$
2: **repeat**
3: $\quad i \leftarrow i+1$
4: $\quad F_{\text{old}} \leftarrow F(\breve{\mathbf{x}})$
5: $\quad$ **for all** $k \in \mathcal{G}$ **do**
6: $\quad\quad$ // Compute the error $\mathbf{e}_k$ and the Jacobian J_k
7: $\quad\quad \mathbf{e}_k \leftarrow \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k$
8: $\quad\quad J_k \leftarrow \left.\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}}$
9: $\quad\quad$ // compute the contribution of this measurement to the linear system
10: $\quad\quad H_k \leftarrow J_k^\top \Omega_k J_k$
11: $\quad\quad \mathbf{b}_k^\top \leftarrow \mathbf{e}_k^\top \Omega_k J_k$
12: $\quad$ **end for**
13: $\quad$ // construct the overall system
14: $\quad H \leftarrow \sum_{k \in \mathcal{G}} H_k$
15: $\quad \mathbf{b} \leftarrow \sum_{k \in \mathcal{G}} \mathbf{b}_k$
16: $\quad$ // solve the linear system
17: $\quad \Delta\mathbf{x}^* \leftarrow \text{solve}(H\,\Delta\mathbf{x} = -\mathbf{b})$
18: $\quad$ // update the state
19: $\quad \breve{\mathbf{x}} \leftarrow \breve{\mathbf{x}} + \Delta\mathbf{x}^*$
20: $\quad$ // compute the new error
21: $\quad F_{\text{new}} \leftarrow F(\breve{\mathbf{x}})$
22: **until** $(F_{\text{old}} - F_{\text{new}} < \varepsilon) \vee \left(i > i_{\max}\right)$
23: **return** $\breve{\mathbf{x}}$

- How smooth are the error functions $\mathbf{e}_k(\cdot)$ or the sensor model $\mathbf{h}_k(\cdot)$?
- How to handle the case in which the matrix H does not have full rank?

Despite these concerns, the method is the building block for other algorithms to address nonlinear least squares. While we address the first two concerns in Chapter 3, there exists an adaption of the Gauss-Newton algorithm that deals with the last point and furthermore guarantees the convergence to a local minimum as defined in Eq. 2.23.

For this purpose, the Levenberg-Marquardt algorithm [178, §15.5] introduces a damping factor as well as backup and restore actions to the Gauss-Newton method. This controls the convergence and ensures that H has full rank, see also Algorithm 2 for the details. Instead of solving Eq. 2.46, Levenberg-Marquardt solves a damped version:

$$(H + \lambda I)\,\Delta\mathbf{x}^*_{\text{lm}} \quad = \quad -\mathbf{b}. \tag{2.48}$$

Here, $\lambda \geq 0$ is the damping factor, whose value is controlled by the Levenberg-Marquardt algorithm. To this end, we monitor the value of the error in each iteration. The control strategy descreases λ for the next iteration if the current step reduces the error. On the other hand, if the error is not reduced, we recover the previous state vector and increase λ. This part of the procedure is given between Line 14 and Line 26 in Algorithm 2. If we are not able to determine a good step, i.e., a step which results in a reduction of the error, within a certain number of iterations (given by the parameter $j_{\max}$), we bail out and give up. Note that other strategies to

Algorithm 2 Levenberg-Marquardt minimization algorithm

Input: $\breve{\mathbf{x}}$: initial guess. $\mathcal{G} = \{\langle \mathbf{z}_k(\cdot), \Omega_k \rangle\}$: measurements
Output: $\mathbf{x}^*$: new solution

1: $\lambda \leftarrow \text{computeInitialLambda}(\mathcal{G}, \breve{\mathbf{x}})$
2: **repeat**
3: **for all** $k \in \mathcal{G}$ **do**
4: $\mathbf{e}_k \leftarrow \mathbf{h}_k(\breve{\mathbf{x}}) - \mathbf{z}_k$
5: $J_k \leftarrow \left.\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\breve{\mathbf{x}}}$
6: $H_k \leftarrow J_k^\top \Omega_k J_k$
7: $\mathbf{b}_k^\top \leftarrow \mathbf{e}_k^\top \Omega_k J_k$
8: **end for**
9: $H \leftarrow \sum_{k \in \mathcal{G}} H_k$
10: $\mathbf{b} \leftarrow \sum_{k \in \mathcal{G}} \mathbf{b}_k$
11: $j \leftarrow 0$
12: $\mathbf{x}_{\text{backup}} \leftarrow \breve{\mathbf{x}}$
13: $\breve{F}_{\text{old}} \leftarrow F(\breve{\mathbf{x}})$
14: **while** $j < j_{\max} \wedge j \geq 0$ **do**
15: $\Delta\mathbf{x}^*_{\text{lm}} \leftarrow \text{solve}((H + \lambda I)\Delta\mathbf{x} = \mathbf{b})$
16: $\breve{\mathbf{x}} \leftarrow \breve{\mathbf{x}} + \Delta\mathbf{x}^*_{\text{lm}}$
17: $F_{\text{new}} \leftarrow F(\breve{\mathbf{x}})$
18: **if** $F_{\text{new}} < \breve{F}_{\text{old}}$ **then**
19: $\lambda \leftarrow \lambda / 2$
20: $j \leftarrow -1$
21: **else**
22: $\lambda \leftarrow \lambda \cdot 4$
23: $\breve{\mathbf{x}} = \mathbf{x}_{\text{backup}}$
24: $j \leftarrow j + 1$
25: **end if**
26: **end while**
27: **until** $\left(\breve{F}_{\text{old}} - F_{\text{new}} < \varepsilon\right) \vee \left(j = j_{\max}\right)$
28: **return** $\breve{\mathbf{x}}$

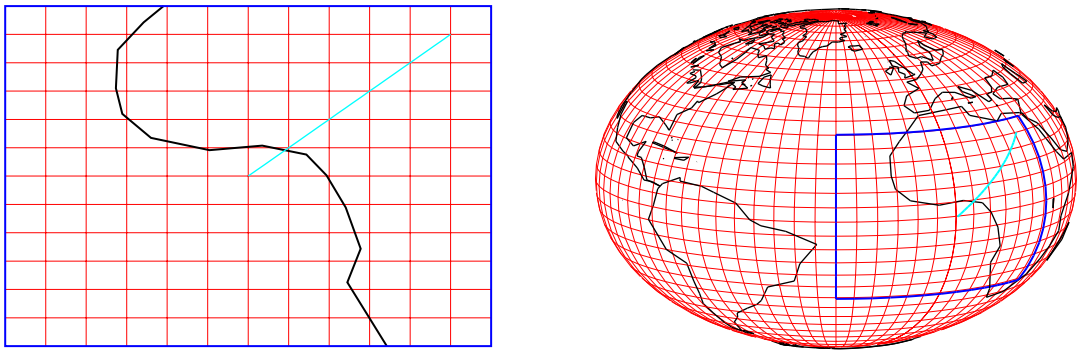
control the damping parameter λ are possible. These modifications ensure that, compared to the Gauss-Newton method, we are able to find a local minimum. Furthermore, we can handle situations in which H does not have full rank. To compute the initial value of λ within the first line of Algorithm 2, we follow the strategy suggested by Lourakis *et al.* [140].

In situations where $\lambda = 0$, the increments determined by the Levenberg-Marquardt algorithm are identical to the increments in the Gauss-Newton method. In contrast to this, large values for λ result in smaller increments and

$$\Delta\mathbf{x}^*_{\text{lm}} \approx -\frac{1}{\lambda}\mathbf{b} \tag{2.49}$$

$$= -\frac{1}{\lambda}\sum_{k \in \mathcal{G}} J_k^\top \Omega_k \mathbf{e}_k. \tag{2.50}$$

Hence, the step $\Delta\mathbf{x}^*_{\text{lm}}$ computed by the Levenberg-Marquardt algorithm for a large λ is a small step along the steepest descent direction (see Eq. 2.24) of the objective function which is useful in situations where $F(\mathbf{x})$ exhibits an irregular surface and we are far from a local minimum.



(a) 2D chart (b) terrestrial globe

Figure 2.7: Example for the $\boxplus$ operator on a sphere. In the particular example, the mapping is performed between a chart and the terrestrial globe. (a) A 2D chart with a partial map of the earth. (b) Here, the $\boxplus$ operator maps the cyan vector, which is a member of $\mathbb{R}^2$, to the unit sphere.

As λ varies in between those two extremes, we typically obtain a step which is a mixture of a Gauss-Newton step and a step along the steepest descent direction.

2.3.2 Alternative Parameterizations

The procedures described above are general approaches to multivariate function minimization. They assume that the space of the parameters $\mathbf{x}$ is Euclidean, which is not valid for several problems like SLAM or Bundle Adjustment [210]. In these problems, the state includes the angular components of the pose of the robot, which are not a Euclidean space but are either elements of the non-Euclidean 2D or 3D rotation group $SO(2)$ or $SO(3)$. To deal with state variables that span over non-Euclidean space, a common approach is to express the increments $\Delta\mathbf{x}_k$ in a space different from the one of the parameters $\mathbf{x}_k$.

For example, in the context of the SLAM problem, each parameter block $\mathbf{x}_k$ consists of a translation vector $\mathbf{t}_k$ and a rotational component α_k. The translation $\mathbf{t}_k$ clearly forms a Euclidean space. In contrast to that, the rotational components α_k span over the non-Euclidean 2D or 3D rotation group $SO(2)$ or $SO(3)$. To avoid singularities, these spaces are usually described in an over-parameterized way, e.g., by rotation matrices or quaternions. Directly applying Eq. 2.47 to these over-parameterized representations breaks the constraints induced by the over-parameterization, for example, the orthogonality of the rotation matrix or the norm of the quaternion. One way to address this issue is to consider a minimal representation, such as Euler angles for a rotation in 3D. This is suboptimal as such a representation is subject to singularities.

An alternative idea is to compute a new error function where $\Delta\mathbf{x}_k$ are perturbations around the current variable $\breve{\mathbf{x}}_k$. In particular, $\Delta\mathbf{x}_k$ uses a minimal representation for the rotations, whereas $\mathbf{x}_k$ utilizes an over-parameterized one. Since the $\Delta\mathbf{x}_k$ are usually small in each step of the iterative optimization procedure, they are far from the singularities. The new value of a variable $\mathbf{x}_k^*$ after the optimization can be obtained by applying the increment through a nonlinear operator as suggested in [88, 130], $\boxplus : \mathtt{Dom}(\mathbf{x}_k) \times \mathtt{Dom}(\Delta\mathbf{x}_k) \rightarrow \mathtt{Dom}(\mathbf{x}_k)$ as follows:

$$\mathbf{x}_k^* \quad = \quad \breve{\mathbf{x}}_k \boxplus \Delta\mathbf{x}_k^*. \tag{2.51}$$

Figure 2.7 illustrates the mapping between a plane and a unit sphere. The example illustrates the projection between a chart and the terrestrial globe. Furthermore, the operator $\boxminus$ computes the difference while accounting for the different domains of the involved variables. Using this

operator, we are able to rewrite the error function $\mathbf{e}(\cdot)$ as

$$\tilde{\mathbf{e}}_k(\check{\mathbf{x}}) = \mathbf{h}_k(\check{\mathbf{x}}) \boxminus \mathbf{z}_k. \tag{2.52}$$

Furthermore, the information matrix Ω_k can be computed by taking into account the uncertainty of the measurement $\Omega_k^{\mathbf{z}}$, for example, by

$$\Omega_k = \left(J_{\mathbf{z}_k} (\Omega_k^{\mathbf{z}})^{-1} J_{\mathbf{z}_k}^\top \right)^{-1}, \tag{2.53}$$

where

$$J_{\mathbf{z}_k} = \left. \frac{\partial \mathbf{h}_k(\check{\mathbf{x}}) \boxminus \mathbf{z}_k}{\partial \mathbf{z}_k} \right|_{\mathbf{z}_k = \mathbf{0}}. \tag{2.54}$$

Clearly, applying the unscented transform [96] is also possible for propagating the covariance through the function instead of the above given linearization method. As the Jacobian depends on $\check{\mathbf{x}}$, we need to perform this operation in every iteration if Ω_k describing the uncertainty of the error is not available by other means.

For instance, in case of 3D SLAM we may represent the increments $\Delta\mathbf{x}_k$ by the translation vector and the rotation vector representation. In this representation the 3D vector itself describes the rotation axis and its length gives the angle with which we have to rotate around the axis. In contrast, the pose $\mathbf{x}_k$ may internally be stored as the elements of a 4×4 isometry matrix. We will denote this by using the variable X_k. An isometry is a special case of a homogeneous transformation matrix that preserves the distance between vectors when mapping them through the matrix. In particular, let us introduce the function $\mathrm{toVec}(\cdot)$ which extracts the translation vector and the rotation vector representation for a given isometry matrix. Additionally, the function $\mathrm{fromVec}(\cdot)$ performs the inverse operation. Using this, we can implement the $\boxplus$ and $\boxminus$ operator as

$$\check{\mathbf{x}}_k \boxplus \Delta\mathbf{x}_k^* \stackrel{\text{def.}}{=} \check{X}_k \,\mathrm{fromVec}\left(\Delta\mathbf{x}_k^*\right) \tag{2.55}$$

$$\mathbf{h}(\check{\mathbf{x}}) \boxminus \mathbf{z}_k \stackrel{\text{def.}}{=} \mathrm{toVec}\left(\mathbf{z}_k^{-1} \,\mathrm{fromVec}\left(\mathbf{h}(\check{\mathbf{x}})\right)\right). \tag{2.56}$$

For SLAM the operator $\boxplus$ is closely related to the standard motion composition operator $\oplus$ which was originally introduced by Smith *et al.* [193], whereas $\boxplus$ additionally takes into account the different representations of the state and the increment.

Since we will use the operator $\oplus$ and its inverse $\ominus$ several times throughout this thesis, we give a possible definition. If the pose $\mathbf{x}_i$ is represented by the isometry matrix X_i, we can specify the operator $\ominus$ as

$$M_{ij} = \mathbf{x}_j \ominus \mathbf{x}_i \stackrel{\text{def.}}{=} X_i^{-1} X_j, \tag{2.57}$$

where M_{ij} is the isometry matrix describing the relative motion between the pose $\mathbf{x}_i$ and the pose $\mathbf{x}_j$. On the other hand, the operator $\oplus$ compounds the relative motion as follows:

$$\mathbf{x}_j = \mathbf{x}_i \oplus M_{ij} \stackrel{\text{def.}}{=} X_i M_{ij}. \tag{2.58}$$

Similar to replacing $+$ by $\boxplus$, we can update the Taylor expansion of the error function to account for the newly introduced operators. This leads to

$$\tilde{\mathbf{e}}_k(\check{\mathbf{x}} \boxplus \Delta\mathbf{x}) \approx \mathbf{e}_k + \tilde{J}_k \Delta\mathbf{x}, \tag{2.59}$$

Algorithm 3 Gauss-Newton minimization algorithm with alternative parameterizations

Input: $\breve{\mathbf{x}}$: initial guess. $\mathcal{G} = \{\langle \mathbf{z}_k, \Omega_k \rangle\}$: measurements
Output: $\mathbf{x}^*$: new solution

$k \leftarrow 0$
repeat
  $k \leftarrow k+1$
  $F_{\text{old}} \leftarrow F(\breve{\mathbf{x}})$
  for all $k \in \mathcal{G}$ **do**
    $\mathbf{e}_k \leftarrow \mathbf{h}_k(\breve{\mathbf{x}}) \boxminus \mathbf{z}_k$
    $\tilde{J}_k \leftarrow \left. \frac{\partial (\mathbf{h}_k(\mathbf{x} \boxplus \Delta\mathbf{x}) \boxminus \mathbf{z}_k)}{\partial \Delta\mathbf{x}} \right|_{\Delta\mathbf{x}=\mathbf{0}}$
    $H_k \leftarrow \tilde{J}_k^\top \Omega_k \tilde{J}_k$
    $\mathbf{b}_k^\top \leftarrow \mathbf{e}_k^\top \Omega_k \tilde{J}_k$
  end for
  $H \leftarrow \sum_{k \in \mathcal{G}} H_k$
  $\mathbf{b} \leftarrow \sum_{k \in \mathcal{G}} \mathbf{b}_k$
  $\Delta\mathbf{x}^* \leftarrow \text{solve}(H\,\Delta\mathbf{x} = -\mathbf{b})$
  $\breve{\mathbf{x}} \leftarrow \breve{\mathbf{x}} \boxplus \Delta\mathbf{x}^*$
  $F_{\text{new}} \leftarrow F(\breve{\mathbf{x}})$
until $\breve{F}_{\text{old}} - F_{\text{new}} < \varepsilon \vee k > k_{\max}$
return $\breve{\mathbf{x}}$

where the Jacobian $\tilde{J}_k$ becomes

$$\tilde{J}_k = \left. \frac{\partial \tilde{\mathbf{e}}_k(\breve{\mathbf{x}} \boxplus \Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \right|_{\Delta\mathbf{x}=\mathbf{0}}. \tag{2.60}$$

In contrast to Eq. 2.29, the Jacobian $\tilde{J}_k$ considers a fixed linearization point $\breve{\mathbf{x}}$ and gives the gradient according to the increments $\Delta\mathbf{x}$. Hence, $\tilde{J}_k$ represents the influence of the parameters to the error as a function of the increments that are applied to the current linearization point through the $\boxplus$ operator.

Since the increments $\Delta\mathbf{x}^*$ are computed in the local Euclidean surroundings of the initial guess $\breve{\mathbf{x}}$, they need to be re-mapped into the original redundant space by the $\boxplus$ operator, see Eq. 2.51. With this subtle changes, we obtain a variant of the Gauss-Newton and Levenberg-Marquardt algorithm which is able to address — among other problems — SLAM in an efficient and more accurate way. Algorithm 3 gives the adapted Gauss-Newton algorithm handling the different parameterizations of the state and the increments. A modified variant of the Levenberg-Marquardt algorithm can be achieved by applying similar changes to Algorithm 2. Regardless the choice of the parameterization, the structure of the Hessian H is in general preserved.

2.3.3 Structure and Properties of the Linearized System

From Eq. 2.36 and Eq. 2.41, we see that the matrix H and the vector $\mathbf{b}$ are obtained by summing up a set of matrices and vectors, respectively. Each constraint will contribute to the system with an addend term. The *structure* of this term depends on the Jacobian of the error function. Let us consider the factor graph illustrated in Figure 2.8. The highlighted factor affects three state variables. In the following, we will use triple indices to indicate this relation. Thus, the Jacobian in Eq. 2.33 of a constraint $\mathbf{e}_{abc}$ between the variable $\mathbf{x}_a$, $\mathbf{x}_b$, and $\mathbf{x}_c$ has the following

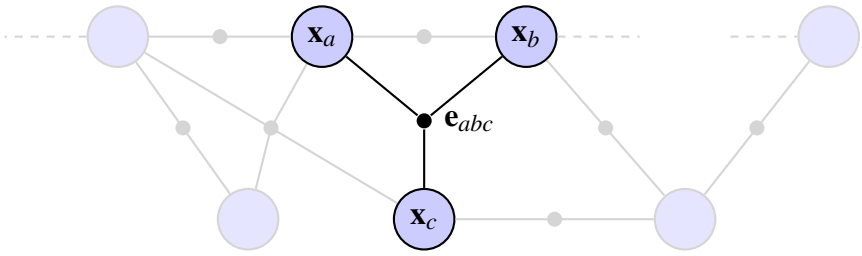


Figure 2.8: Example of a ternary factor. The highlighted factor constraints three state variables. Thus, its Jacobian will have three non-zero blocks that affect the structure of the linearized system.

form:

$$J_{abc} = \begin{pmatrix} \mathbf{0} \cdots \mathbf{0} \underbrace{A}_{\frac{\partial \mathbf{e}_{abc}}{\partial \mathbf{x}_a}} \mathbf{0} \cdots \mathbf{0} \underbrace{B}_{\frac{\partial \mathbf{e}_{abc}}{\partial \mathbf{x}_b}} \mathbf{0} \cdots \mathbf{0} \underbrace{C}_{\frac{\partial \mathbf{e}_{abc}}{\partial \mathbf{x}_c}} \mathbf{0} \cdots \mathbf{0} \end{pmatrix}^\top . \tag{2.61}$$

Here A, B, and C are the derivatives of the error function with respect to $\mathbf{x}_a$, $\mathbf{x}_b$, and $\mathbf{x}_c$, respectively, i.e., the blocks in our state vector $\mathbf{x}$ corresponding to the entities involved in the error function. From Eq. 2.35, we obtain the following structure for the block matrix H_{abc} and the vector $\mathbf{b}_{abc}$:

$$H_{abc} = \begin{pmatrix} \ddots & & & & & & \iddots \\ & A^\top \Omega_{abc} A & \cdots & A^\top \Omega_{abc} B & \cdots & A^\top \Omega_{abc} C & \\ & \vdots & & \vdots & & \vdots & \\ & B^\top \Omega_{abc} A & \cdots & B^\top \Omega_{abc} B & \cdots & B^\top \Omega_{abc} C & \\ & \vdots & & \vdots & & \vdots & \\ & C^\top \Omega_{abc} A & \cdots & C^\top \Omega_{abc} B & \cdots & C^\top \Omega_{abc} C & \\ \iddots & & & & & & \ddots \end{pmatrix} \tag{2.62}$$

$$\mathbf{b}_{abc} = \begin{pmatrix} \cdots \mathbf{e}_{abc}^\top \Omega_{abc} A \cdots \mathbf{e}_{abc}^\top \Omega_{abc} B \cdots \mathbf{e}_{abc}^\top \Omega_{abc} C \cdots \end{pmatrix}^\top . \tag{2.63}$$

For simplicity of notation, we omitted the zero blocks. The reader might notice that the block structure of the matrix H is the adjacency matrix of the graph. Thus, it has a number of non-zero blocks proportional to the number of edges in the graph. This typically results in a sparse matrix H.

Furthermore, let us state some properties of the matrix H, which will prove useful for us. From Eq. 2.36 and Eq. 2.41, we see that

$$H = \sum_{k \in \mathcal{G}} J_k^\top \Omega_k J_k \tag{2.64}$$

$$= \sum_{k \in \mathcal{G}} (\Omega_k J_k)^\top \left(J_k^\top \right)^\top \tag{2.65}$$

$$= \sum_{k \in \mathcal{G}} \left(J_k^\top \Omega_k J_k \right)^\top \tag{2.66}$$

$$= \left(\sum_{k \in \mathcal{G}} J_k^\top \Omega_k J_k \right)^\top = H^\top . \tag{2.67}$$

Thus, H is a symmetric matrix due to its construction. In Eq. 2.66, we exploit the fact that the information matrix $\Omega_k = \Omega_k^\top$ is symmetric. Additionally, we observe that

$$\begin{aligned}
\Delta\mathbf{x}^\top H \Delta\mathbf{x} &= \Delta\mathbf{x}^\top \sum_{k\in\mathscr{G}} (H_k)\,\Delta\mathbf{x} && (2.68)\\
&= \sum_{k\in\mathscr{G}} \Delta\mathbf{x}^\top J_k^\top \Omega_k J_k \Delta\mathbf{x} && (2.69)\\
&= \sum_{k\in\mathscr{G}} \Delta\mathbf{x}^\top J_k^\top C_k^\top \underbrace{C_k J_k \Delta\mathbf{x}}_{\mathbf{y}_k} && (2.70)\\
&= \sum_{k\in\mathscr{G}} \mathbf{y}_k^\top \mathbf{y}_k \geq 0. && (2.71)
\end{aligned}$$

Hence, H is positive semi-definite. Note that within the derivation we considered the decomposition of $\Omega_k = C_k^\top C_k$ as given in Eq. 2.27.

2.3.4 Systems Having Special Structure

Certain problems, for instance, Bundle Adjustment (BA) [210], result in a matrix H that has an even more characteristic structure. Our system can take advantage of these special structures to improve the performance. In BA there are in general two types of variables, namely the poses $\mathbf{p}$ of the camera and the poses $\mathbf{l}$ of the landmarks observed by the camera. By reordering the variables in Eq. 2.46 so that the camera poses have the lower indices, we obtain the system

$$\begin{pmatrix} H_{\mathbf{pp}} & H_{\mathbf{pl}} \\ H_{\mathbf{pl}}^\top & H_{\mathbf{ll}} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}_{\mathbf{p}}^* \\ \Delta\mathbf{x}_{\mathbf{l}}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{b}_{\mathbf{p}} \\ -\mathbf{b}_{\mathbf{l}} \end{pmatrix}. \tag{2.72}$$

It can be shown that an equivalent reduced system is formed by taking the so-called Schur complement of the matrix H [64]:

$$\left(H_{\mathbf{pp}} - H_{\mathbf{pl}} H_{\mathbf{ll}}^{-1} H_{\mathbf{pl}}^\top\right) \Delta\mathbf{x}_{\mathbf{p}}^* = -\mathbf{b}_{\mathbf{p}} + H_{\mathbf{pl}} H_{\mathbf{ll}}^{-1} \mathbf{b}_{\mathbf{l}}. \tag{2.73}$$

Note that calculating $H_{\mathbf{ll}}^{-1}$ is easy. Since $H_{\mathbf{ll}}$ is a block-diagonal matrix, its inverse is given by inverting each block individually. We obtain the equation above by multiplying the bottom block-row of Eq. 2.72 with $H_{\mathbf{pl}} H_{\mathbf{ll}}^{-1}$:

$$\begin{pmatrix} H_{\mathbf{pp}} & H_{\mathbf{pl}} \\ H_{\mathbf{pl}} H_{\mathbf{ll}}^{-1} H_{\mathbf{pl}}^\top & H_{\mathbf{pl}} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}_{\mathbf{p}}^* \\ \Delta\mathbf{x}_{\mathbf{l}}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{b}_{\mathbf{p}} \\ -H_{\mathbf{pl}} H_{\mathbf{ll}}^{-1} \mathbf{b}_{\mathbf{l}} \end{pmatrix}. \tag{2.74}$$

Now, subtracting the second block-row from the first results in Eq. 2.73, whose solution yields the increments $\Delta\mathbf{x}_{\mathbf{p}}^*$ for the cameras and using this we can solve

$$H_{\mathbf{ll}} \Delta\mathbf{x}_{\mathbf{l}}^* = -\mathbf{b}_{\mathbf{l}} - H_{\mathbf{pl}}^\top \Delta\mathbf{x}_{\mathbf{p}}^*, \tag{2.75}$$

which results in $\Delta\mathbf{x}_{\mathbf{l}}^*$ for adjusting the observed world features. Typically the world features outnumber the camera poses, therefore Eq. 2.73 can be solved faster than Eq. 2.46 despite the additional time spent to calculate the left-hand side matrix in Eq. 2.73.

2.3.5 Gaussian Conditional: $p(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mathbf{x}^*, H^{-1})$

In this section, we will show that $p(\mathbf{x} \mid \mathbf{z})$ under certain assumptions, which we recall below, corresponds to the Gaussian distribution $\mathcal{N}(\mathbf{x}^*, H^{-1})$. This means we will show that our system matrix H is the information matrix for the state given all the measurements. To this end, let us first recall some concepts which have already been stated above.

- We assume that the measurements $\mathbf{z}_k$ are affected by a Gaussian noise. The noise of each measurement is represented by the information matrix Ω_k.
- We assume that the measurement function $\hat{\mathbf{z}}_k = \mathbf{h}_k(\mathbf{x})$ can be locally approximated by its first-order Taylor expansion around $\mathbf{x}^*$, i.e.,

$$\mathbf{h}_k(\mathbf{x}^* + \Delta\mathbf{x}) \approx \mathbf{h}_k(\mathbf{x}^*) + J_k \Delta\mathbf{x}. \tag{2.76}$$

- We assume that the conditional distribution of the measurements given the increments is normally distributed

$$p(\mathbf{z}_k \mid \mathbf{x}^* + \Delta\mathbf{x}) \sim \mathcal{N}\left(\mathbf{z}_k^* + J_k \Delta\mathbf{x}, \Omega_k^{-1}\right). \tag{2.77}$$

  Here $\mathbf{z}_k^* = \mathbf{h}_k(\mathbf{x}^*)$ is the expected measurement at the optimum.

The conditional over *all* measurements $\mathbf{z}$ is again a multivariate Gaussian

$$p(\mathbf{z} \mid \mathbf{x}^* + \Delta\mathbf{x}) \sim \mathcal{N}\left(\mu_\mathbf{z}, \Omega_\mathbf{z}^{-1}\right), \tag{2.78}$$

where

$$\mu_\mathbf{z} = \begin{pmatrix} J_1\Delta\mathbf{x} + \mathbf{z}_1^* \\ J_2\Delta\mathbf{x} + \mathbf{z}_2^* \\ \vdots \\ J_K\Delta\mathbf{x} + \mathbf{z}_K^* \end{pmatrix} = \underbrace{\begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_K \end{pmatrix}}_{J} \Delta\mathbf{x} + \underbrace{\begin{pmatrix} \mathbf{z}_1^* \\ \mathbf{z}_2^* \\ \vdots \\ \mathbf{z}_K^* \end{pmatrix}}_{\mathbf{z}^*} \tag{2.79}$$

$$\Omega_\mathbf{z} = \begin{pmatrix} \Omega_1 & & & \\ & \Omega_2 & & \\ & & \ddots & \\ & & & \Omega_K \end{pmatrix}. \tag{2.80}$$

If we have no prior about the increments, but we assume that they are normally distributed, we can describe them with $p(\Delta\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \Sigma_\mathbf{x})$, in which $\Sigma_\mathbf{x} = \mathrm{diag}(\infty)$ reflects that we assume to have no prior for the increments. A better way to express this is to set $\Omega_x = \mathbf{0}$. We can express the joint distribution $p(\Delta\mathbf{x}, \mathbf{z})$ as

$$p(\Delta\mathbf{x}, \mathbf{z}) \sim \mathcal{N}\left((\mathbf{0}, \mathbf{z}^*), \Omega_{\mathbf{x},\mathbf{z}}^{-1}\right), \tag{2.81}$$

where $\mathbf{z}^* = (\mathbf{z}_1^*, \ldots, \mathbf{z}_K^*)^\top$ and recalling Theorem 3 of [187] we know that

$$\Omega_{\mathbf{x},\mathbf{z}} = \begin{pmatrix} J^\top \Omega_\mathbf{z} J & -J^\top \Omega_\mathbf{z} \\ -\Omega_\mathbf{z} J & \Omega_\mathbf{z} \end{pmatrix}. \tag{2.82}$$

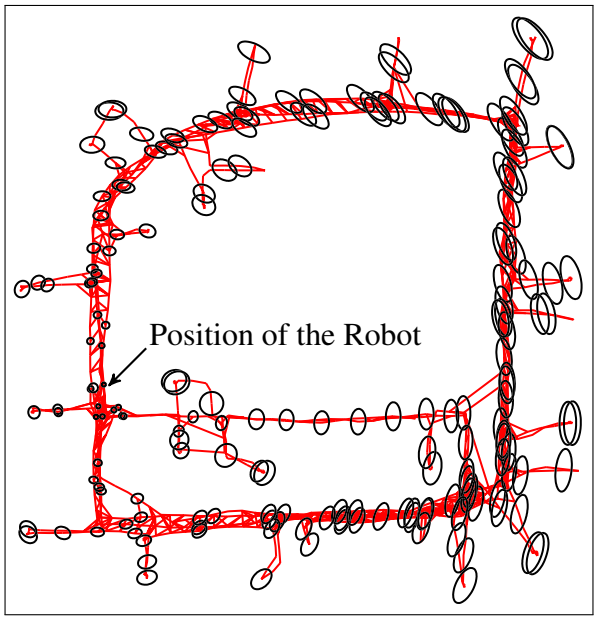


Figure 2.9: The uncertainty of the estimated parameters along with the trajectory in a SLAM data set. Information about the pose uncertainty of the robot is, for example, useful for performing the data association in the front-end.

Let us now consider the upper left block of $\Omega_{\mathbf{x},\mathbf{z}}$:

$$J^\top \Omega_{\mathbf{z}} J = \begin{pmatrix} J_1 & & \\ & J_2 & \\ & & \ddots \\ & & J_K \end{pmatrix}^\top \begin{pmatrix} \Omega_1 & & \\ & \Omega_2 & \\ & & \ddots \\ & & \Omega_K \end{pmatrix} \begin{pmatrix} J_1 & & \\ & J_2 & \\ & & \ddots \\ & & J_K \end{pmatrix} \tag{2.83}$$

$$= J_1^\top \Omega_1 J_1 + J_2^\top \Omega_2 J_2 + \cdots + \underbrace{J_k^\top \Omega_k J_k}_{H_k} + \cdots + J_K^\top \Omega_K J_K \tag{2.84}$$

$$= \sum_{k \in \mathcal{G}} H_k = H. \tag{2.85}$$

Following the derivation so far, we know that H is the upper left block of $\Omega_{\mathbf{x},\mathbf{z}}$, which is the information matrix of the joint Gaussian distribution $p(\Delta\mathbf{x}, \mathbf{z})$ of the increments and the measurements.

Given the joint distribution, we now want to determine the *conditional* distribution $p(\Delta\mathbf{x} \mid \mathbf{z})$. In particular, we are interested in the information matrix $\Omega_{\Delta\mathbf{x}|\mathbf{z}}$. Theorem 2 in [187] tells us how to compute the conditional distribution given the joint distribution. Just by suppressing the rows and columns of the information matrix which correspond to the given variables in the conditional, we obtain the conditional distribution from the joint distribution. In our case, we have to drop the blocks $\Omega_{\mathbf{z}}$, $-J^\top \Omega_{\mathbf{z}}$ and $-\Omega_{\mathbf{z}} J$. Thus, only H remains. This means that $p(\Delta\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mathbf{0}, H^{-1})$. Intuitively, the mean $\mu_{\Delta\mathbf{x}}$ is $\mathbf{0}$ because we assumed $\mathbf{x}^*$ to be our optimal state. As a last step, we introduce the random variable $\mathbf{x} = \mathbf{x}^* + \Delta\mathbf{x}$. Since $p(\Delta\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(0, \Sigma_{\Delta\mathbf{x}|\mathbf{z}})$ and $\mathbf{x}^*$ is a constant, $\mathbf{x}$ has the same covariance as $\Delta\mathbf{x}$, but its mean is shifted by $\mathbf{x}^*$. Hence, we have shown that $p(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mathbf{x}^*, H^{-1})$.

This means, we are able to calculate the covariance of each parameter in our state vector. Such information is, for example, handy for data association [99] or for determining when to stop data-collection because the desired accuracy has been reached. Note that H^{-1} in general

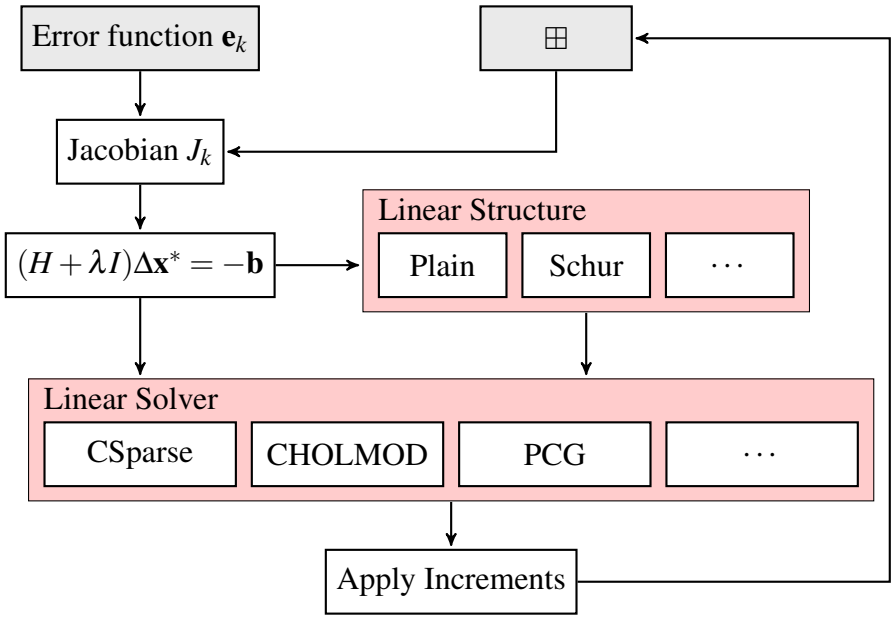


Figure 2.10: Overview of our framework. For addressing a new optimization problem, only the boxes in gray need to be specified. Furthermore, the framework allows us to add different linear solvers.

is not sparse. Based on the sparse Cholesky factorization of H, where we exploit the symmetry and positive definiteness, we are, however, able to evaluate the desired elements of H^{-1} efficiently [70, 99, 210]. We are in particular interested in the elements along the main diagonal of H^{-1} as those correspond to the covariance of our state elements. Figure 2.9 shows an example for the uncertainty of the poses of the robot as they are estimated by our approach in a 2D SLAM example.

2.4 The g^{2}o Framework

Our C++ implementation for solving least squares, which we call g^{2}o (short for General Graph Optimization), aims to be as fast as possible while remaining general. We achieve this goal by implementing abstract base classes for vertices and edges in our graph. Both base classes provide a set of virtual functions for easy sub-classing, whereas most of the internal operations are implemented using template arguments for efficiency. We use the Eigen linear algebra package [81] which applies SSE instructions among other optimization techniques, such as lazy evaluation and loop unrolling to achieve high performance. g^{2}o has been released as open-source software and attracted a large amount of users. We attribute the popularity of g^{2}o to its easy usage, its wide application range due to the general design, and the overall good performance compared to other frameworks.

Figure 2.10 depicts the design of our system. Only the boxes in gray need to be defined to address a new optimization problem. Using the provided base class, deriving a new type of node only requires defining the $\boxplus$ operator for applying the increments. An edge connecting a a subset of nodes $\mathbf{x}_k$ requires the definition of the error function $\mathbf{e}_k(\cdot)$. The Jacobian J_k is then evaluated numerically or, for higher efficiency, the user can specify J_k explicitly by overwriting the virtual base-class function. Optionally, Automatic Differentiation [72] is available, which provides a higher accuracy than evaluating the Jacobian numerically by finite differences while achieving an acceptable runtime. Thus, implementing types for addressing a new optimization problem or comparing different parameterizations is a matter of writing a few lines of code.

The computation of H exploits the underlying block-structure. In the general case each block matrix has a variable size. If the dimension of the variables of the system, i.e., the dimension of $\Delta\mathbf{x}_k$, is known in advance, our framework applies fixed-size matrix computations. Exploiting the a-priori known dimensions enables compile-time optimizations, such as loop unrolling to carry out matrix multiplications.

Special care has been taken in implementing the matrix multiplications required for the Schur reduction in Eq. 2.73. The sparse structure of the underlying graph is exploited to only multiply non-zero entries required to form the extra entries of $H_{\mathbf{pp}}$. Additionally, we operate on the block structures of the underlying matrix (see [107]), which results in a cache efficient matrix multiplication compared to a scalar matrix multiplication.

Our framework is agnostic with respect to the embedded linear solver, so we can apply for each problem the appropriate one. We currently have implemented two different methods for solving a linear system as the one given in Eq. 2.46. The first one exploits the characteristic structure of the system matrix H — namely its typical sparseness — and the two properties stated in Section 2.3.3 (H is symmetric and positive definite). The linear system is solved by utilizing state-of-the-art approaches for computing the Cholesky factor of such a sparse matrix [37, 40]. Cholesky decomposition factorizes the matrix H as

$$H = LL^\top, \tag{2.86}$$

where L is a lower triangle matrix. This allows us to solve Eq. 2.46 by forward and backward substitution as follows

$$L\,\Delta\mathbf{y} = -\mathbf{b} \tag{2.87}$$

$$L^\top \Delta\mathbf{x} = \Delta\mathbf{y}. \tag{2.88}$$

We implemented solvers utilizing CSparse [40], which is a library also suited for embedded systems, and CHOLMOD [37], which results in additional dependencies. As we show in the experiments CHOLMOD, however, outperforms CSparse on larger matrices. Note that the non-zero pattern during the least squares iterations is constant in Gauss-Newton (see Algorithm 1) and Levenberg-Marquardt (see Algorithm 2). We are therefore able to reuse a symbolic decomposition computed within the first iteration. The symbolic decomposition determines a permutation matrix that leads to less fill-in for L and thus reduces the overall computation time in subsequent iterations. Obtaining the best permutation matrix is NP-complete [219]. However, using a heuristic like Approximate Minimum Degree ordering [40] on the block-matrix gives good results [2]. Note that the Cholesky decomposition does not consider the block structure of the parameters apart from the ordering but operates on a scalar matrix.

The second method implemented for solving a large linear system is Preconditioned Conjugate Gradient (PCG) with a block Jacobi pre-conditioner [94], which takes advantage of block matrix operations throughout. PCG itself is an iterative method and solving a linear system requires n iterations for a $n \times n$ matrix. Since carrying out n iterations of PCG is typically slower than Cholesky decomposition [110], we limit the number of iterations based on the relative decrease in the squared residual of PCG. By this, we are able to quantify the loss in the accuracy of the solution introduced by terminating PCG early. In the experiments, we will compare the different solvers implemented in our framework.

It is worth to mention that a variety of different approaches for nonlinear least squares is available which amends the discussed Gauss-Newton and Levenberg-Marquardt algorithms. The alternatives include methods applying gradient descent and approaches based on conjugate gradients such as Polak-Ribière and Fletcher-Reeves [195]. Also variants combining several

techniques have been proposed. For example, Powell's Dog-Leg [146] combines Gauss-Newton and gradient descent and furthermore introduces a modification that guarantees that the length of the step does not exceed the diameter of a trust region around the current linearization point. Due to the modular architecture, these methods can easily be added to the framework.

2.5 Experiments

In this section, we present experiments in which we compare g^{2}o with other state-of-the-art optimization approaches using both real-world and synthetic data sets.

We compare g^{2}o with other state-of-the-art implementations: $\sqrt{\text{SAM}}$ [43] using the open-source implementation by M. Kaess[1], SPA [110][2], sSBA [107][3], RobotVision [201][4], TORO [78][5] and Ceres Solver [3][6]. Note that these approaches except Ceres Solver are only targeting a subset of optimization problems, while g^{2}o is able to solve all of them and also extends easily to new problems. Hence, we implemented our solution to addressing each of the optimization problems as they are implemented by the approaches mentioned above. A notable exception in the list of approaches is the Ceres Solver, which is also an efficient general least squares framework. Here, we compare g^{2}o and Ceres Solver on a Bundle Adjustment problem that includes unknown camera intrinsics [4]. We restrict ourselves to this problem since Ceres Solver itself provides an efficient implementation for this type of problems and our own implementations for other types using Ceres Solver might bias the results.

2.5.1 Real-World Experiments

In a first set of experiments, we evaluate g^{2}o on real-world data. It shows that g^{2}o allows us to generate accurate maps in 2D and 3D as well as given different types of sensor modalities. We show examples using laser or camera data to generate the constraints that serve as input for our approach.

The first one is the MIT CSAIL data set. It was processed with the front-end described in Section 2.2. The resulting factor graph features 686 robot poses and 763 constraints. Figure 2.11a visualizes the odometry of the robot. Employing our optimization framework results in the map depicted in Figure 2.11b. Here, the factor graph is visualized in red. Performing the optimization took around 0.01 s.

As second example, we consider a data set recorded in the Intel research lab located in Seattle. Again, the raw data visualized in Figure 2.12a was processed by the front-end yielding a pose-graph with 1,311 robot poses and 1,824 constraints. By optimizing the factor graph we achieve an estimate for the map which is shown in Figure 2.12b. The least squares optimization is fast, it required less than 0.05 s.

As next data set, we show the results from the DLR data set. In contrast to the MIT CSAIL and the Intel Research Lab, the DLR data set considers camera images to detect white circular disks placed on the floor, which serve as 2D landmarks instead of raw laser data. Additionally, the data association was performed manually by a human instead of running a front-end. Thus, the raw data set may serve as benchmark for data association algorithms. It consists of 3,297

[1] https://svn.csail.mit.edu/isam, Revision 8

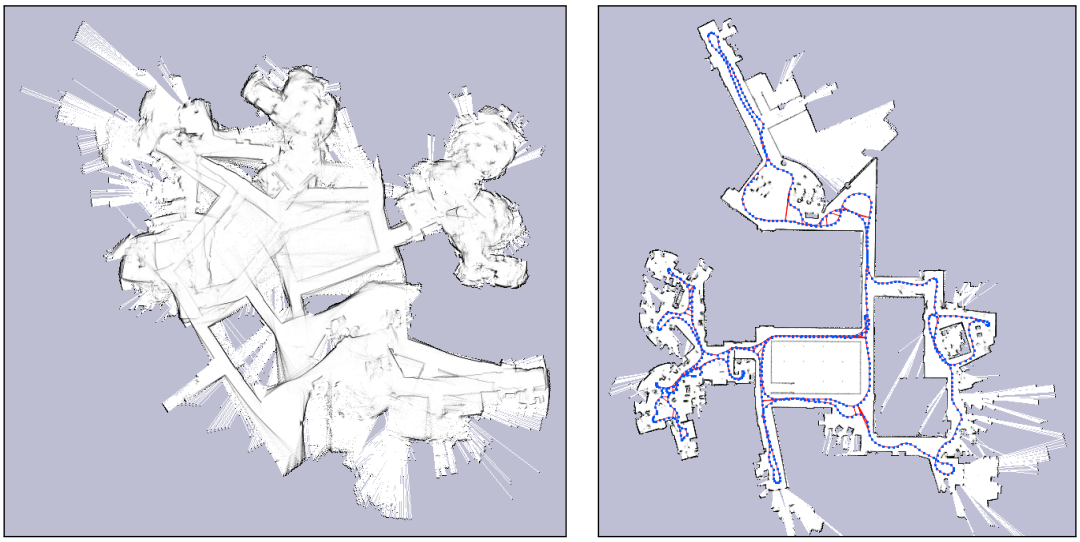
[2] https://code.ros.org/svn/ros-pkg/stacks/vslam/trunk/, Revision 40053

[3] https://code.ros.org/svn/ros-pkg/stacks/vslam/trunk/, Revision 40053

[4] https://www.openslam.org/robotvision, Revision 10

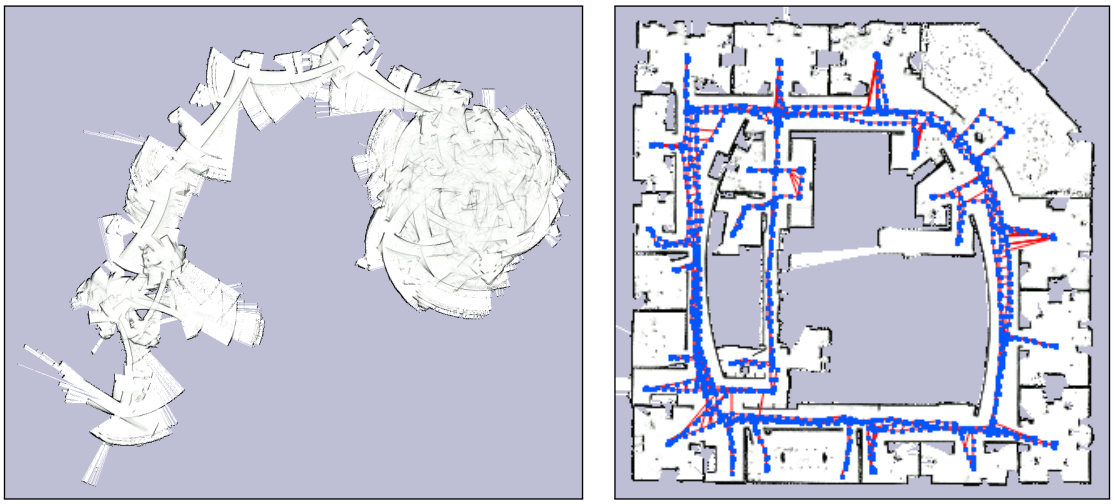
[5] https://www.openslam.org/toro, Revision 19

[6] https://code.google.com/p/ceres-solver/, Version 1.3.0



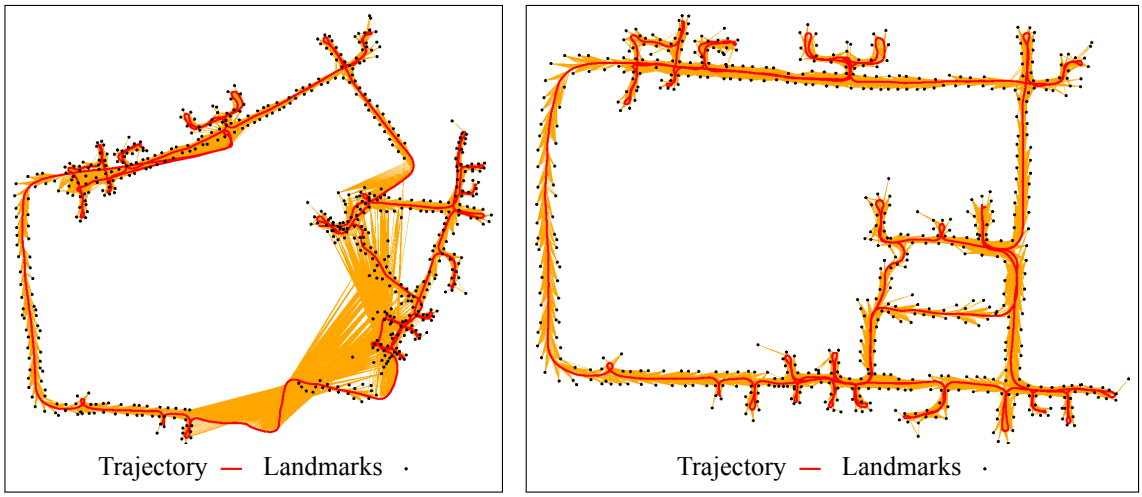
(a) raw odometry (b) estimate of our approach

Figure 2.11: (a) The initial state of the data set as given by the odometry estimate. (b) The map as it is estimated by our approach, in which the nodes of the factor graph are shown in blue and the edges in red.



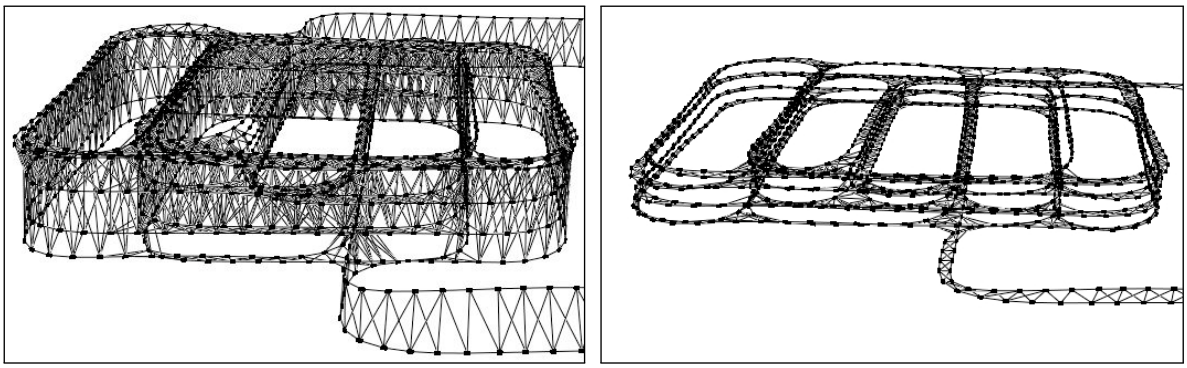
(a) raw odometry (b) estimate of our approach

Figure 2.12: (a) The initial state of the data set as given by the odometry estimate. (b) The map as it is estimated by our approach, in which the nodes of the factor graph are shown in blue and the edges in red.



(a) raw odometry (b) estimate of our approach

Figure 2.13: (a) The initial state of the data set as given by the odometry estimate. (b) The trajectory of the robot after optimizing using our approach.

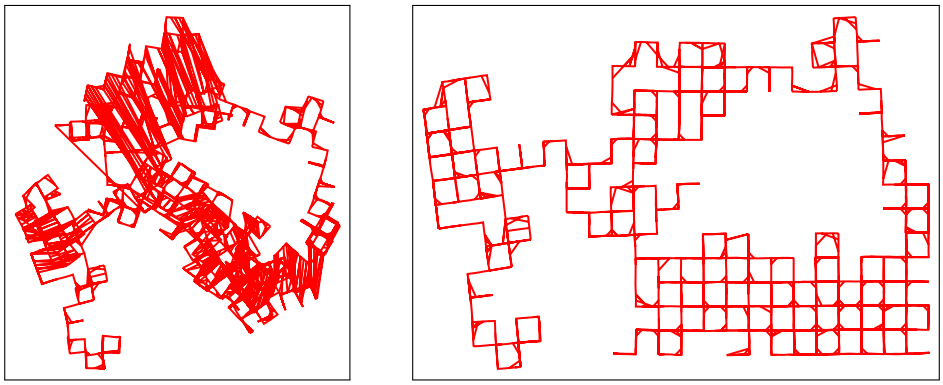


(a) raw odometry (b) estimate of our approach

Figure 2.14: A partial view of the parking garage data set. (a) The initial state of the data set as given by the odometry estimate. (b) The trajectory of the robot after optimizing using our approach.

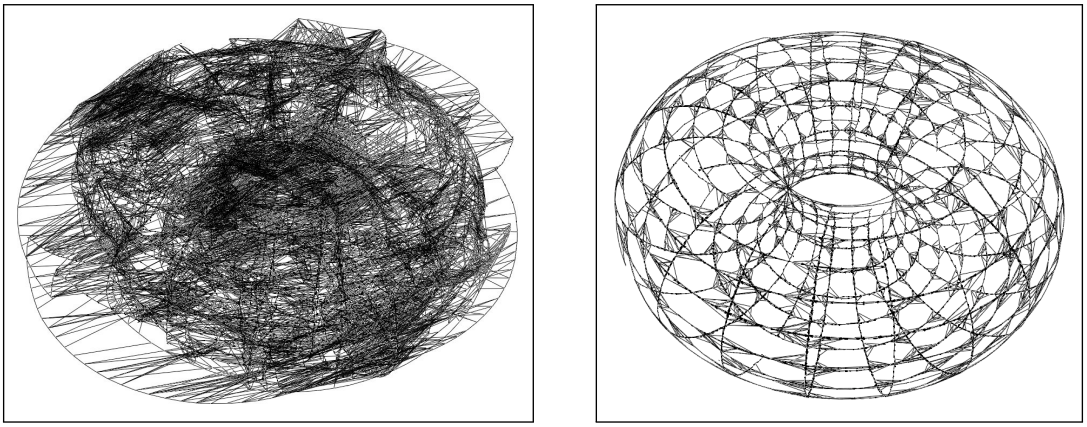
robot poses, 576 landmarks, 3,296 odometry readings, and 14,309 landmark observations. Figure 2.13a shows the initial state given by the odometry of the robot. After running our approach we obtain the consistent estimate depicted in Figure 2.13b. It requires around 0.2 s to optimize.

Finally, we consider a real-world 3D SLAM data set. In particular, due to the Grand Challenges organized by DARPA the robotics community gained interest in using a car as a robot. For this data set, such a car [152] equipped with a 3D range finder, an inertial navigation system which fuses GPS and IMU measurements, and various other sensors (which are not considered for the experiment) was steered through a multi-level parking garage. The whole trajectory is approximately 7 km long and contains several nested loops on all four levels of the garage and the surrounding streets. An adapted version of the front-end operating on 3D range data generated 1,661 robot poses, which are related to each other by 6,275 constraints. Each robot pose is associated with a local three-dimensional map, a so-called Multi-Level Surface map [209]. The odometry of the car provided by the inertial navigation system is considered for building the constraints connecting subsequent nodes. Additional constraints, the loop closures, are obtained by scan-matching the local maps whenever the robot revisits a region. As we can see in Figure 2.14a, the odometry of the car leads to an inconsistent estimate, which is mostly visible



(a) simulated odometry (b) estimate of our approach

Figure 2.15: (a) The simulated 2D Manhattan data set as given by the odometry and (b) after optimization with our approach.



(a) simulated odometry (b) estimate of our approach

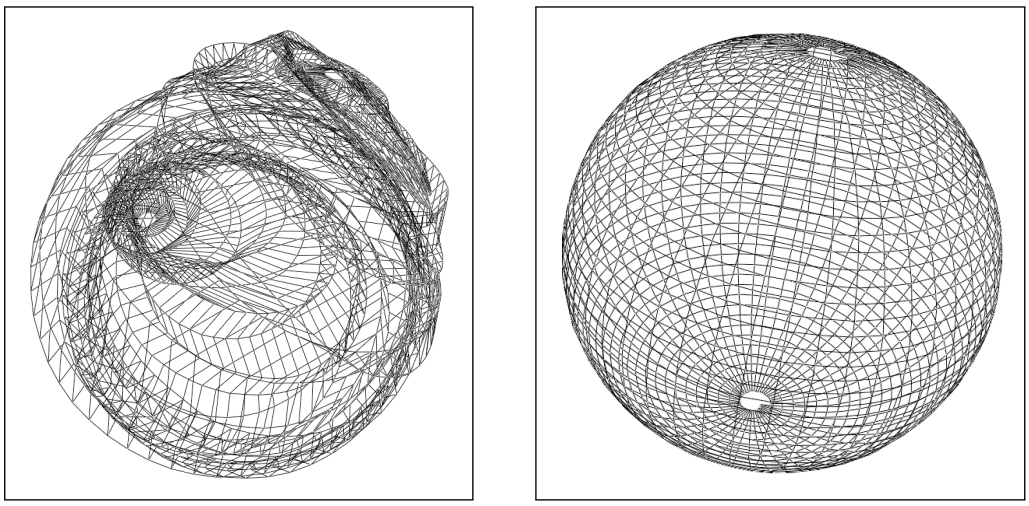
Figure 2.16: (a) The simulated Torus data set as given by the odometry estimate and (b) after optimization with our approach.

as incorrect height estimates. Running our framework, we are able to generate a decent estimate of the environment. In Figure 2.14b, we can clearly see the separate levels of the parking garage. The resulting map, obtained after around 0.2 s, was subsequently used to realize an autonomous parking maneuver, which is described in Chapter 8.

2.5.2 Simulation Experiments

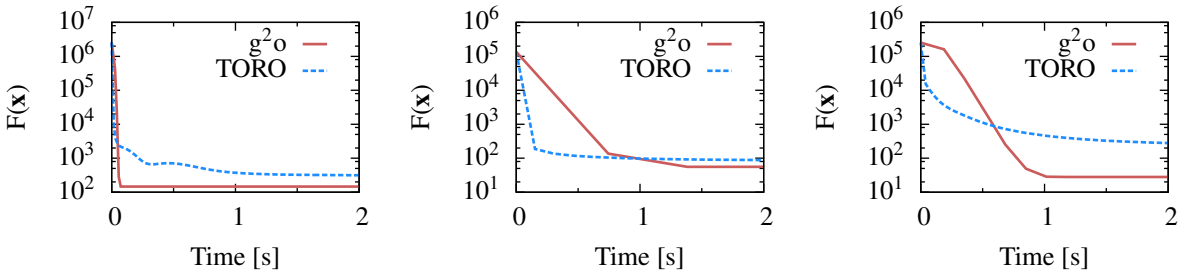
Here, we want to use large-scale simulated data sets to evaluate our approach quantitatively against TORO [78], which applies a variant of stochastic gradient descent [164] for optimizing Eq. 2.20. As TORO is only able to handle factor graphs with pose-pose constraints, we consider large-scale simulated data sets falling into this category: the Manhattan data with 3,500 poses and 5,598 constraints (see Figure 2.15), the Torus data set with 10,000 poses and 22,280 constraints (see Figure 2.16), and the Sphere data set with 2,500 poses and 4,949 constraints (see Figure 2.17).

We initialized both approaches with the initial state as given by the odometry. TORO, however, always computes a minimal spanning tree to determine the initial state by propagating the estimates along the tree. As we can see in Figure 2.18 the initialization performed by TORO reduces the error within the first iteration. Compared to g^{2}o, TORO requires more time to



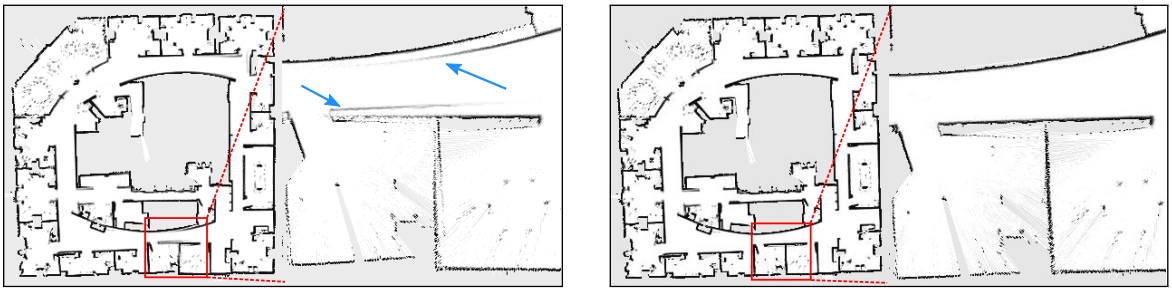
(a) simulated odometry (b) estimate of our approach

Figure 2.17: (a) The simulated Sphere data set as given by the odometry estimate and (b) after optimization with our approach.



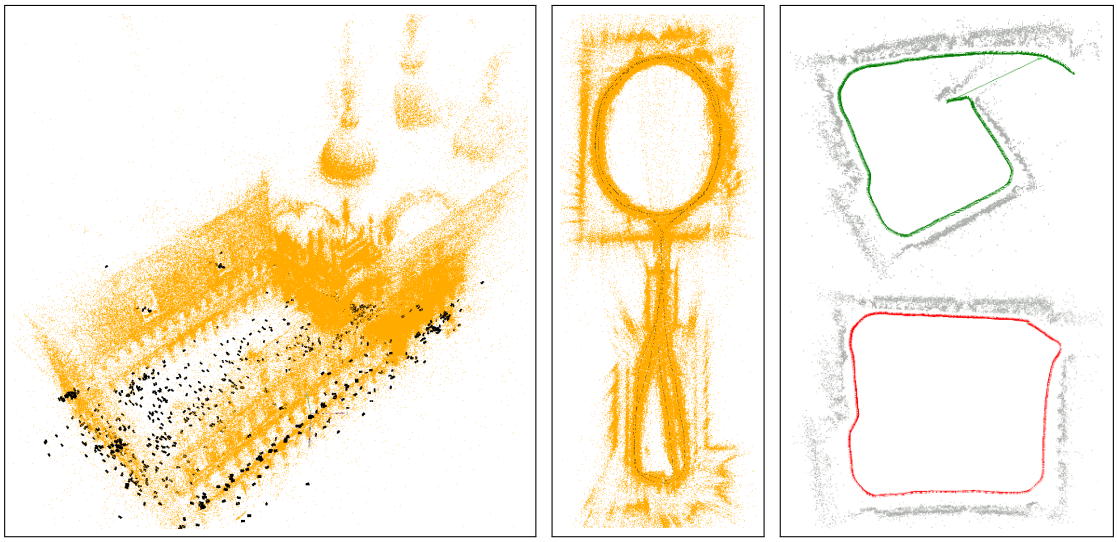
(a) Manhattan3500 (b) Torus (c) Sphere

Figure 2.18: Comparison of TORO and g^{2}o on (a) the Manhattan data set, (b) the Torus data set, and (c) the Sphere. As we can see g^{2}o quickly converges to the true solution. TORO reduces the error by the spanning tree initialization in the first iteration, whereas it exhibits a slow convergence afterwards compared to g^{2}o.



(a) map with a large error (b) map with a small error

Figure 2.19: Influence of the value of $F(\mathbf{x})$ on the quality of the resulting map. This image visualizes the influence of the remaining error on the map quality. The map shown in (a) has an error which is approximately ten times larger than the error of the map depicted in (b). While both maps represent the structure of the environment well, the higher error leads to artifacts in the map which are indicated by arrows in (a).



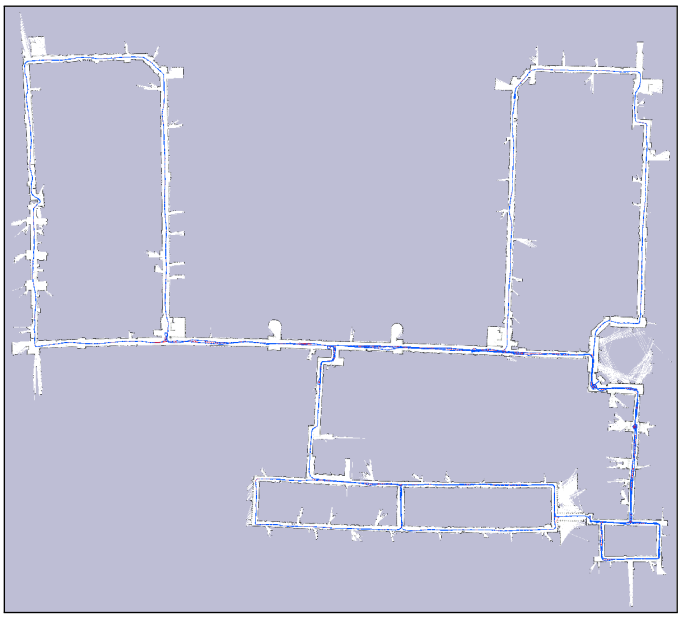
(a) Venice (b) New College (c) Keble College

Figure 2.20: The BA real-world data sets and the scale-drift data set used for evaluating g^{2}o: (a) Venice data set, a monocular BA data set. Orange dots indicate the position of world features, wheres the small black boxes correspond to the locations of the cameras. (b) New College data set [192] acquired with a stereo camera mounted on a robot. Again, orange dots indicate the position of features in the environment. (c) The pair of images shows the Keble college data set which was processed by monocular SLAM [201]. Here, scale drift occurs ((c) top), which can be corrected when closing the loop using 7 DoF similarity transformations ((c) bottom).

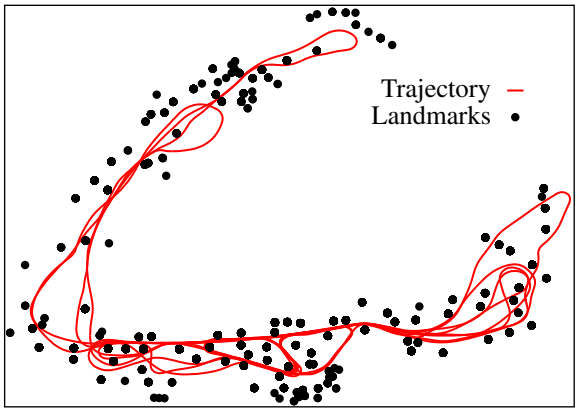
converge and the achieved value for $F(\mathbf{x})$ is larger. Figure 2.19 visualizes the influence of a higher error value on the map quality. As we can see, a higher error may have a crucial effect on the details of the map. While both maps capture the main structures of the environment, the higher error shows up as artifacts in the maps.

2.5.3 Runtime Comparison

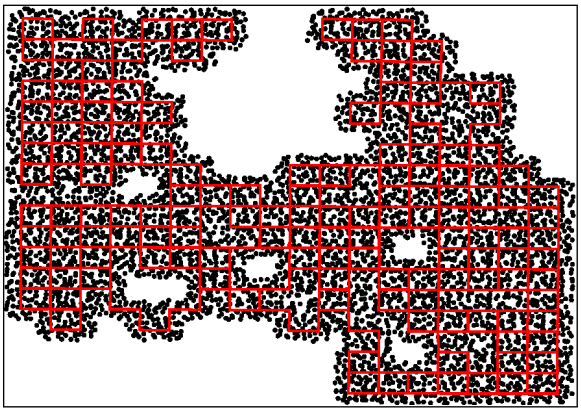
In the following, we report the time needed by each approach except TORO to carry out one iteration. Note that we here focus on the approaches which also implement nonlinear least squares by means of Gauss-Newton or Levenberg-Marquardt, as we have already examined the runtime of gradient descent in the comparison with TORO as reported above. As 3D pose-graph data-sets we considered the real-world parking garage data set as depicted in Figure 2.14, and the simulated sphere data set shown in Figure 2.17. Figure 2.20 depicts the real-world BA data sets: A monocular BA data set taken in Venice and a stereo one acquired with a real robot at the New College in Oxford, and the pose-graph of the Keble college, which is used to perform scale drift-aware SLAM using 7 DoF similarity constraints [201]. In addition to these 3D data sets we considered the following 2D data sets: The Intel Research Lab, a real-world data set taken in an office environment (see Figure 2.12); the simulated Manhattan3500 data set, a pose-graph simulated in a grid-world (see Figure 2.15); the Killian Court data set which is a large-scale indoor environment (see Figure 2.21a); the Victoria Park data set acquired with a car where trees are used as landmarks (see Figure 2.21b); and the grid5000 which simulates a robot observing landmarks in a grid-world (see Figure 2.21c). The number of variables and constraints is given in Table 2.1 for each of the data sets. We choose this collection of data sets for evaluation as it covers a large variety of different optimization problems to highlight



(a) MIT Killian Court



(b) Victoria Park



(c) Grid5000

Figure 2.21: 2D Datasets used for evaluating g^{2}o. (a) 2D pose-graph of the Killian Court at the MIT. (b) 2D data set with landmarks of the Victoria Park. (c) Grid5000, a simulated landmark data set.

Dataset	# poses	# landmarks	# constraints
Intel	943	-	1837
MIT	5489	-	7629
Manhattan3500	3500	-	5598
Victoria	6969	151	10608
Grid5000	5000	6412	82486
Sphere	2500	-	4949
Garage	1661	-	6275
Venice	871	530304	2838740
New College	3500	488141	2124449
Scale Drift	740	-	740

Table 2.1: Overview of the test data sets.

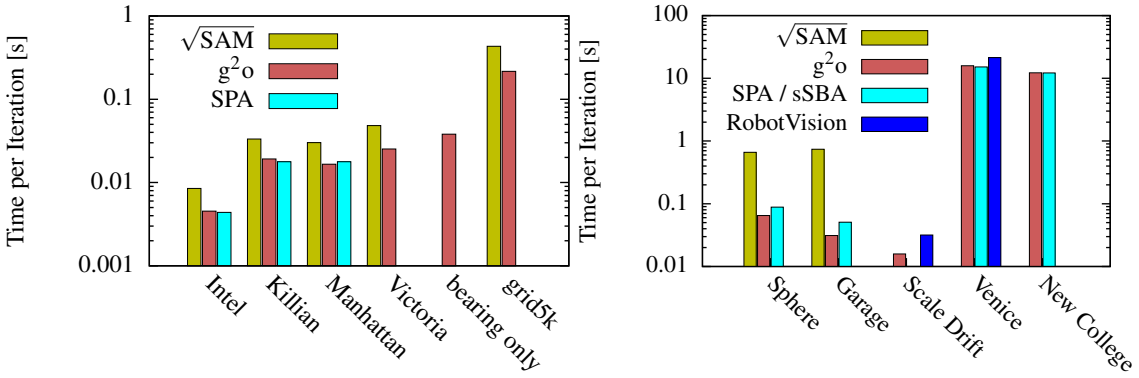
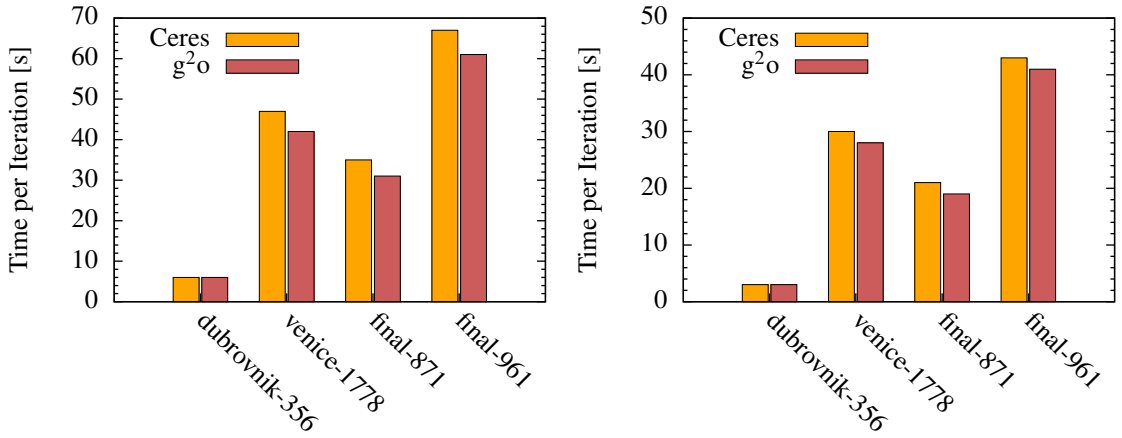


Figure 2.22: Time per iteration for each approach on each data set.



(a) single thread (b) four threads in parallel

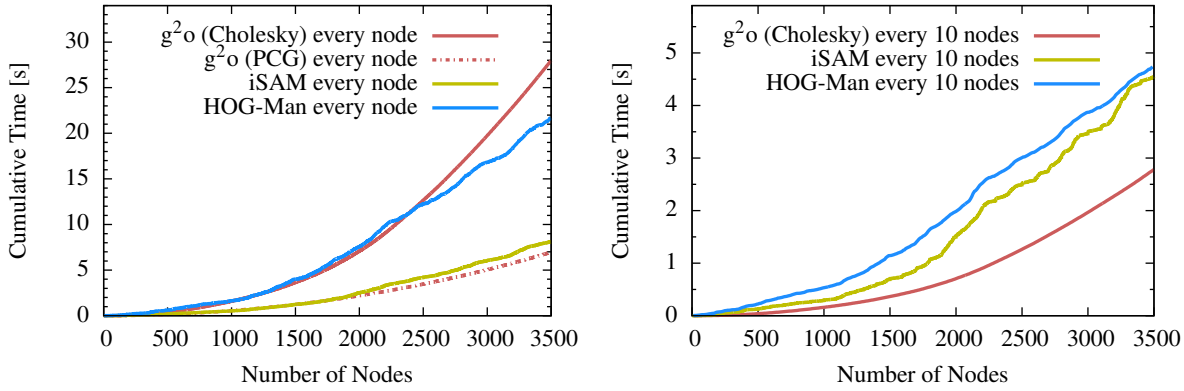
Figure 2.23: Comparison of g^{2}o and Ceres Solver. (a) The time required for one iteration on several data sets with a single threaded implementation. (b) Running four threads in parallel allows us to take advantage of the architecture of nowadays CPUs.

the general applicability of our approach. All experiments are executed on one core of an Intel Core i7-930 running at 2.8 Ghz.

We provided each approach with the full optimization problem and carried out 10 iterations, and measured the average time spent per iteration. In this set of experiments g^{2}o applies Cholesky decomposition to solve the linear system using CHOLMOD, which is also used by the approaches we compare to. Therefore, the time required to solve the linear system is similar for all approaches and the difference reflects the efficiency in constructing the linear system. The results are summarized in Figure 2.22.

Our system g^{2}o is faster than the implementation of $\sqrt{\text{SAM}}$ on all the 2D and 3D data sets we tested. While in principle they implement the same algorithm, g^{2}o takes advantage of an efficient front end to generate the linearized problem. On the 2D pose graph data sets the runtime of our framework is comparable to the highly optimized but specific SPA implementation. On the BA data sets g^{2}o achieves a similar performance to sSBA, which is slightly faster than our general framework. Compared to RobotVision, g^{2}o is on average two times faster.

Additionally, we compared our system g^{2}o with Ceres Solver, a general framework released by Google. Instead of modifying Ceres Solver to operate on our data sets, we adapted g^{2}o for dealing with the case handled by Ceres Solver, namely Bundle Adjustment with unknown camera intrinsics. Both systems are able to run several threads which allows us to fully load nowadays CPUs, which consist of several cores, with parallel computations. Figure 2.23a shows



(a) optimization after every node (b) optimization after ten nodes

Figure 2.24: Online processing of the Manhattan3500 data set. (a) Runtime for optimizing after inserting a single node. (b) Runtime for optimizing after inserting ten nodes.

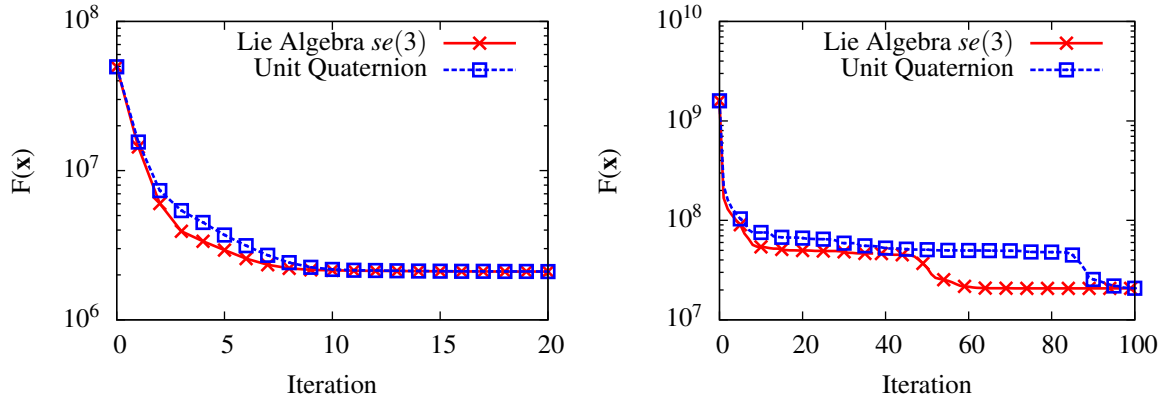
the time required for one iteration by g^{2}o and Ceres Solver using a single threaded implementation, while Figure 2.23b depicts the performance on the same data but running four threads in parallel. The evaluation shown here consists of a sub-set of the data sets created by Agarwal *et al.* [4]. g^{2}o requires the same amount of time or less than Ceres Solver on all considered examples regardless of the number of threads running in parallel. When utilizing more than one thread, both approaches build the Schur complement given in Eq. 2.72 by executing block-matrix multiplications in parallel. While both approaches construct the system in parallel, its solution is evaluated using the same number of threads in both settings. This explains why the decrease in runtime does not scale with the numbers of threads. Note that solving the system may also exploit the availability of multiple cores. We did not evaluate this scenario since both g^{2}o and Ceres Solver utilize CHOLMOD as a black box for solving the system and thus the effect would be the same for both systems.

Note that while g^{2}o focuses on batch optimization, it can be used to process the data incrementally by optimizing after adding nodes to the graph. On small data sets the efficiency of g^{2}o yields performance similar to approaches that are designed for incremental use, such as iSAM [101] or HOG-Man (our approach for online processing a pose-graph by constructing a hierarchy, see Chapter 3). As visualized in Figure 2.24, by optimizing every ten nodes or by relaxing the termination criterion of PCG for optimizing after inserting a single node g^{2}o can achieve acceptable runtimes. On larger data sets, however, solving Eq. 2.16 from scratch in each time step becomes inefficient. We will deal with this issue in the next chapter, where we present our hierarchical approach for online mapping, which employs g^{2}o as an efficient building block.

As mentioned, g^{2}o can compute the Jacobian J_k numerically, which allows us to rapidly prototype a new optimization problem or a new error function. By specifying the Jacobians analytically one can achieve a substantial speed-up. For instance, the duration of one iteration on the Garage data set drops from 80 ms to 40 ms when applying the analytic Jacobian. Despite the reduced efficiency we did not observe a decrease in the accuracy when applying the numeric Jacobian.

2.5.4 Testing different Parameterizations

Since our framework only requires to implement the error function and the update step, we are able to easily compare different parameterizations. To this end, we implemented two different



(a) New College (b) Venice

Figure 2.25: Evolution of $F(\mathbf{x})$ using unit quaternions versus the Lie algebra $se(3)$ on (a) the New College data set and (b) Venice data set.

Dataset	CHOLMOD	CSparse	PCG
Intel	0.0028	**0.0025**	0.0064 ± 0.0026
MIT	0.0086	**0.0077**	0.381 ± 0.364
Manhattan3500	0.018	0.018	**0.011 ± 0.0009**
Victoria	0.026	**0.023**	1.559 ± 0.683
Grid5000	**0.178**	0.484	1.996 ± 1.185
Sphere	0.055	0.398	**0.022 ± 0.019**
Garage	0.019	0.032	**0.017 ± 0.016**
New College	6.19	200.6	**0.778 ± 0.201**
Venice	1.86	39.1	**0.287 ± 0.135**
Scale Drift	0.0034	**0.0032**	0.005 ± 0.01

Table 2.2: Comparison of different linear solvers (time in seconds).

parameterizations for representing poses in BA. In the first parametrization, the increment $\Delta\mathbf{x}_i$ is represented by a translation vector and the axis of a unit quaternion, whereas in the second one the increments $\Delta\mathbf{x}_i$ are represented by members of the Lie algebra $se(3)$ [201]. We applied the different parameterizations to the New College and Venice data sets. The evolution of the error is depicted in Figure 2.25. Both parameterizations converge to the same solution, but convergence occurs faster using $se(3)$.

2.5.5 Comparison of Linear Solvers

Our system is able to apply different linear solvers for determining the solution of either Eq. 2.46 or Eq. 2.73. We currently have implemented two solvers based on Cholesky decomposition, namely CHOLMOD and CSparse [40]. Additionally, we implemented PCG as an iterative method using a block-Jacobi preconditioner. Table 2.2 summarizes the time required for solving the linear system on several data sets. PCG performs very well on the New College and Venice data sets, where it is around 7 times faster than CHOLMOD. The PCG convergence depends on how close the initial guess is to the optimum. We terminate PCG if the relative residual is below a given threshold (10^{-8} in the experiments). Therefore, PCG requires more time to converge, for example, on the MIT or Victoria data sets. CHOLMOD is faster by up to a factor of 30 than

Dataset	direct solution solve	Schur decomposition build / solve / total
Victoria	0.026	0.029 / 0.121 / 0.150
Grid5000	0.18	0.12 / 0.16 / 0.28
New College	15.18	3.37 / 7.07 / 10.44
Venice	33.87	11.25 / 1.78 / 13.03

Table 2.3: Comparison of different linear solvers. We measured the average time per iteration of g^{2}o (in seconds).

CSparse on the larger data sets. But surprisingly CSparse is the fastest solver on the smaller instances like the MIT data set where it outperforms both CHOLMOD and PCG.

2.5.6 Utilizing the Knowledge about the Structure

As discussed in Section 2.3.4, certain problems have a characteristic structure. Using this structure may result in substantial improvements in the solution of the linear system. Landmark-based SLAM and BA have the same linear structure: the landmarks/points can be only connected with the robot poses/cameras, resulting in a block diagonal structure for the landmark part of the Hessian H_{ll}.

In this experiment we evaluate the advantages of using this specific decomposition for landmark based-SLAM and BA. Table 2.3 shows the timing for the different data sets where we enabled and disabled the decomposition. From the table it is evident that preforming the decomposition results in a substantial speedup when the landmarks outnumber the poses, which is typically the case in BA. If the number of poses, however, becomes dominant, performing the Schur marginalization leads to a highly connected system that is only slightly reduced in size, and requires more effort to be solved.

2.6 Related Work

During the last decades, SLAM has been — and still is — an active research topic. One way to group the approaches developed throughout these years is the underlying estimation technique, see also the overview given by Durrant-Whyte and Bailey [11, 50]. There are approaches implementing a Bayes filter by means of an Extended Kalman filter (EKF) [132, 194] or its dual representation, the (Sparse) Extended Information Filter [57, 207]. Other approaches consider a particle filter [77, 154]. Instead of filtering, least squares error minimization [43, 78, 143] has become a standard approach for addressing SLAM.

Early solutions to SLAM were mostly based on applications of Extended Kalman Filters [132, 194] (EKF) for implementing a recursive Bayes filter. While such an EKF effectively estimates a fully correlated posterior about the robot poses and the landmarks in the environment, the weakness of the approach is caused by the assumption that have to be introduced for both the motion model and the sensor noise. Furthermore, linearization errors may lead to divergence and a violation of the assumptions may increase this risk [98, 211].

An alternative for the EKF is its dual representation, the (Sparse) Extended Information Filter [57, 207] (SEIF). Instead of maintaining the mean μ and its covariance Σ a SEIF approach updates the information matrix $\Omega = \Sigma^{-1}$ and the information vector $\Omega\mu$. One advantage of SEIF is that it is able to incorporate additional information in constant time. In particular,

Thrun *et al.* [207] take advantage of the approximative sparsity of the information matrix for obtaining the uncertainty estimate by neglecting small entries in the information matrix caused by marginalization. In contrast, this approximation may lead to a overconfident estimate of the pose the robot. To this end, Eustice *et al.* [57] presented a technique that more accurately computes the error-bounds within the SEIF framework and therefore reduces the risk of becoming overly confident. More recent work [56, 148] demonstrated that the SEIF techniques can be made exactly sparse. Since SEIF is the dual representation of the EKF, it suffers from the same drawbacks, namely the approximation by a linear function and thus the introduction of linearization errors into the estimation process which may cause divergence.

Another technique for implementing a Bayes filter is the so-called Rao-Blackwellized Particle Filter [48, 77, 154]. In this approach, each particle carries its own estimate of the trajectory of the robot and hence the map of the environment. The different trajectories arise from the noise in the motion of the robot. As each particle stores its own map, a particle filter does not suffer from the above mentioned linearization errors. A sophisticated data structure for the map [51] reduces the cost for re-sampling, while in general the computational complexity scales with the number of particles. For example, an improved proposal distribution [77, 83] allows us to reduce the required sample size. Likewise, KLD-sampling [63] adapts the number of particles dynamically depending on the difference to the true distribution. Additionally, special care has to be taken to avoid particle impoverishment, for example, re-sample only when needed instead of after each time step [77].

In contrast to the above mentioned techniques based on filtering, a smoothing approach based on graph optimization is able to modify the whole trajectory at all the time as they keep all measurements in memory. This allows us to re-linearize when needed. Such graph optimization problems have been studied intensively in the area of robotics and computer vision. One seminal work is that of Lu and Milios [143] where the relative motion between two scans was measured by scan-matching and the resulting graph was optimized by iterative linearization. At that time, optimization of the graph was regarded as too time-consuming for online performance. Due to recent advancements in the development of direct linear solvers (e.g., the work by Davis and his colleagues [37, 40]), graph-based SLAM has re-gained popularity and a huge variety of different approaches to solve SLAM by graph optimization have been proposed. For example, Howard *et al.* [92] apply relaxation to build a map. Duckett *et al.* [49] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of constraints. They assume that the orientation of the robot is know by employing a compass. This renders the problem to be linear.

Olson *et al.* [164] suggested a stochastic gradient descent approach to optimize pose graphs. Later, Grisetti *et al.* [78] extended this approach by applying a tree-based parameterization that increases the convergence speed. Both approaches are robust to the initial guess and rather easy to implement. They, however, assume that the covariance is roughly spherical and thus have difficulties in optimizing pose-graphs where some constraints have covariances with null spaces or substantial differences in the eigenvalues.

Graph optimization can be viewed as a nonlinear least squares problem, which typically is solved by forming a linear system around the current state, solving, and iterating. One promising technique for solving the linear system is preconditioned conjugate gradient (PCG), which was used by Konolige [111] as well as Montemerlo and Thrun [153] as an efficient solver for large sparse pose constraint systems. Because of its high efficiency on certain problems, g^{2}o includes an implementation of a sparse PCG solver which applies a block-Jacobi pre-conditioner [94].

More recently, Dellaert and colleagues suggested a system called $\sqrt{\text{SAM}}$ [43] which they

implement using sparse direct linear solvers [40]. In a joint work with Konolige *et al.* [110], we showed how to construct the linear matrix efficiently by exploiting the typical sparse structure of the linear system for applying an online variant of the Levenberg algorithm. The latter approach is restricted to 2D pose graphs. Moreover, the experiments carried out by Konolige *et al.* [110] show the advantage of a least squares approach in terms of accuracy and runtime against TORO, and SEIF approaches. In g^{2}o we share similar ideas with these systems. Our system can be applied to both SLAM and BA optimization problems in all their variants, e.g., 2D SLAM with landmarks, BA using a monocular camera, or BA using stereo vision. g^{2}o showed a substantially improved performance compared these systems on all the data we used for evaluation purposes.

Our approach assumes the node and constraints as given and focuses on determining the most likely map of the environment. We concentrated on implementing an efficient optimization back-end that can operate with different front-ends. For example, Bosse's ATLAS framework [22] or the work presented by Nüchter *et al.* [162] may generate the input data for our approach. Gutmann and Konolige [82] also demonstrated an approach which is suitable for generating the constraints. Finally, the work by Olson [163] describes a front-end for data association. Here, a correlative scan-matcher estimates the motion of the robot between subsequent time steps. Furthermore, loop closures are obtained by matching observations based on features, while wrong ones are filtered by spectral clustering. As outliers in the data association have a crucial effect on the estimate, several approaches for handling them have been proposed. For example, introducing multi-modal constraints, which are then approximated by the maximum likely component [166], deals with outliers efficiently. Here, one mode is the null-hypothesis. Likewise, the switching priors method [203] deactivates constraints which presumably correspond to outliers in the data.

In computer vision, Sparse Bundle Adjustment [140, 210] is a nonlinear least squares optimization method that takes advantage of the sparsity of the Jacobian pattern between points and camera poses which arises while performing Structure from Motion. Recently, there have been several systems [94, 107] that advance concepts of sparse linear solvers and efficient calculation of the Schur reduction (see Section 2.3.4) for large systems ($\sim$100M sparse matrix elements). There are also new systems based on nonlinear conjugate gradient that never form the linear system explicitly [4, 27]; these converge more slowly but can work with extremely large data sets ($\approx$1000M matrix elements). We compared, g^{2}o to the SSBA system of Konolige [107], which is an efficient publicly available system. Our approach g^{2}o achieves a comparable performance.

Another publicly available framework for addressing nonlinear least squares is the Ceres Solver [3] released by Google, where it is used for realizing Google Street View among addressing various optimization problems. For solving the linear system Ceres Solver also employs CHOLMOD [37] as g^{2}o does. Our comparison shows that g^{2}o outperforms Ceres Solver in terms of runtime on BA problems, whereas both approaches compute the same solution.

2.7 Conclusions

In this chapter we explained the probabilistic formulation of the full SLAM problem which can be solved by phrasing it as a least squares problem. This led us to g^{2}o, our extensible and efficient open-source framework for batch optimization of functions that can be embedded in a graph. Relevant problems falling into this class are graph-based SLAM and SfM, two fundamental and highly related problems in robotics and computer vision. Modifying g^{2}o to a new problem instance only requires to define the error function a procedure for applying a perturbation to the current solution. Furthermore, we can easily embed a new optimization

algorithm into the framework, which enables us to verify the characteristics of the specific solver on a wide range of problems sharing this graph structure. In particular, we showed in our experiments the applicability of g^{2}o to various variants of SLAM (2D, 3D, pose-only, and with landmarks) and to Bundle Adjustment. Our evaluation, which we performed on an extensive collection of data sets, demonstrates the overall good performance of our approach. g^{2}o achieves a performance comparable to implementations of problem-specific algorithms and often even outperforms them. An open-source implementation of the entire system is freely available as part of ROS and on OpenSLAM.org.

The research community quickly adopted g^{2}o after its release as open-source. Hence, we are only able to give an incomplete overview about the approaches utilizing g^{2}o. For example, g^{2}o is considered as optimization back-end for implementing Visual SLAM [199], SLAM with a Kinect depth sensor [53], or monocular cameras mounted on a car [141]. Additionally, researchers considered g^{2}o as the underlying optimization framework for implementing extensions [128, 166, 203] to handle outliers in the data association. Julier *et al.* [97] utilize g^{2}o for multi-rate estimation by decomposing the system into "slow" and "fast" states.

In the following chapter, we will address the issues that we have identified in this chapter. Namely that the iterative approach may converge to a local minimum. As we have mentioned, this effect is a result of the shape of the sensor model and the quality of the initial guess. Based on the technique presented in this chapter, we will improve the convergence properties by introducing a hierarchy of factor graphs. Additionally, the hierarchy is also beneficial for online mapping, as only a smaller problem that approximates the original one has to be optimized in each time step.

Chapter 3

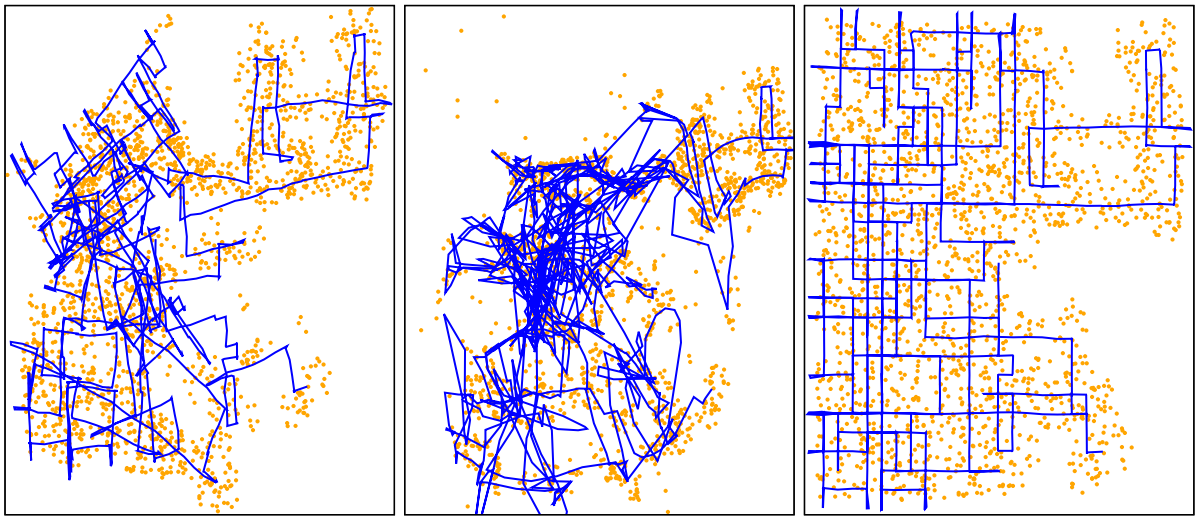
Hierarchical Optimization for Graph-Based SLAM

In the previous chapter, we discussed how to obtain a solution to popular problems in robotics and computer vision like simultaneous localization and mapping (SLAM) or Structure from Motion (SfM) for a given data association, by constructing a least-squares optimization problem. In this chapter, we present an approximation to the solution which addresses the issues that we have identified in the previous chapter. First, the approximation converges to the correct solution on problem instances where the standard methods fail. Second, we derive an algorithm which shows an increased efficiency allowing us to apply it online while the robot is exploring. To this end, we employ a divide-and-conquer approach that exploits the structure of the factor graph. Our approach has been validated on real-world data and in simulated experiments and is able to succeed in finding the global minimum in situations where other state-of-the-art methods fail.

• • • • • • • • • • •

As we have seen in the previous chapter, several problems in autonomous robotics and computer vision, like simultaneous localization and mapping (SLAM) or Structure from Motion (SfM), can be addressed by solving a least squares problem for a given data association. Furthermore, we outlined a common way to solve these least squares problems by applying iterative methods — like Gauss-Newton or Levenberg-Marquardt — which refine an initial guess until convergence or matching another termination criterion. Unfortunately, if this guess is out of the convergence basin of the algorithms, the iterative optimization may, however, fail to reach the global minimum. We will address this issue in the following. In particular, we suggest an approximation of the original problem which increases the chance to converge to the global minimum.

Additionally, we observed that SLAM and SfM exhibit a strong locality, which results in the sparse structure of the corresponding factor graph. The variables in these problems are correlated when they are temporally or spatially close. The temporal locality is the consequence of the sequential data acquisition, while the spatial locality arises from the limited range of the sensors. In this chapter, we will discuss an algorithm which exploits the locality among other typical properties of the SLAM problem.



(a) simulated odometry (b) estimate of LM (c) estimate of our approach

Figure 3.1: A robot equipped with a stereo camera is simulated in a Manhattan world. The trajectory is drawn in blue, whereas the features are depicted in orange. (a) The initial guess is computed by composing the odometry of the robot. (b) Running a standard iterative Levenberg-Marquardt algorithm yields a sub-optimal estimate. (c) Our approach converges to the correct solution.

The key idea of our approach is to partition the input factor graph into small locally connected sub-graphs. Each of these sub-graph represents a small part of the whole factor graph and could be viewed as a local map, which can be solved independently in a robust and efficient manner. The solution of these local maps, however, cannot be combined together in a straightforward way. To this end, from each partial solution we construct a simple factor graph that constrains the relative positions of the variables in the solution. Each sub-graph represents an approximation of the original one. The union of them, however, exhibits a larger convergence basin than the original problem. The factors in these graphs incorporate the information contained in the sub-graph from which they were generated. Thus, we refer to them as *condensed* measurements. To determine the global layout of the partial solutions in the space, we find the minimum of the union of all translated sub-graphs. As this translated problem has a larger convergence basin, we have increased the chance of finding the correct minimum. We utilize a hierarchy to represent the different layers of abstraction. Figure 3.1 shows a motivating example of our approach, where we simulated a robot with a stereo camera driving in a Manhattan maze. A standard iterative Levenberg-Marquardt (LM) algorithm converges to a local minimum. Our approach, however, estimates the correct solution.

In addition to improving the convergence characteristics of the standard methods, we also gain another advantage. So far, we have presented means to solve the nonlinear least squares problem that arises in SLAM, but we ignored that the moving robot might need a solution while still exploring the environment. In such an online scenario, two problems need to be addressed: First, constraints need to be extracted from sensor data. This is referred to as the SLAM front-end. Second, given the constraints the most likely configuration as well as the pose uncertainty needs to be computed. During online operation, these two problems are addressed at the same time and the performance of the front-end typically benefits from an accurate estimate determined by the back-end. To this end, our hierarchy provides efficient means to estimate an accurate approximation of the solution which facilitates the data association process. In this scenario, the hierarchy reduces the complexity of the optimization problem but still contains all the relevant information needed by the front-end to operate successfully. Thus, the solution

can be provided faster than for the original problem. Note that while we focus on SLAM as a motivating application in the remainder of this chapter, our approach has the potential to address other problems which require to solve a least squares problem.

The remainder of this chapter is organized as follows. After discussing the typical properties of the SLAM problem in the next section, we present our approach which exploits these properties to improve the convergence in Section 3.2. Afterwards, in Section 3.3 we construct a hierarchy of pose-graphs for online mapping. In Section 3.4 we evaluate both the hierarchy of pose-graphs for online mapping and the hierarchical approach for batch or offline processing on a collection of real-world data sets and on simulated data. Subsequently, we then present the related work in Section 3.5.

3.1 Considerations about SLAM-Like Problems

As we have seen previously, the variables in the factor graph for SLAM and SfM represent spatial entities that are either poses of the moving sensor (such as laser range finders, cameras) or positions of the observed entities (landmarks, scans, local maps and so on). The factors correspond to the measurements that depend on single robot positions (like GPS, magnetic field, or attitude), temporally subsequent robot positions (like odometry, velocity, or acceleration), or they arise by the observation of a map element from a certain position of the observer. Usually, each measurement involves a set of variables that are spatially close. This results in a *local connectivity* of the graph since variables are related to spatial entities and only variables that are within a certain range between each other are connected.

In the presence of devices having a highly nonlinear model, like a bearing-only sensor or a monocular camera, and in absence of a good initial guess, iterative methods can fail. Observing the same scene with more informative sensors that are, for instance, capable of measuring also the range to a landmark or the depth of a point, increases the chances of success for the iterative methods. Thus, the approaches presented in Chapter 2 applied to SLAM-like problems are *sensitive to the sensor model*. The sensor model has a great impact on the profile of the error function, and thus on the size of the convergence basin. Clearly, the sensor model is a characteristic of the sensor and cannot be arbitrarily changed.

For short trajectories the open loop estimate obtained by using incremental approaches (wheel odometry, visual odometry, integration of the accelerometers) is sufficient to obtain a good solution despite using less informative measures, like bearing-only data. Thus, direct approaches work well on *small-sized* problems.

The key insight gained from the analysis of the SLAM problem is the following. Consistent solutions for small portions of the problem allow us to derive a new set of virtual measurements from the partial solution. Particularly, these virtual measurements are able to observe all dimensions of the variables. Hence, we can formulate another problem with a similar solution that uses a different sensor model, which is also less nonlinear. It is in general convenient to apply sensor models that have a smooth error profile and that observe the highest possible number of dimensions of the involved state variables.
We recall the three characteristics of graph-based SLAM that form the base of our approach:

1. **Local Connectivity**: The SLAM factor graph has a strong locality. Variables are related to spatial entities and only variables that are within a certain range between each other are connected.

2. **Sensitivity to the Sensor Model**: The used sensor model has a great impact on the profile of the error function and thus on the chances of finding the global minimum. It is

convenient to use sensor models that have the smoothest possible error profile and that observe the highest possible number of dimensions of the involved state variables.

3. **Sensitivity to the Size of the Problem**: The error in the estimate typically grows with the distance traveled by the robot. Optimizing small portions of the graph usually allows us to determine a reasonable solution for that portions.

3.2 Robust Optimization using Condensed Measurements

In this section, we present our approach which exploits the three characteristic features of SLAM outlined above. The advantage of our method is twofold. On the one hand, we are able to reduce the computational demands to compute an approximative solution. On the other hand, we address the shortcomings of the standard methods. To this end, let us recall the problems that we identified in Chapter 2. The result obtained by the standard Gauss-Newton and Levenberg-Marquardt algorithms heavily depends on the initial guess of the state and the smoothness of the measurement functions. For instance, a poor initial guess might lead to a suboptimal result because the approach may converge to a local minimum. We refer to Gauss-Newton and Levenberg-Marquardt as direct methods in the following. The novel approach proposed in this chapter allows us to obtain a solution for problems where the standard algorithms fail.

In Figure 3.2 we provide a graphical explanation for our approach on a landmark-based SLAM instance. Here, we restricted the example to binary factors only. Furthermore, the poses of the robot are indicated by triangles, whereas the landmarks are visualized as circles. As indicated in the figure, our approach assumes the availability of odometry measurements constraining subsequent poses of the robot. Our approach works as follows.

First, the original problem is partitioned into small chunks based on the trajectory of the vehicle. The dashed line Figure 3.2a indicates the border along which the problem is divided into smaller chunks. Each of these chunks forms a small factor graph describing a portion of the problem. Because of the local connectivity, each factor will capture a small contiguous portion of the environment that can be seen as a local map (see Figure 3.2b). These local maps are related to each other through variables belonging to more than a single local map. These *shared* variables are illustrated in red.

Second, we can obtain a reasonable solution for each of these problems by applying a direct method. As each individual problem only consists of a small sub-graph, the direct method is able to determine a correct solution. In absence of global measurements, for example, GPS or similar reference systems, these local maps can move arbitrarily in space without affecting the value of the objective function. This is known as gauge freedom, see, for example, the discussion by Triggs *et al.* [210]. Thus, to determine a unique solution, we need to artificially constrain some variables, i.e., we enforce that one pose of the robot cannot move. In the remainder of this chapter, we will refer to these variables as the *origin* (gauge) of the local maps. They are illustrated in dark blue in Fig. 3.2c. Having a solution for a local map means that we know a Gaussian approximation of each variable within the local map, relative to its origin. Furthermore, whatever sensor measurements have been considered to obtain the solution, the measurements yield a Gaussian approximation. For example, in a 2D landmark-based SLAM problem we can estimate the full 2D position of the landmark, whereas a single measurement itself may only be able to cover the relative bearing of the landmark with respect to the robot. This is due to the fusion of multiple measurements into a full estimate for the landmark. Section 3.2.1 provides a detailed description of this.

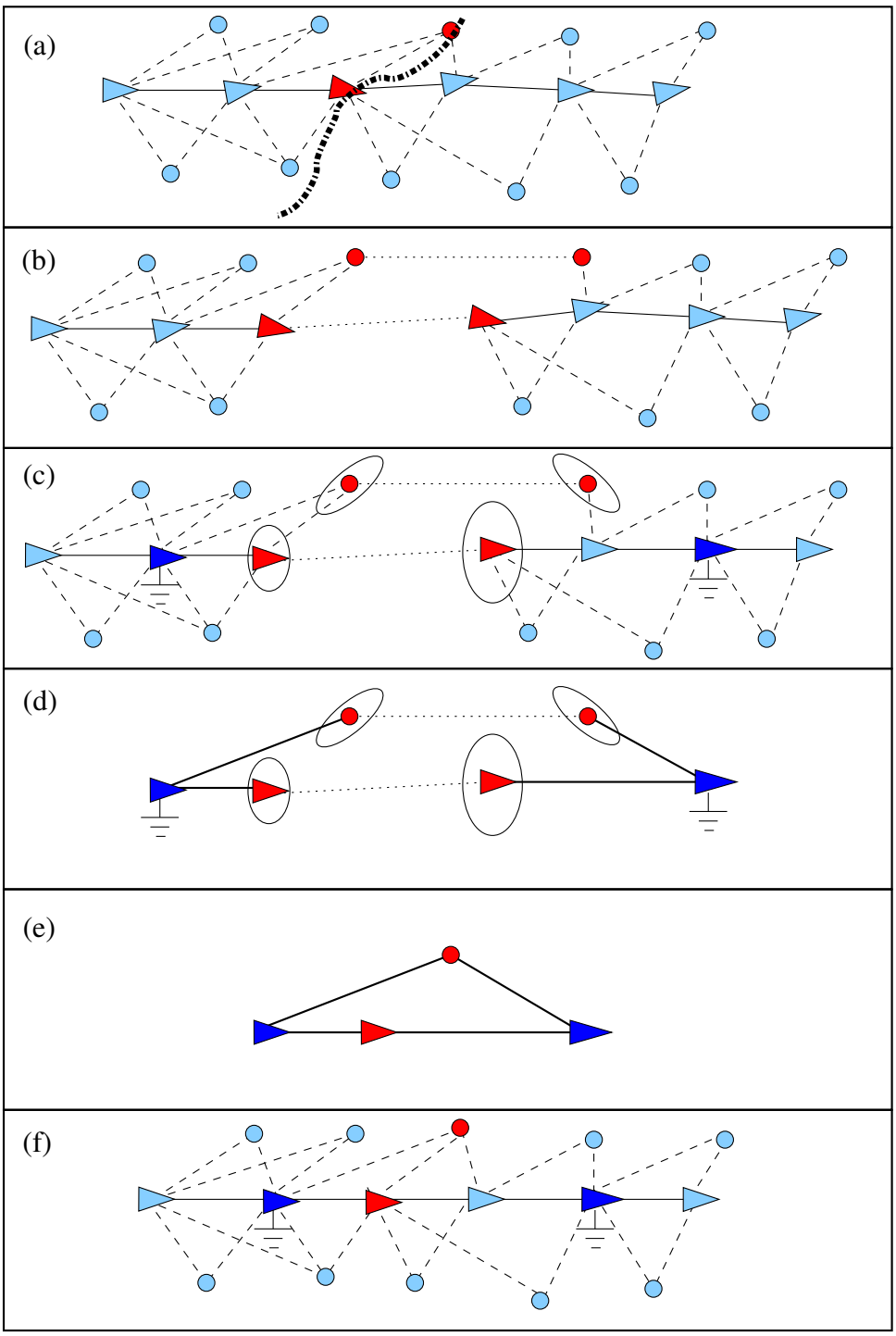


Figure 3.2: Overview of our optimization procedure on a small landmark-based SLAM problem. Here, we illustrate the factor graph by highlighting only the variables. The factors denote binary measurements and are encoded in the edges. (a),(b) We partition the problem into sub-graphs. The shared variables are in red, and dotted lines show the corresponding variables in different partitions. (c) We solve these problems independently with respect to their origins (dark blue), and we determine the marginal covariances of the shared variables. (d) We compute condensed factors connecting each shared variable to its origin. (e) We solve the complete problem on the condensed factors to align the local maps. (f) The alignment of the local maps provides a good initial guess for optimizing the original problem.

After obtaining a solution for the local maps, we seek for a global alignment that satisfies all the equality constraints induced by the shared variables. These constraints are depicted with dotted lines in Figures 3.2b, 3.2c, and 3.2d. Approaching this problem by initializing the original factor graph with the local solutions can fail since these local solutions will be destroyed during the global optimization. Hence, we reduce the problem from determining a global initial guess to finding a global alignment of the local maps and of the shared variables, while preserving the structures of the local maps computed before.

To this end, we replace the factor graph of each local map with another one. In particular, we modify the topology of the graph and consider different sensor models. This results in a factor graph which is easier to solve by applying a direct method. While constructing this factor graph, we utilize smooth sensor models that exploit our knowledge about the involved variables. Namely, we employ sensor models which are able to fully qualify each variable within the local map with respect to its origin. Referring to our example above, we will use a sensor model observing the x/y location of a landmark relative to the robot instead of only the relative bearing. We construct the reduced problem by considering the origins along with all the variables which are shared between the involved local maps. We then add a factor between each shared variable and the origin, as illustrated in Figure 3.2d. This factor depends on the entities which are related to each other. Clearly, the type of factor considered for constraining a pair of poses of the robot differs from the one representing an observation of a landmark. The mean and the information matrix of the measurement are computed by projecting the marginal covariance of a state variable in the measurement space via the unscented transform [96]. We provide a detailed discussion of this in Section 3.2.2.

This procedure converts the factor graph of a local map, which can have an arbitrary topology, into a star topology. The center of the star is the origin of the local map and the other nodes are the shared variables. Each shared variable is connected by only a single factor to the origin of the map. The variables which are not shared between several local maps are not considered in this step as they have no direct influence on the solution for aligning the sub-maps. The shared variables are connected to the origin by factors that are generated based on the solution of the local map. We call these factors *condensed* factors since they summarize the relationship between a variable and the origin of the local map by considering all measurements while optimizing the local map.

Based on the star topology of each local map, we assemble an approximation of the original global factor graph by combining all the newly computed factor graphs into a new sparser factor graph. The solution of this graph yields a global configuration of the origins of the individual local maps and the shared variables. This is illustrated in Figure 3.2e. Furthermore, since the sensor models are smoother than the original ones, the new problem will have a larger convergence basin. Thus, direct approaches are likely to find the optimal solution. The approximated solution for the origins of the local maps and the shared variables yields a good initial guess for the full problem. To this end, we arrange the local maps computed at the beginning of the procedure accordingly as shown in Figure 3.2f. At this point, an optimization, which considers the original factors, can further refine the approximated solution. Note that while we only described two layers of factors, the procedure outlined above may be applied recursively to form additional levels of abstraction.

In the remainder of this section, we will describe in detail how to partition the input factor graph into sub-graphs (local-maps), how to solve the local maps, and how to compute the marginal covariance of the shared variables. Finally, we describe how to compute the condensed measurements.

3.2.1 Constructing and Solving the Local Maps

The procedure outlined above requires to first partition the input into small sub-graphs leading to local maps. Since the local maps need to be merged in a later step, we should prevent the information stored in the original problem from being used multiple times. The relations between variables are modeled by the factors of the graph. Hence, it is sufficient to partition the factors of the input problem into different local maps. Each factor should be assigned to a different local map. Conversely, the variables can be replicated; a variable that appears in more than one local map becomes a shared variable. It acts as a vertex separator of the original graph.

Additionally, we want these local maps to admit unique solutions, which means that the resulting linear system (see Eq. 2.46) should have full rank, once we fix the vertex in the origin. In general, this is a challenging problem. Since we assume that we have odometry, we can construct a solvable sub-graph by selecting a contiguous segment of the robot trajectory. In our experiments, we considered a length of five meters. Knowing one pose of the robot in a segment and chaining the odometry measurements fully specifies the position of all other robot poses in the particular segment. Without odometry we would need to consider an additional criterion during clustering. For example, in SfM, when we assume a known camera calibration, we need at least five points that are visible in each of the two views for estimating the motion (up to scale) between the two cameras [136]. Subsequently, we consider all landmarks that have been observed by poses contained in this segment of the trajectory. Depending on the type of sensors, the landmarks can be fully observable or not. For instance, to triangulate the position of a 2D landmark observed with a bearing-only sensor, we need two observations from two different robot positions. If the senor also measures the range, a single observation is sufficient. Based on the odometry guess, we attempt to determine the position of all observed landmarks. The landmarks for which a position cannot be determined are not considered in this step. Hence, we obtain a set of factors that leads to a fully specified problem. Note that certain landmarks may be discarded from all local maps because their position cannot be determined in any of them. This results in ignoring some of the information when approximating the initial guess. We, however, recover this information in the final refinement stage of the algorithm, where we employ a direct method on the original factors starting from the initial guess computed by our approach.

Given the partitioning of the original graph into sub-problems, we solve each of them independently by using the direct methods described in Section 2.3. Furthermore, as outlined in Section 2.3.5, we can then compute the marginal covariances of the separators as the corresponding blocks of H^{-1}. Note that we obtain the marginal covariances of the increments $\Delta\mathbf{x}_i$ and not of the state variables $\mathbf{x}_i$. Hence, the covariances need to be propagated from the increments to the state variables.

3.2.2 Computing Condensed Factors

As final element of our approach, we need to describe how to compute a new set of factors that relate the separators $\{\mathbf{x}_i\}$ and the origin $\mathbf{x}_g$, given the marginal covariances $\{\Sigma_i\}$ previously computed. To this end, we note that our factor graph features a set of measurement functions

$$\mathbf{h}^{\text{typeOf}(\mathbf{x}_i)}(\mathbf{x}_g, \mathbf{x}_i) \overset{\text{def.}}{=} \mathbf{h}(\mathbf{x}_g, \mathbf{x}_i). \tag{3.1}$$

Each of those functions depends on the type of the separator $\mathbf{x}_i$. For example, in SLAM we typically have two types of variables: the poses of the robot, which either belong to $SE(2)$ or $SE(3)$,

and the landmarks, which are represented as vectors in either $\mathbb{R}^2$ or $\mathbb{R}^3$. The procedure described above guarantees that the origin of each local map is a pose of the robot. Consequently, this allows us to specify the virtual measurement function as

$$\mathbf{h}(\mathbf{x}_g, \mathbf{x}_i) = \begin{cases} \text{toVec}(X_g^{-1} X_i) & \text{if } (\text{typeOf}(\mathbf{x}_i) = \text{pose}) \\ X_g^{-1} \mathbf{x}_i & \text{if } (\text{typeOf}(\mathbf{x}_i) = \text{landmark}) \end{cases}. \tag{3.2}$$

Here, we assume that X_k is the transformation matrix corresponding to the pose $\mathbf{x}_k$. Intuitively, the expected measurement between two poses is the relative movement between one and the other. The expected measurement between a landmark and a pose is the projection of the landmark in the frame of the observing pose. The above choices are not unique. We selected these measurement functions to compute our condensed factors because the experiments demonstrate that they behave better than other models.

As we know which particular measurement function to select for each separator, we can determine the factors F_i connecting the origin and the separators:

$$F_i = \mathbf{e}_i(\mathbf{x}_g, \mathbf{x}_i)^\top \Omega_i \mathbf{e}_i(\mathbf{x}_g, \mathbf{x}_i). \tag{3.3}$$

To this end, we recall Eq. 2.52 that relates measurement function and error vector through the $\boxminus$ operator: $\mathbf{e}_i(\mathbf{x}_g, \mathbf{x}_i) = \mathbf{h}(\mathbf{x}_g, \mathbf{x}_i) \boxminus \mathbf{z}_i$. Thus, we need to determine the virtual measurement $\mathbf{z}_i$ and the information matrix Ω_i. The error function depends on the known measurement function $\mathbf{h}(\cdot)$ and on the yet unknown virtual measurement $\mathbf{z}_i$. Since the error should be $\mathbf{0}$ at the current solution of the local map, the virtual measurement vector at the equilibrium is

$$\mathbf{z}_i = \mathbf{h}(\mathbf{x}_g^*, \mathbf{x}_i^*), \tag{3.4}$$

where $\mathbf{x}_g^*$ and $\mathbf{x}_i^*$ are the actual values of the origin and of a separator after solving the subproblem.

So far, we have only determined the virtual measurement $\mathbf{z}_i$ of F_i. Thus, we still have to compute the information matrix Ω_i. Since the origin node is fixed, its covariance matrix is zero. This means that only the marginal covariance of $\mathbf{x}_i$ contributes to Ω_i. The procedure outlined in Section 2.3.2 yields the covariance matrices of the increments $\Delta\mathbf{x}_i$. Hence, we need to remap them through the error function. By rewriting the error function, we highlight the contribution of the increments

$$\mathbf{e}_i(\mathbf{x}_g, \mathbf{x}_i \boxplus \Delta\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_g, \mathbf{x}_i \boxplus \Delta\mathbf{x}_i) \boxminus \mathbf{z}_i. \tag{3.5}$$

We then remap the marginal covariance of $\Delta\mathbf{x}_i$ by using the unscented transform [96].

The (scaled) unscented transform estimates the result of applying a nonlinear function $f(\cdot)$ to a random variable $\mathbf{y}$, which is normally distributed and described by its mean $\mu_\mathbf{y}$ and covariance matrix $\Sigma_\mathbf{y}$. Let us introduce a second random variable $\acute{\mathbf{y}}$ which is defined as

$$\acute{\mathbf{y}} = f(\mathbf{y}). \tag{3.6}$$

The unscented transform is a method to recover the mean $\mu_{\acute{\mathbf{y}}}$ and the covariance $\Sigma_{\acute{\mathbf{y}}}$ of $\acute{\mathbf{y}}$, see also Figure 3.3. To this end, a set of weighted samples, the so-called sigma-points is considered. The set of sigma-points $\mathscr{Y}$ for the n-dimensional random variable $\mathbf{y}$ is given by

$$\mathscr{Y}^{[0]} = \mu_\mathbf{y} \tag{3.7}$$

$$\mathscr{Y}^{[i]} = \mu_\mathbf{y} + \sqrt{n+\zeta}\left[\sqrt{\Sigma_\mathbf{y}}\right]_i \tag{3.8}$$

$$\mathscr{Y}^{[i+n]} = \mu_\mathbf{y} - \sqrt{n+\zeta}\left[\sqrt{\Sigma_\mathbf{y}}\right]_i \quad \text{for } i = 1, \ldots, n, \tag{3.9}$$

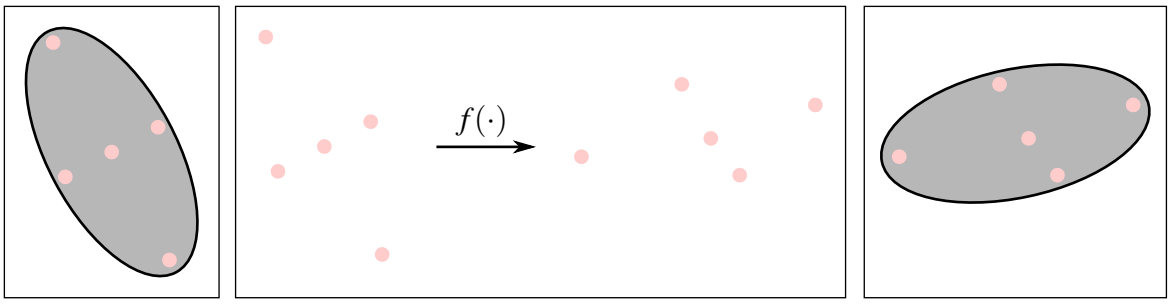


Figure 3.3: The unscented transform estimates the outcome of applying a nonlinear function $f(\cdot)$ to a Gaussian. (a) The original Gaussian distribution with the so-called sigma-points shown in pink. (b) Propagating the sigma-points through the nonlinear function. (c) Recovering of the Gaussian by considering the transformed sigma-points.

where ζ is a parameter, $[\cdot]_i$ corresponds to the i-th column of the matrix, and $\sqrt{\Sigma_\mathbf{y}}$ denotes the square root of the covariance matrix. A commonly used definition for it in context of the unscented transform is the Cholesky decomposition of $\Sigma_\mathbf{y}$, i.e.,

$$\sqrt{\Sigma_\mathbf{y}} \overset{\text{def.}}{=} L_\Sigma, \tag{3.10}$$

where $\Sigma_\mathbf{y} = L_\Sigma L_\Sigma^\top$. Furthermore, the weight of each sigma-point is set according to

$$w_\mu^{[0]} = \frac{\zeta}{n+\zeta} \tag{3.11}$$

$$w_\Sigma^{[0]} = w_\mu^{[0]} + \left(1-\alpha^2+\beta\right) \tag{3.12}$$

$$w_\mu^{[i]} = w_\Sigma^{[i]} = \frac{1}{2(n+\zeta)} \quad \text{for } i = 1,\ldots,2n, \tag{3.13}$$

where α and β are parameters. The parameters are chosen as follows:

$$\kappa \geq 0 \tag{3.14}$$

$$\alpha \in (0,1] \tag{3.15}$$

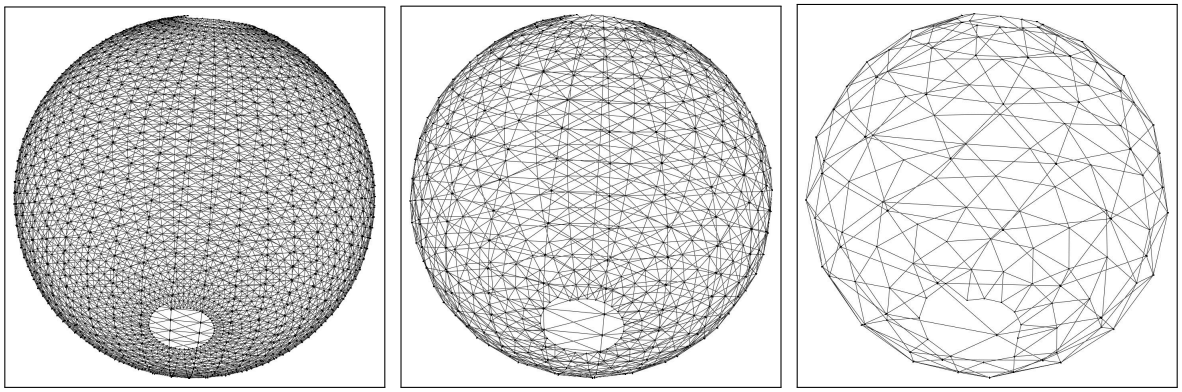
$$\zeta = \alpha^2 (n+\kappa) \tag{3.16}$$

$$\beta = 2. \tag{3.17}$$

The parameters κ and α influence the distance of the sigma-points to the mean $\mu_\mathbf{y}$ and $\beta = 2$ is the optimal choice for a Gaussian distribution [96]. Furthermore, $\kappa > 0$ guarantees that the reconstruction of the covariance given below results in a positive semi-definite matrix. In our implementation, we set $\kappa = n$ and $\alpha = 10^{-3}$. Given the sigma-points and the weights, we can compute $\mu_\acute{\mathbf{y}}$ and $\Sigma_\acute{\mathbf{y}}$ as

$$\mu_\acute{\mathbf{y}} = \sum_{i=0}^{2n} w_\mu^{[i]} \, f\left(\mathscr{Y}^{[i]}\right) \tag{3.18}$$

$$\Sigma_\acute{\mathbf{y}} = \sum_{i=0}^{2n} w_\Sigma^{[i]} \left(f\left(\mathscr{Y}^{[i]}\right) - \mu_\acute{\mathbf{y}}\right)\left(f\left(\mathscr{Y}^{[i]}\right) - \mu_\acute{\mathbf{y}}\right)^\top. \tag{3.19}$$



(a) lower level (b) intermediate level (c) top level

Figure 3.4: A hierarchy with three levels constructed by HOG-Man while optimizing a 3D network of 2,000 nodes and 8,647 constraints. (a) The lower level representing the original problem. (b) The intermediate level condenses the input data, whereas (c) the top level subsumes the information of the intermediate level. Each upper level gives a coarser representation of the underlying one.

To apply the unscented transform in our case, we extract a set of sigma-points $\{\boldsymbol{\sigma}^{[k]}_{\Delta\mathbf{x}_i}\}$ from the marginal covariance $\Sigma_{\Delta\mathbf{x}_i}$ of the increments $\Delta\mathbf{x}_i$ and we remap them through Eq. 3.5 as follows:

$$\boldsymbol{\sigma}^{[k]} = \mathbf{e}_i(\mathbf{x}^*_g, \mathbf{x}^*_i \boxplus \boldsymbol{\sigma}^{[k]}_{\Delta\mathbf{x}_i}). \tag{3.20}$$

We then compute Ω_i by inverting the covariance matrix reconstructed from the projected sigma-points.

The procedure outlined above determines the new factors used to describe a local map in a compact manner at a higher level of abstraction. The new factors are computed after considering the solution of a full portion of the problem and model the relationships between the origin of a local map and the separators. Furthermore, the set of new factors optimizes the position of the local maps with respect to each other.

A similar concept is applied for the hierarchy of pose-graphs to be described in the next section. As we have mentioned above, each cluster can be regarded as a local map and their alignment is refined in the upper level. This allows the robot to focus on the portion of the factor graph which encodes the vicinity around the current position of the robot. In the experiments, we will show that this results in an efficient method for addressing the SLAM problem online. Particularly, the advantages of our method become more evident on larger data sets.

3.3 Hierarchical Pose-Graph for Online Mapping

As we have seen in the previous section, we can subsume the information contained in multiple measurements, which are represented as several constraints, into a single constraint. In this section, we will specialize this concept to pose-graphs for online mapping. We call this approach HOG-Man in the following. HOG-Man implements the hierarchical strategies to reduce the computational demands during online mapping. It furthermore accurately models the coarse structure of the environment while the robot explores an environment. An accurate estimate of the structure of the environment is essential for making good data associations in the SLAM front-end. It is worth mentioning that our sparsification procedure is an accurate nonlinear approximation and accordingly, one can compute the covariances of the nodes by considering the sparse graph only. This enables the front-end to operate efficiently and to use

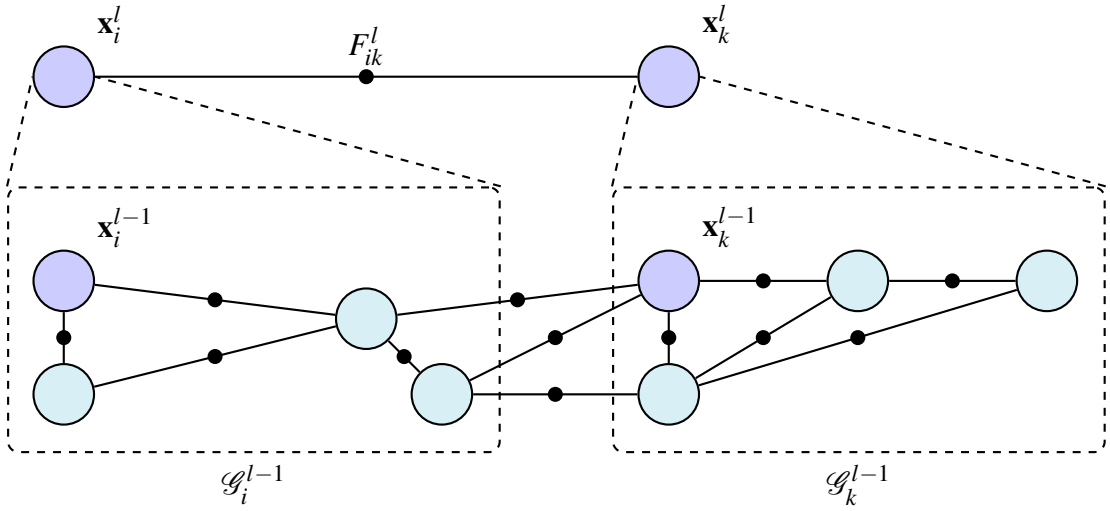


Figure 3.5: Example for a graph structure exhibiting two levels in the hierarchy. Every node $\mathbf{x}_i^l$ in the higher level corresponds to a connected sub-graph $\mathcal{G}_i^{l-1}$ at the lower level and to a node $\mathbf{x}_i^{l-1}$ within the sub-graph. A factor F_{ik}^l exists if two sub-graphs are connected in the lower level.

popular approaches for data association like the χ^2 test or the joint compatibility test [157].

The key idea of our hierarchical pose-graph is to represent the problem at different levels of abstraction. While the lowest level ($l = 0$) represents the original input, each upper level $l+1$ represent a coarser approximation of the underlying pose-graph at level $l \geq 0$. Figure 3.4 illustrates an example for a hierarchy with three levels. Each level in our hierarchy is a pose-graph, in which a node in the level $l+1$ represents a sub-graph at level l. To this end, a factor in any upper level $l > 0$ has to model the constraints between the sub-graphs in the underlying level. An important effect of our hierarchical representation is the reduction of parameters in each level for describing the environment. A reduced number of parameters leads to a faster optimization. The construction of the hierarchy itself introduces some additional operations, though. On the other hand, the reduction comes along with a coarser description of the environment.

More formally, we represent the problem using a hierarchy of L graphs. Let $\mathcal{G}^l$ be the pose-graph at level l. The graph $\mathcal{G}^l$ consists of a set of nodes $\{\mathbf{x}_i^l\}$ and a set of factors or edges $\{F_{ik}^l\}$ forming constraints on the nodes. Each node $\mathbf{x}_i^l$ at level l is associated to

(i) a "representative" node $\mathbf{x}_i^{l-1}$ at level $l-1$ and

(ii) a connected sub-graph $\mathcal{G}_i^{l-1}$ at level $l-1$.

In particular, we obtain a factor F_{ik}^l between the nodes $\mathbf{x}_i^l$ and $\mathbf{x}_k^l$ at level $l > 0$ if the two sub-graphs $\mathcal{G}_i^{l-1}$ and $\mathcal{G}_k^{l-1}$ are connected at the underlying level. Figure 3.5 illustrates a hierarchy having two levels. Note that the representative robot pose $\mathbf{x}_i^l$ in our hierarchy of pose-graphs is actually the origin of the local map constructed out of the sub-graph $\mathcal{G}_i^{l-1}$. The construction of the higher levels is controlled by partitioning the respective lower level into sub-graphs. Such a sub-graph can be interpreted as a local map, which is represented by a node in the higher level of the hierarchy. The factors, in turn, model the relations between those local maps and they arise from the connectivity between the neighboring local maps, which can also be interpreted as the shared variables.

For constructing the hierarchy, we first need to build the graph $\mathcal{G}^l$ at level l by taking into account the factors in $\mathcal{G}^{l-1}$ at level $l-1$. To this end, we form groups of the nodes in $\mathcal{G}^{l-1}$,

for example, by generating clusters based on the distance of the nodes to each other. Here, the distance between nodes is determined by considering the edges of the graph instead of the straightforward Euclidean distance. Our approach generates the clusters so that the diameter of each cluster does not exceed a specified threshold. Clearly, other choices are possible, but this straightforward threshold criterion worked well in our case. A possible extension might consider the segmentation of the environment into entities, such as rooms, buildings, or blocks of houses. Another viable solution is to apply general graph partitioning algorithms, for example, nested dissection as done in TSAM [158]. Let us denote the clusters built in each level by the set $\{\mathcal{G}_i^{l-1}\}$. The node $\mathbf{x}_i^l$ at level l is established by choosing the node $\mathbf{x}_i^{l-1}$ as a representative for the cluster $\mathcal{G}_i^{l-1}$.

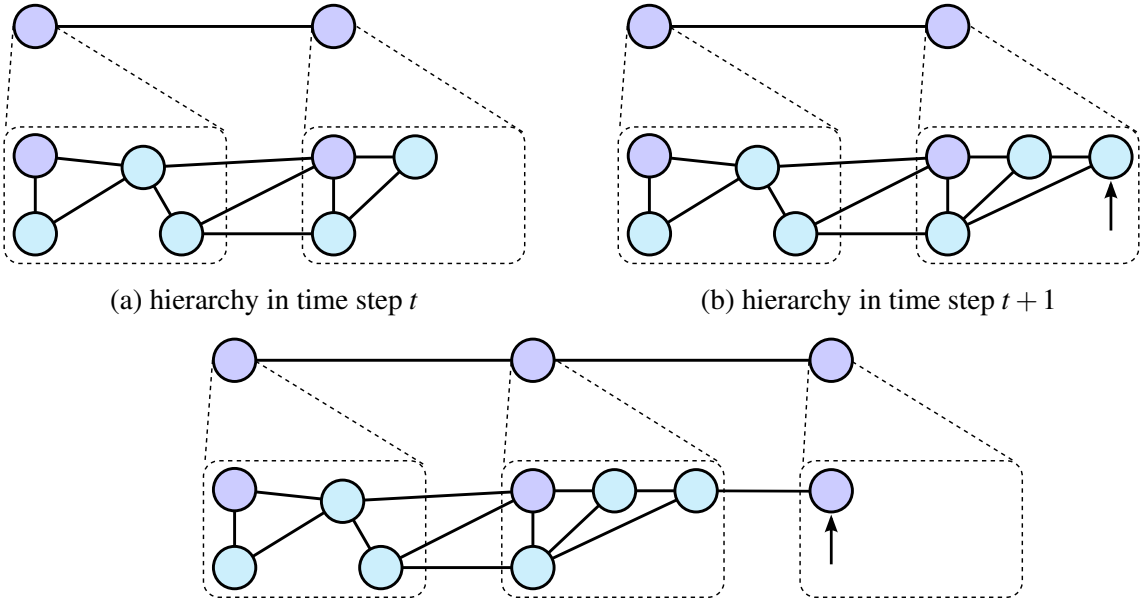
Given the clusters in the lower level $l-1$ and the nodes that are propagated to the upper level l, we need to add factors between the representatives of the clusters if the clusters are connected by factors, see also Figure 3.5. Such a factor F_{ik}^l has to integrate all the information provided by the factors in $\mathcal{G}_i^{l-1}$ and $\mathcal{G}_k^{l-1}$ as well as the factors relating both clusters. The factor $F_{ik}^l = \langle \mathbf{z}_{ik}^l, \Omega_{ik}^l \rangle$ is defined by a Gaussian. Hence, we need to determine the mean $\mathbf{z}_{ik}^l$ and information matrix Ω_{ik}^l which specify the factor F_{ik}^l. The parameters depend only on the configuration of the sub-graphs $\mathcal{G}_i^{l-1}$ and $\mathcal{G}_k^{l-1}$.

To this end, let $\mathcal{G}_{i\cup k}^{l-1}$ be the union of the graphs $\mathcal{G}_i^{l-1}$ and $\mathcal{G}_k^{l-1}$ as well as the factors establishing connections between the two sub-graphs. By optimizing the union graph $\mathcal{G}_{i\cup k}^{l-1}$, we obtain the maximum likelihood transformation for the relative motion between $\mathbf{x}_i^{l-1}$ and $\mathbf{x}_k^{l-1}$. This maximum likelihood transformation yields the mean $\mathbf{z}_{ik}^l$ of the factor F_{ik}^l in the upper level. The outcome of taking into account all the factors in $\mathcal{G}_{i\cup k}^{l-1}$ is the information matrix $H_{i\cup k}^{l-1}$. As we have previously seen in Section 3.2.2, we can extract the covariance of the factor F_{ik}^l by propagating the covariance for the increments through the error function. For this operation, we can again exploit the sparseness of $H_{i\cup k}^{l-1}$ which computes the desired block of the inverse efficiently [70, 99] without computing the full inverse.

As the robot moves through the environment, information has to be added to the hierarchical pose-graph. This is done by adding a newly created node and a set of edges connecting this node to the bottom level of the hierarchy just as in standard graph-based SLAM. According to a distance-based threshold criterion, the newly created node is either added to an existing group or it becomes the representative of a new one at level 0. This procedure is recursively executed upwards the hierarchy until no new groups need to be created. Edges are added and its parameters are updated following the procedure outlined above. Figure 3.6 provides an illustration of this procedure.

When the robot revisits a known region new edges on the bottom level $l = 0$ are introduced, but the revisiting does not lead to new nodes in the upper levels. We instead only need to update the factors in the upper level. Lets us assume that the new node $\mathbf{x}_t^{l-1}$ is added to the existing cluster $\mathcal{G}_i^{l-1}$. Optimizing the graph $\mathcal{G}_{i\cup k}^{l-1}$, which corresponds to the union of the two graphs $\mathcal{G}_i^{l-1}$ and $\mathcal{G}_k^{l-1}$, yields the updated mean $\mathbf{z}_{ik}^l$ and the updated information matrix Ω_{ik}^l of the factor F_{ik}^l. The update is performed for each cluster $\mathcal{G}_k^{l-1}$ that is related to the cluster $\mathcal{G}_i^{l-1}$. The updated factors are propagated upwards in the hierarchy.

After an update of the hierarchical pose-graph, an optimization is carried out. The optimization always starts at the top level using the nonlinear optimization approach presented in Chapter 2. As a result, all nodes at the highest level are updated. Changes, however, are only propagated to the lower levels if the optimization leads to significant changes in the configuration of the nodes. These changes are detected by monitoring the difference between each node $\mathbf{x}_i^l$ and its representative $\mathbf{x}_i^{l-1}$ in the lower level of the hierarchy. Whenever the distance



(a) hierarchy in time step t

(b) hierarchy in time step $t+1$

(c) hierarchy in time step $t+2$

Figure 3.6: Adding nodes to the hierarchy of pose-graphs. For convenience we indicate the binary factors by edges which connect the nodes of the binary factor. (a) The initial state of the hierarchy up to time step t. (b) In time step $t+1$, a new node (indicated by the arrow) is added to one of the sub-graphs. This requires to update the factor on the higher level which connect to updated sub-graph. (c) In the next time step, we again add a new node. Since it is not added to an existing sub-graph, we instantiate a new representative node in the upper level. Additionally, our approach adds a factor to the higher level which represents the constraints between the sub-graphs of the lower level.

between $\mathbf{x}_i^l$ and $\mathbf{x}_i^{l-1}$ exceeds a given threshold (in our current implementation: 0.05 m or 2 deg), we propagate the changes downwards. This is achieved by applying a rigid body transformation to each subgraph $\mathscr{G}_i^{l-1}$ so that $\mathbf{x}_i^l = \mathbf{x}_i^{l-1}$.

When required by the front-end, we generate a locally consistent estimate of a portion of the map by optimizing the corresponding sub-graph at the lowest level which exhibits the highest level of detail. During the optimization, we impose the additional constraints $\mathbf{x}_i^{l-1} = \mathbf{x}_i^l$. This allows us to propagate the estimate from the higher level to the lower level. Thus, the optimization performed by our approach can be regarded as lazy optimization, as only a subset of the variables is directly modified. After the mapping process one may consider to run a last optimization at level $l = 0$ to obtain the best possible map, though. This is related to our lazy optimization. The optimization of the hierarchy treats each cluster and hence each local map as rigid. The full optimization on the lowest level, however, allows them to move to determine the best configuration.

The description above illustrates means to generate and update a hierarchy of pose-graphs, each higher level in the hierarchy thereby provides a coarser representation. This hierarchy is in particular useful for implementing a SLAM back-end which efficiently provides a consistent approximation while the robot is mapping. It is worth to note that the number of nodes in the highest level scales with the mapped area and not with the distance traveled by the robot.

Furthermore, the bottom layer of the hierarchy in the description above was assumend to consist only of the poses of the robot. In the presence of landmarks, we could apply the technique discussed in Section 3.2 to generate a pose-graph that condenses all the information given by the individual landmark measurements and apply the hierarchical strategies discussed in this chapter on top.

	Prob. mass not covered	Prob. mass outside
Intel Research Lab	0.10%	10.18%
W-10000	2.53%	24.05%
Stanford Parking Garage	0.01%	7.88%
Sphere	2.75%	10.21%

Table 3.1: Comparison of the 3σ covariance ellipses between the original problem and the levels of our hierarchy.

3.4 Experiments

In our experimental evaluation, we focus on two different aspects. First, we analyze how our approach performs in online mapping, i.e., the graph is optimized after inserting one ore more nodes along with their measurements to the graph. Second, we evaluate the behavior of the hierarchy for batch optimization which is typically done after collecting all the data. Particularly, we investigate the convergence properties of the hierarchical optimization compared to standard approaches for nonlinear least squares.

3.4.1 Online Mapping

The experiments are designed to show that the hierarchical pose-graph is able to efficiently compute consistent estimates at all levels. For evaluating the accuracy we consider the mean of the estimate along with the uncertainty of the estimate. Furthermore the experiments demonstrate that our approach performs well with respect to other state-of-the-art techniques in terms of runtime, such that it can operate online.

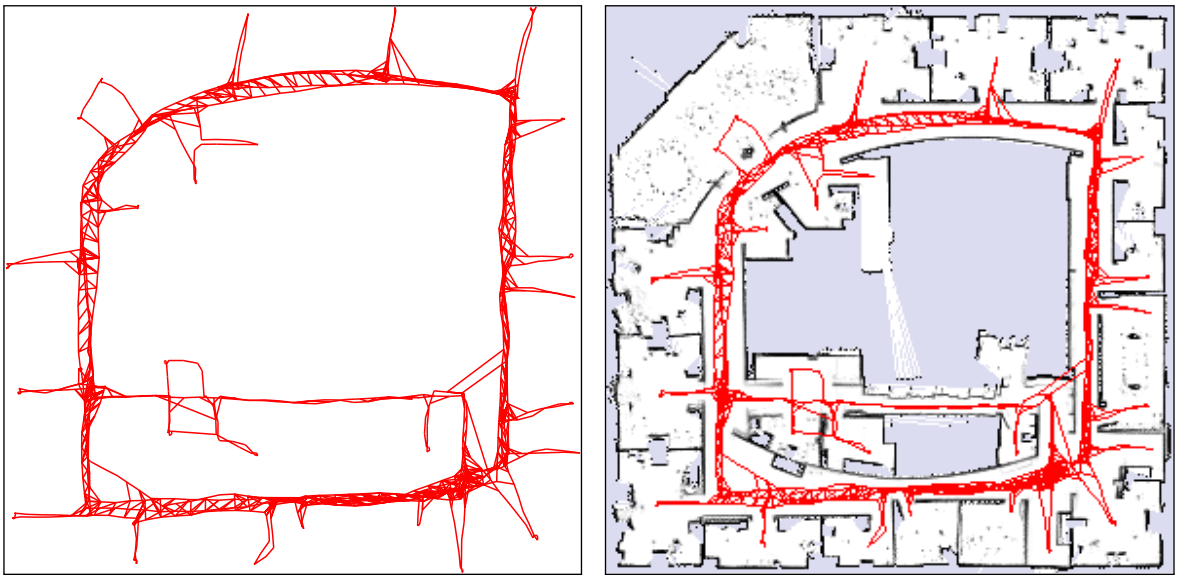
To support our claims, we compared our approach with TreeMap[1] [65], TORO[2] utilizing its incremental [76] and batch [78] version, and iSAM[3] [101]. We performed our tests on 2D data sets (Intel research lab data set and a simulated one) and 3D data sets (Stanford parking garage and a simulated sphere). Figure 3.7 illustrates the data sets. Both simulated data sets are the ones used in [78]. During all experiments, we use a hierarchy with three levels ($l = 0, 1, 2$).

Consistency of the Hierarchical Approach

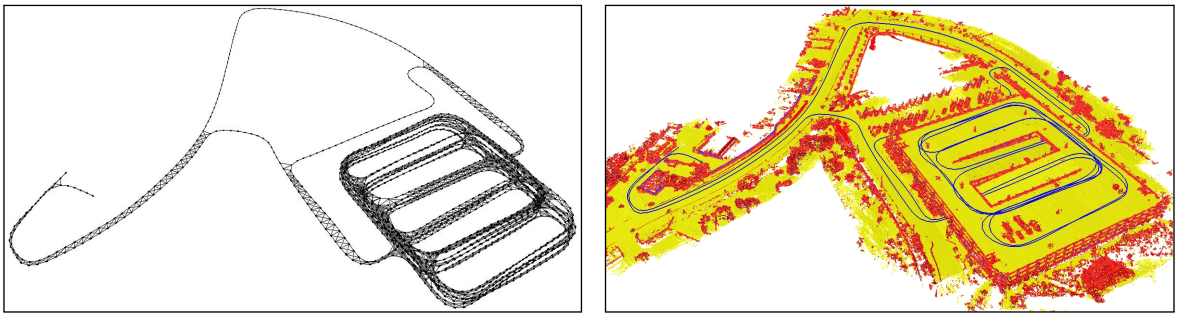
The first experiment is designed to show that the sparsified pose-graphs (levels greater than 0) yield a good approximation of the original problem. A consistent estimate is especially important for the SLAM front-end to establish good data associations. As we optimize the graph after inserting a single node to the graph, each approach has to solve an optimization problem that grows in each time step.

The baseline for evaluating the accuracy of our sparsified pose-graph is the original problem, fully optimized without the hierarchical approach. To evaluate the quality of the most sparsified pose-graph (top level), we compare the Gaussian associated to each node of these graphs with the corresponding distribution of the original problem. Table 3.1 illustrates the results evaluating the different levels of the hierarchy based on all data sets. As can be seen, the hierarchy approximates the original problem well. Especially, the probability mass that is not covered by the sparse pose-graphs (over-confident estimates) is around or below 0.1% for all real-world

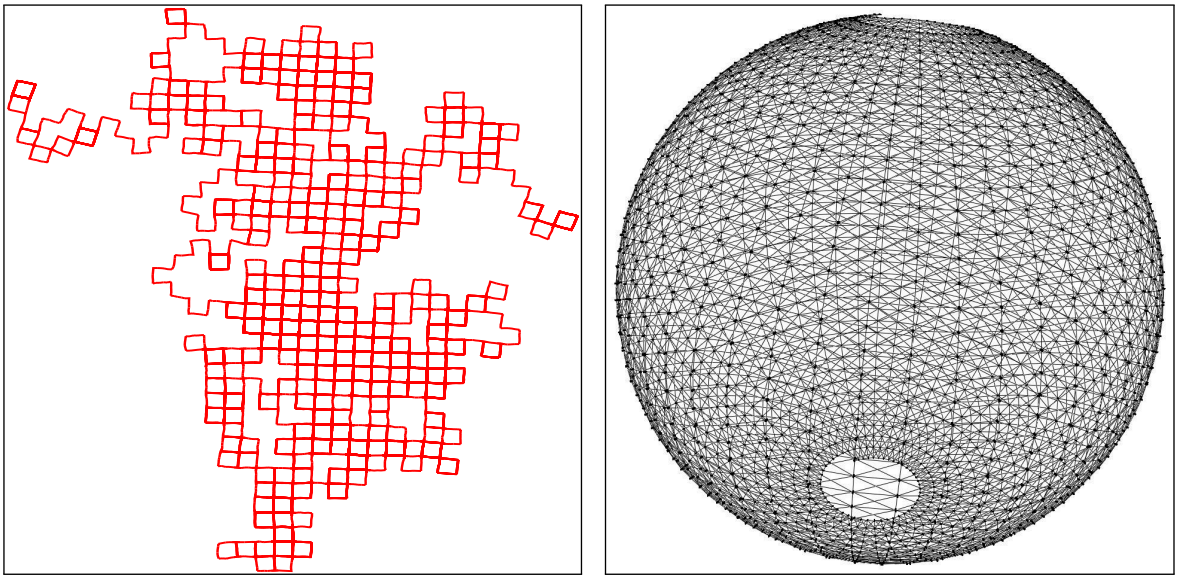
[1] https://www.openslam.org/treemap, Revision 5
[2] https://www.openslam.org/toro, Revision 19
[3] https://svn.csail.mit.edu/isam, Revision 8



(a) Intel Research Lab

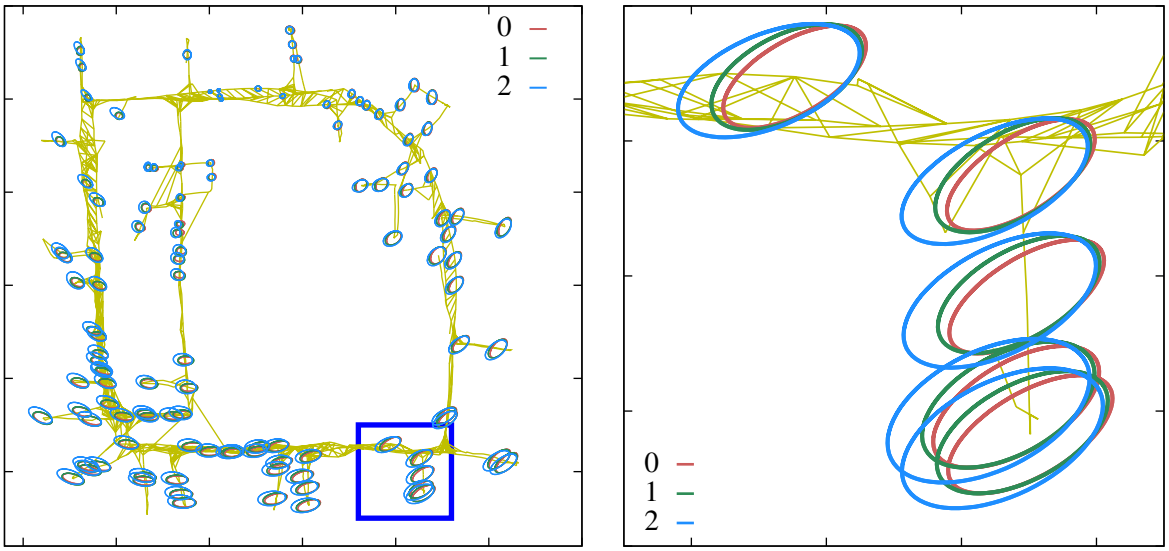


(b) Stanford Parking Garage



(c) W-10000 (d) Sphere

Figure 3.7: The four data sets used in our experimental evaluation: (a) the pose-graph and the grid map of the Intel research lab; (b) the pose-graph and the 3D map of the Stanford parking garage; (c) a simulated 2D data set (W-10000); (d) a simulated 3D data set (Sphere).



(a) full graph (b) zoomed view

Figure 3.8: The covariance ellipsoids obtained by the sparsified pose-graphs on the different levels of the hierarchy. Only the ellipsoids for the poses still present in the highest level are shown. (a) The full graph, in which the blue rectangle indicates the area of (b) the zoomed image.

avg./std./max. [ms]	Our Approach	TreeMap [65]	TORO [78]	iSAM [101]
Intel Research Lab	3 / 3 / 26	6 / 5 / 58	2 / 1 / 15	2 / 2 / 15
W-10000	25 / 20 / 183	1426 / 1342 / 9987	146 / 97 / 323	34 / 56 / 596
Stanford Garage	12 / 19 / 191	2D only	20 / 38 / 210	41 / 62 / 636
Sphere	38 / 16 / 82	2D only	78 / 67 / 234	54 / 70 / 831

Table 3.2: Runtime comparison for the different approaches.

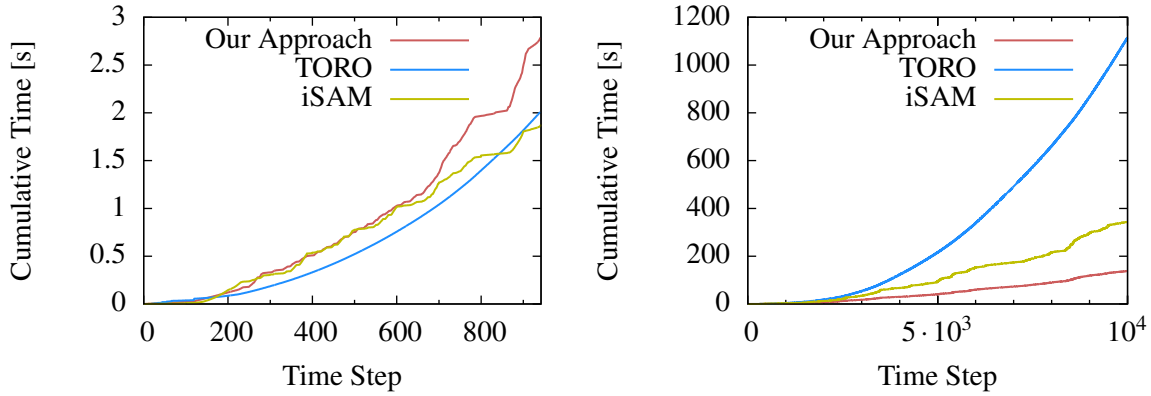
data sets. In general, the uncertainty ellipses of the sparse graphs are typically bigger than the ones in the original problem (around 10% for the real-world data sets).

For the evaluation, we compute the probability mass within the 3σ bounds of the original problem that lies outside the same bound of the sparsified graph and vice versa. We concentrated on the 3σ probability mass here since this forms the search bounds for data association in our SLAM front-end. Obviously, this evaluation can only be done for the nodes in the sparsified graph. In addition to the quantitative evaluation, Figure 3.8 visualizes the obtained uncertainty ellipsoids for the Intel Research Lab data set. As we can see, the ellipsoids as they are estimated by our approach in the highest level of the hierarchy represent the uncertainty of the original problem well. This means the robot is able to exploit this information for performing data association in the front-end.

Runtime Comparison

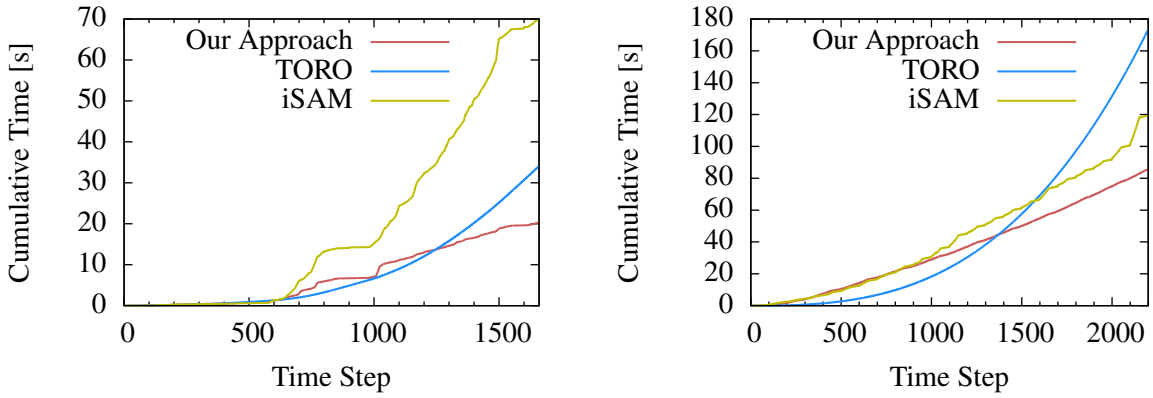
Furthermore, we analyzed the runtime required by the different approaches to optimize the pose-graph. We analyzed the average, the standard deviation, and the maximum runtime time of the optimization engine which was always executed after adding a single node. The timings are provided for all data sets analyzed in this chapter. The experiment has been executed on an Intel i7@2.8 GHz utilizing only one core.

Table 3.2 summarizes the results. As can be seen from the table, our approach clearly



(a) Intel Research Lab (b) W-10000

Figure 3.9: Online processing of the 2D data sets. The plots show the timings for (a) the Intel data set and (b) the W-10000 data set.



(a) Stanford Parking Garage (b) Sphere

Figure 3.10: Online processing of the 3D data sets. The plots show the timings for (a) the Stanford Parking Garage data set and (b) the Sphere data set.

outperforms TreeMap. A detailed investigation of the data structure of TreeMap showed that heavy leaves in the tree, i.e., leaves with many poses, led to the poor performance. This is caused by revisited places leading to a fully connected clique of poses. Even worse, TreeMap combines several successive poses into one leaf during the first visit and has to add a duplicate pose to each of these after each revisit.

While achieving a similar performance on the Intel data set, our method outperforms TORO on larger instances. In the comparably densely connected simulated pose-graphs this effect was more prominent compared to the real-world data sets. On the Intel Research Lab data set our approach performs similar to iSAM, whereas iSAM is slightly faster. In this rather small data set the overhead introduced by the additional operations for computing the hierarchy does not scale with the efficiency gained on the higher levels. Furthermore, we observe that the batch steps required in iSAM lead to a large maximum amount of time which is spent on optimizing after inserting a single node. This high cost, which occurs every 100 nodes utilizing the standard parameters, is amortized over the whole process.

Figure 3.9 shows the cumulative time of each approach except Treemap for online processing the 2D data sets, i.e., we optimized after inserting a single node along with the measurements of that node. We left out Treemap from the plots due to its poor performance caused by the effects described above. As we can see, on the small Intel data set all approaches achieve

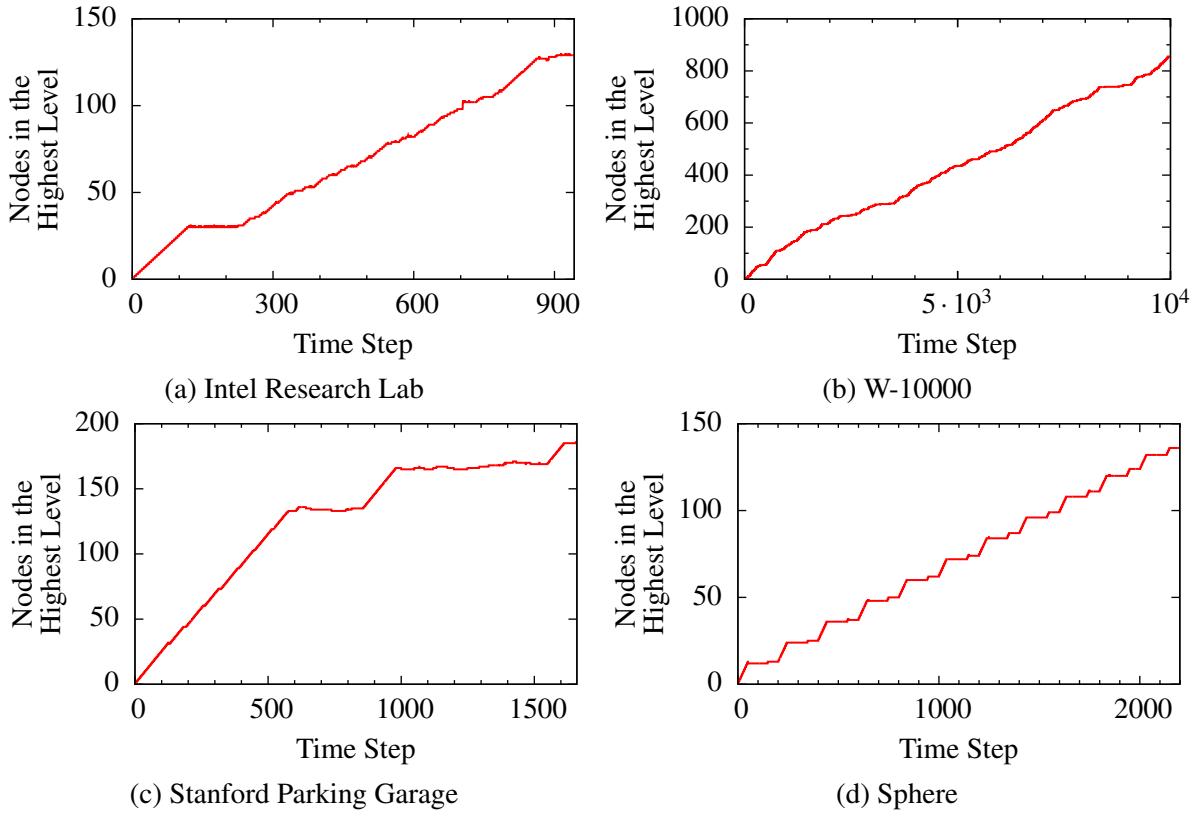


Figure 3.11: Number of nodes in the highest level of the hierarchy. (a) Intel; (b) W-10000; (c) Stanford Parking Garage; (d) Sphere.

a similar performance. On the large and densely connected W-10000 data set, our approach clearly outperforms the others. In Figure 3.10 we depict the cumulative timing for the 3D data sets. As we can see, our approach outperforms both TORO and iSAM on these data sets. In particular, the amount of loop closures in the Stanford garage data set causes a great amount of fill-in which iSAM has to handle until it carries out the next re-ordering step.

For a front-end utilizing our approach as a back-end the hierarchy has an additional advantage. For example, the computation of the uncertainty ellipsoids of the given state can be carried out on the sparsified representation of the highest level. As this level yields an accurate non-linear approximation, the robot can determine the covariances just based on this information. Additionally, the number of nodes present in the highest level scales with the area explored by the robot instead of the distance traveled by the robot. Figure 3.11 depicts the number of nodes in the highest level of the hierarchy while incrementally inserting nodes into the graph. In these plots, horizontal parts indicate that the robot is revisiting a known area, as no new nodes are constructed in the highest level. This effect is most notable in Figure 3.11c which shows the Stanford parking garage. In this data set the robot first explores the garage up to the highest level, then drives inside the garage, and finally moves to one of the exits. The exploration of the garage is finished after around 600 time steps, no new nodes are added to the highest level of the hierarchy while the robot remains in the garage. The robot reaches the exit after around 900 time steps. This leads to adding new nodes to the highest level as the robot explores the surrounding of the garage. Between time step 1,000 and 1,550 the robot revisits the parking structure, which results in no additional nodes on the highest level of our approach.

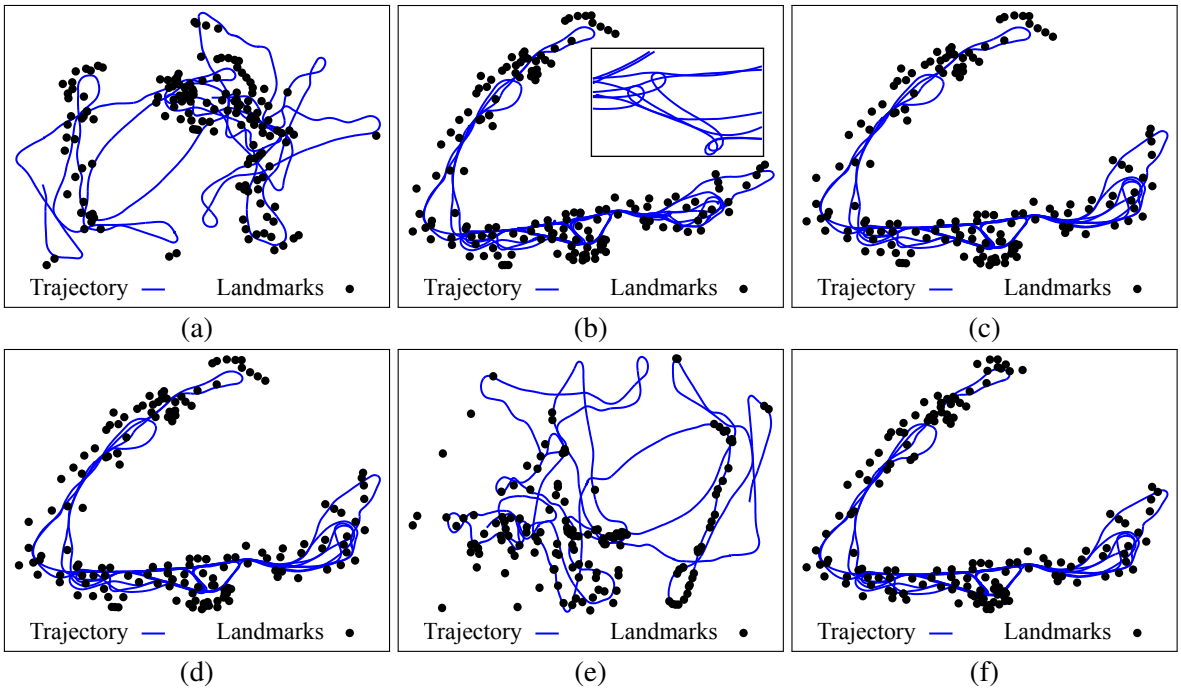


Figure 3.12: The Victoria-Park data set. The landmarks are shown as black dots and blue curves are the robot trajectories as they are estimated by different methods. (a) The initial guess. (b) By using a Cartesian sensor model the direct approaches fail in batch optimization, $F = 30607$. (c) The direct approaches succeed when run incrementally, $F = 389$. (d) Our approach succeeds. (e) The bearing-only data set cannot be optimized with direct approaches neither incrementally nor in the batch mode. (f) The correct map is obtained by our hierarchical method.

3.4.2 Batch Optimization

In the previous set of experiments, we focused on the efficiency gains of our strategies for optimizing the least squares problems by a divide-and-conquer scheme. In contrast to this, we now investigate the properties of our approach for offline or batch optimization. Here, the algorithm has to determine a solution once. In particular, we examine the convergence properties of our approach when starting from an initial guess. To this end, we validated our hierarchical approach on real-world data and performed extensive statistical tests on simulated data. On all data sets we compare our approach with the Levenberg-Marquardt (LM) implementation in our g^{2}o package. All results have been verified by both visual inspection and by comparing the errors of the final solution. Real-world experiments provide evidence on the real applicability of the results, whereas the ground truth of simulated data sets allow us to characterize the behavior of the approaches in a more detailed way.

Real-World Experiments

The first experiment is based on the popular Victoria-Park data set. It was acquired with a car equipped with a laser range finder and odometer. The trees in the park serve as point features and a feature extraction algorithm executed on the laser range reports the $x-y$ location of the trees. We will refer to this sensor model as the "Cartesian" data set. From this data set, we constructed a bearing-only data set, where we replaced the Cartesian observations of a landmark with the corresponding bearing measurement. As shown in Figure 3.12a, the noise in the odometry is relatively high.

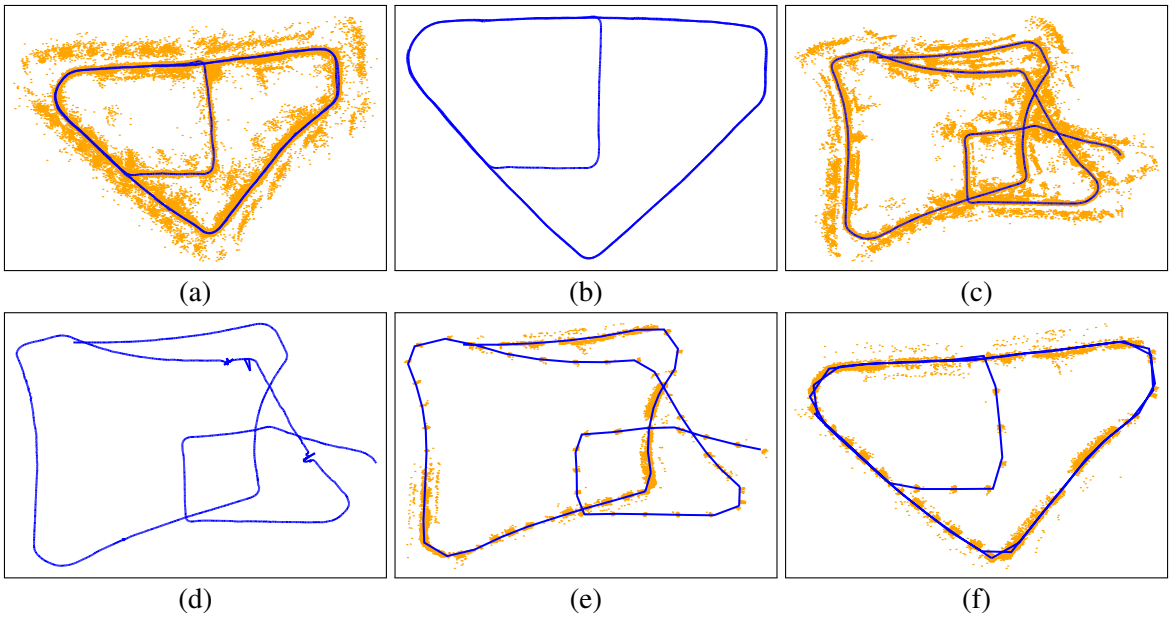


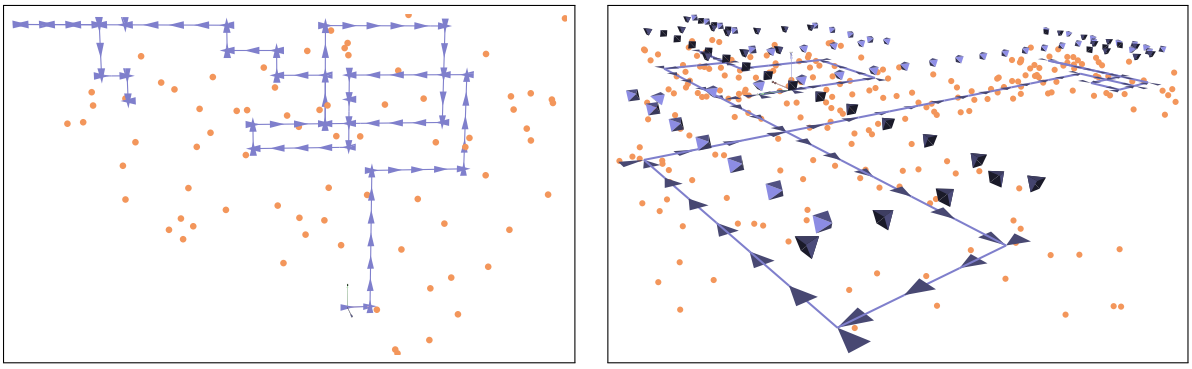
Figure 3.13: The top view of our real-world visual SLAM data set: the point features are displayed as little dots, the trajectory of the robot is depicted in blue. (a) The initial guess obtained by the SLAM algorithm. (b) The trajectory of the robot obtained by running LM on the good guess. (c) The noisy initial guess obtained by composing the odometry. (d) The result of the same algorithm using the noisy initial guess is even worse than the input. (e) The layout of the condensed problem solved by our approach after being initialized with the noisy initial guess. (f) The final result after the global alignment.

Processing the whole Cartesian data set with direct approaches does not give the correct solution (Fig. 3.12b), which can however be obtained by running the direct approaches incrementally, after inserting 50 sequential odometry measurements (Fig. 3.12c). Our approach, however, always finds the correct solution (Fig. 3.12d). The solution is the same as the one computed by the direct approaches run incrementally. The bearing-only data set cannot be solved by direct approaches when run either in batch mode or incrementally (Fig. 3.12e), due to the high nonlinearities in the sensor model. Conversely, our approach succeeds (Fig. 3.12f).

In the second experiment, we describe the results of processing a 3D data set acquired at the Freiburg University campus with a mobile robot equipped with a Bumblebee stereo camera. From each frame, we extracted visual features [16] along with disparity and constructed one large bundle adjustment problem enriched with odometry information. The results of the experiments are illustrated in Figure 3.13. For this data set, we considered two initial guesses: one obtained by optimizing the pose graph constructed by densely matching pair-wise observations (thus pretty accurate), and one based only on the wheel odometry. We processed this data set with direct methods both batch and incrementally. Direct approaches always succeeded in finding the optimal solution when initialized with the good guess, while they failed in all cases starting from the bad guess. Our approach succeeded in creating the local maps and determining a good initial alignment.

Simulated Experiments

We generated a set of synthetic 2D and 3D data sets by simulating a robot moving in a grid-world and sensing point landmarks in its neighborhood. In all cases, we created a synthetic world by placing a set of landmarks in the environment and letting the robot move for increasingly long trajectories through a simulated Manhattan world. The same synthetic world



(a) simulated 2D landmark data set (b) simulated 3D visual SLAM data set

Figure 3.14: Two simulated data sets, in which points represent point features, while blue lines indicate the trajectory of the robot: (a) The top view of a 2D data set and (b) the perspective view of a 3D data set.

was used to create different data sets, one for each sensor setup. Thus, in these experiments we tested the influence different trajectory lengths and different sensor modalities on the optimization outcome. Figure 3.14 shows the ground truth of two synthetic data sets used in our experiments.

In 2D we used point landmarks and simulated both a Cartesian sensor, similar to the one used in the Victoria Park experiment and a bearing-only sensor. In 3D we used an ideal Cartesian sensor capable of measuring the position of a landmark in the reference frame of the observer (this can model a 3D laser), a depth sensor, which measures the "depth" of points in the image plane (this can model RGBD cameras like the Microsoft Kinect), and finally we used a disparity sensor suitable to model stereo cameras.

Table 3.3 shows the characteristic of the different data set in terms of number of poses, landmarks and factors in the graph, and the type of sensor used to perceive the landmarks. Within Table 3.4 we report for each data set the value of the error $F(\mathbf{x})$ at the minimum found by the algorithms for the different data sets. F_{ideal} is the theoretical minimum value obtained by running LM using the ground truth as initial guess. F_{init} represents the error of the initial guess derived from the odometry. F_{lm} is the error of the solution obtained by running 100 iterations of the LM algorithm as it is implemented in the g^{2}o package, and F_{cond} using our approach to determine the initial guess and then running ten iterations of standard LM. We marked in bold the cases in which the respective approach failed to converge to the correct solution. The reader might observe that in some cases our approach did not reach the absolute minimum, but this is due to the fact that we limited the LM iterations to ten. We verified that running 100 iterations of LM results in reaching the theoretical minimum in our simulated experiments. Additionally, we observed a substantial speedup in using the condensed measurements. For a large problem of 5,001 poses computing the solution with LM takes approximately 18 minutes, whereas our approach required 9 minutes for computing the condensed graph and the subsequent ten iterations of LM. In total the generation of the condensed factors and the solution of the sparse problem took less than 4 minutes on a Core 2 Duo 2.6 Ghz using one single core. The column "# cond" reports the number of condensed factors in the global sparse problem, while the column F_{sparse} reports the initial and final error of the optimization of this sparse problem. The significant reduction of the error is possible because the constructed problem features a larger convergence basin than the input. For problems having a very small size, no condensed factors are generated. This is the case for the 2D data set with eleven pose variables, where the solution is computed with standard LM.

Type	# poses	# landm.	# factors	sensor	ID
2D	11	403	1229	bearing	1
				Cartesian	2
	101	1102	10223	bearing	3
				Cartesian	4
	1001	2000	105399	bearing	5
				Cartesian	6
	5001	2000	534688	bearing	7
				Cartesian	8
3D	11	175	226	depth	9
				disparity	10
		177	232	Cartesian	11
	101	708	1809	depth	12
				disparity	13
		710	1868	Cartesian	14
	1001	3875	19267	depth	15
				disparity	16
			19806	Cartesian	17
	5001	4922	96659	depth	18
				disparity	19
			99332	Cartesian	20

Table 3.3: Characteristics of the simulated data sets used in the experiments.

As it can be seen from the table, the larger the problem becomes, the harder it is for LM to converge to the correct solution, whereas our method succeeds on all tested instances. Also the nonlinearity of the sensor has a great effect on the convergence. In 2D, LM fails if we use a bearing-only sensor and the number of constraints increases. In 3D, using a Cartesian sensor leads to the correct solution for standard approaches, while the same approach fails when using a depth or a disparity model, the former being better than the latter. Our method succeeds in all cases.

3.5 Related Work

Also other researchers in the past focused on solving the incrementally growing SLAM problem. For instances, approaches based on filtering are directly applicable in this case, which we discussed in the related work in Chapter 2. Here, we focus on online variants and put an additional emphasis on multi-level or divide-and-conquer methods.

For example, Kaess *et al.* [101] introduced iSAM, an algorithm which updates a QR factorization by Givens Rotations whenever new information is added as the robot moves through the environment. This variant of the algorithm requires to linearize and re-order the factor to reduce the fill-in in the factorization. Kaess *et al.* [100] recently presented iSAM2 which overcomes these limitations by utilizing the so-called Bayes tree, which fluidly re-linearizes and re-orders the system. While the partial back-substitution done in iSAM2 reduces the computational complexity, the fluid re-linearization may increase the computational demands. The experiments reported by Kaess *et al.* [100] show mixed results when benchmarking the efficiency of the im-

ID	F_{init}	F_{ideal}	Levenberg F_{lm}	Our Approach F_{cond}	# cond.	Condensed Graph F_{sparse} (init/final)
1	$3.351 \cdot 10^{7}$	533.837	555.927	593.097	-	- / -
2	24883	1656	1656	1656	-	- / -
3	$4.825 \cdot 10^{8}$	8088.46	8088.46	8121.86	1031	208938 / 3845.72
4	367552	17982	17982	17982	1328	6778.08 / 594.761
5	$7.056 \cdot 10^{9}$	101483	101483	101483	15218	$2.681 \cdot 10^{6}$ / 43868.3
6	$1.267 \cdot 10^{9}$	205351	205351	205351	17348	$2.202 \cdot 10^{6}$ / 4482.67
7	$7.666 \cdot 10^{10}$	528335	$\mathbf{2.226 \cdot 10^{7}}$	528339	74437	$5.031 \cdot 10^{8}$ / 248863
8	$1.792 \cdot 10^{10}$	$1.056 \cdot 10^{6}$	$1.056 \cdot 10^{6}$	$1.056 \cdot 10^{6}$	85693	$7.792 \cdot 10^{7}$ / 26300
9	5339.7	123.436	123.436	123.436	16	8.69811 / 2.25886
10	7947.67	126.067	126.067	126.067	16	11.6747 / 2.80877
11	2550.35	129.623	129.623	129.623	30	64.0997 / 16.4429
12	$9.490 \cdot 10^{7}$	3008.07	3008.07	3008.07	1207	9452.95 / 330.039
13	$1.308 \cdot 10^{8}$	3006.33	**183734**	3006.33	1207	12263 / 423.092
14	553392	3226.68	3226.68	3226.68	1219	22150 / 937.877
15	$5.477 \cdot 10^{10}$	43646.9	$\mathbf{3.018 \cdot 10^{7}}$	43646.9	14882	$1.631 \cdot 10^{7}$ / 5845.47
16	$3.148 \cdot 10^{12}$	43551.8	$\mathbf{1.167 \cdot 10^{7}}$	43551.8	14882	$2.22 \cdot 10^{7}$ / 7361.63
17	$3.636 \cdot 10^{8}$	45162.9	45162.9	45162.9	15040	$1.899 \cdot 10^{7}$ / 13897.1
18	$1.5684 \cdot 10^{14}$	261146	$\mathbf{2.010 \cdot 10^{9}}$	261146	79142	$5.967 \cdot 10^{8}$ / 35200.3
19	$4.518 \cdot 10^{12}$	261364	$\mathbf{3.239 \cdot 10^{8}}$	261371	79142	$6.304 \cdot 10^{8}$ / 44369.1
20	$8.086 \cdot 10^{9}$	268993	268993	268993	79985	$4.818 \cdot 10^{8}$ / 81782.2

Table 3.4: Summary of the simulated experiments. The column F_{init} gives the error of the initial state, while F_{ideal} corresponds to the global minimum. Within the colmuns F_{lm} and F_{cond} we report the result of a standard LM algorithm and our approach, respectively. We highlighted the instances for which the standard approach failed. # cond. reports the number of condensed measurements generated by our approach. Finally, F_{sparse} corresponds to the initial and final error for the condensed graph.

plementations of iSAM and iSAM2. Furthermore, Rosen *et al.* [179] presents an extension of iSAM which utilizes Powell's Dog-Leg instead of Gauss-Newton for incrementally solving the SLAM problem. Both, Levenberg and Dog-Leg feature better convergence properties in case of a non-smooth surface of the objective function.

By introducing adaptive learning rates, the tree network optimizer by Grisetti *et al.* [76] provides efficient means for online maximum likelihood mapping. The Treemap approach presented by Frese [65] creates a tree leading to logarithmic update time. As we have seen in the experiments, revisiting leafs of the tree, however, introduces additional variables leading to a poor performance on data sets exhibiting a lot of revisits.

Applying standard partition algorithms, for example, the method suggested by Karypis and Kumar [102], yields sub-optimal results as those methods do not exploit the domain knowledge. For example, in bearing-only SLAM we have to verify the observability of the landmarks. A fact not considered by the standard algorithms. Hence, several authors addressed SLAM on different levels of abstraction. For instance, Frese *et al.* [66] introduced multi-level relaxation (MLR), a variant of Gauss-Seidel relaxation that applies the relaxation at different levels of resolution. Ni *et al.* [158] presented TSAM, which employs a divide-and-conquer strategy to partition the original large problem into smaller ones and then utilize these partial solutions to construct a global sparse problem. The solution of the global problem is the arrangement of the partial

solutions (local maps) in the space. In particular, TSAM [158] utilizes nested dissection and an efficient parameterization to efficiently solve large off-core problems involving arbitrary types of variables. Subsequently, Ni and Dellaert [159] extended the divide-and-conquer strategy to Structure from Motion. The approaches presented in this chapter are orthogonal to TSAM and extend it by expressing the solution of the sub-problems by using more general measurement functions that condense most of the relevant information in the local solution. These error functions are user defined, so they can be chosen to be as smooth as possible, regardless of the factors in the input problem. The smoothness of the condensed measurements, makes our approach more robust to poor initial guesses as shown in the experiments.

Furthermore, the ATLAS framework [22] employs a hierarchy with two levels, in which the lower level consists of local maps built by Kalman filters and the higher level performs a global optimization. Likewise, the hierarchical SLAM method by Estrada *et al.* [54] estimates local maps that are re-arranged if the robot revisits them.

Recently, Carlone *et al.* [29] presented a linear approximation which is independent of the initial guess. Their method, however, assumes spherical covariances and is limited to 2D pose-graphs as it exploits the commutative property of rotations in 2D. Huang *et al.* [93] examined the convexity of merging local maps of 2D landmarks. They found out that assuming spherical covariances the alignment of the maps phrased as least squares problem has at most two local minima. A discovery which supports the idea of our approach to apply a divide-and-conquer strategy to mapping to obtain smoother functions.

Several approaches to Visual SLAM [108, 138, 199] employ a similar concept like our approach. Those systems generate a pose-graph representation out of the camera measurements obtained at key frame locations. In particular, Strasdat *et al.* [199] — in addition to building the hierarchy — limit the number of active constraints in the pose-graph layer such that a constant runtime per frame is obtained. All these approaches, however, are restricted to generate pair-wise camera constraints for their pose-graph layer. Furthermore, the usage of a relative coordinate frame as proposed by Sibley *et al.* [189] overcomes the convergence problems when a loop is closed because the relative formulation avoids the error propagation along the whole loop. Their approach can be interpreted as continuous sub-mapping.

3.6 Conclusions

We presented a novel approach for optimizing factor graphs obtained from SLAM or SfM problems. The algorithm is robust to noisy initial guesses and highly nonlinear sensor models. The key idea is to construct an approximation of the original problem having a larger convergence basin by computing condensed measurements from partial solutions, to determine a good initial guess. Our approach can solve problems that cannot be handled by other state-of-the-art methods.

In addition to the improved convergence property our hierarchical method leads to an approach that is able to model the problem at different levels of abstraction. The highest level of abstraction can be optimized fast while providing support for making data associations. This addresses the online characteristics of SLAM for a robot exploring its surrounding environment as we focus the computational resources on the vicinity of the robot. As we showed in the experiments, this leads to an efficient method for solving the SLAM problem online.

Furthermore, our approach employing the hierarchical strategies was successfully applied within the EUROPA project [55] for obtaining large-scale maps, as we will describe in Chapter 9. The overall approach is accurate, efficient, overcomes singularities, provides a hierarchical representation, and outperforms a series of state-of-the-art methods.

Chapter 4

Evaluating the Accuracy of Graph-Based SLAM

In this chapter, we address the problem of creating a benchmark for comparing SLAM approaches. We propose a framework for analyzing the results of SLAM approaches based on a metric for measuring the error of the corrected trajectory. The metric considers the relative relations between poses and does not rely on a global reference frame. The idea is related to graph-based SLAM approaches in the sense that it considers the energy needed to deform the trajectory estimated by a SLAM approach to the ground truth trajectory. Our method enables us to compare SLAM approaches that use different estimation techniques or different sensor modalities since all computations are made based on the corrected trajectory of the robot. This allows us to evaluate the accuracy of our graph-based SLAM method against other approaches to SLAM. To this end, we performed the benchmark on a collection of data sets frequently used in the SLAM community.

• • • • • • • • • • •

Throughout Chapter 2 and Chapter 3 we have presented our approaches for addressing SLAM by its graph-based formulation. During the evaluation of the accuracy of our optimization algorithms, which can serve as back-ends in a SLAM system, we focused on the value of the objective function since it is the most interesting aspect. Furthermore, we evaluated the runtime in batch and online settings as well as the convergence properties of our approach compared to other state-of-the-art methods which employ a similar methodology. In the following, we want to provide empirical evidence that our graph-based approach yields accurate models of the environment. To this end, we suggest a metric that allows us to compare a heterogeneous set of SLAM approaches. Particularly, we compare the results of graph-based SLAM with other state-of-the-art methods for SLAM.

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

kind of evaluation becomes more and more difficult as new SLAM approaches show increasing capabilities and thus large-scale environments are needed for evaluation. Therefore, there is a strong need for methods allowing objective comparisons of different approaches. Ideally, such a method is capable of performing comparisons between mapping systems that apply different estimation techniques and operate on different sensing modalities.

In this chapter, we propose a technique for comparing the result of SLAM algorithms. This allows us to assess the quality of the maps generated by our graph-based mapping system compared to other approaches. The proposed metric is based on an idea that is actually similar to the concept of the graph-based SLAM approaches (see Chapter 2). It uses the energy that is (virtually) needed to deform the trajectory estimated by a SLAM approach into the ground truth trajectory as a quality measure.

Our metric operates based on relative geometric relations between poses along the trajectory of the robot. This exploits that estimating a model of the environment is straightforward given the poses of the robot. This principle is also exploited when realizing an efficient particle filter for SLAM [77, 154]. Our metric allows us to objectively compare our graph-based approach to other methods based on data sets that are frequently used in the robotics community, such as the MIT Killian Court or the Intel Research Lab data set. The disadvantage of our method is that it requires manual work to be carried out by a human that knows the topology of the environment. The manual work, however, has to be done only once for a data set and then the evaluation requires only minimal effort. Therefore, we provide a web page that hosts such manually matched relations for existing log files[1].

The remainder of this chapter is organized as follows. After presenting the key idea and advantages of our metric in the next section, we briefly discuss how to apply the metric to SLAM algorithms that do not estimate the full trajectory taken by the robot in Section 4.2. In Section 4.3 we present our experimental evaluation on a collection of publicly available data sets. Finally, in Section 4.4 we provide an overview about the related work.

4.1 Metric for Benchmarking SLAM Algorithms

To evaluate the performance of a SLAM algorithm, we focus on the estimate of the trajectory instead of comparing the map estimate. The advantage of this is twofold. We can, for example, compare an approach generating a grid-based occupancy map with a feature-map generated by another approach. Furthermore, the comparison can be conducted independent of the sensor modalities of the robot and the underlying estimation technique, e.g., the metric can be applied to compare our laser-based framework presented in Chapter 2 with a vision-based FastSLAM.

4.1.1 The Metric

Let $\mathbf{x}_{1:T}$ be the poses of the robot as they are estimated by a SLAM algorithm from time step 1 to T, where $\mathbf{x}_t \in SE(2)$ or $SE(3)$. Let $\bar{\mathbf{x}}_{1:T}$ be the reference poses of the robot, ideally the true poses. An error metric, which compares the poses in the global reference frame, could be defined as

$$\varepsilon(\mathbf{x}_{1:T}) = \min_A \sum_{t=1}^{T} (\mathbf{x}_t \ominus A\bar{\mathbf{x}}_t)^2, \tag{4.1}$$

[1] http://ais.informatik.uni-freiburg.de/slamevaluation

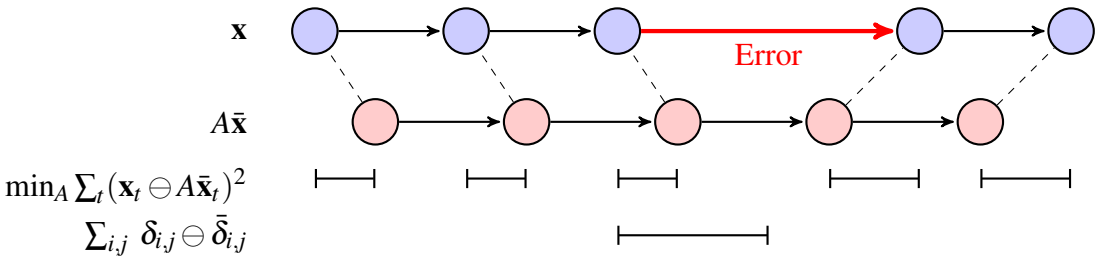


Figure 4.1: Example where the metric in Eq. 4.1 is suboptimal. The robot moves along a straight line and after n poses, it makes an error (bold arrow) but then continues without any further error. Both parts are perfectly mapped and only the connection between both submaps contains an error.

where A is a rigid body transformation and the best one, which minimizes the alignment error between the two trajectories, can be obtained in closed form [167, 212]. Furthermore, see Eq. 2.57 and Eq. 2.58 for the definition of $\ominus$ and $\oplus$, respectively.

We claim that this metric is suboptimal for comparing the results of different SLAM algorithms. To illustrate this, consider the example illustrated in Figure 4.1. Here, a robot travels along a straight line. Let the robot make perfect pose estimates in general but one error somewhere along the line. The part before and after the estimation error are perfectly mapped. According to Eq. 4.1, the error of this estimate changes with every node that is added. Thus, the error depends on the point in time where the robot made an estimation error. This is a result of the comparison between the reference trajectory and the result of the SLAM algorithm in global coordinates in Eq. 4.1.

In contrast to this, we propose to use a metric that considers the deformation energy that is needed to transform the estimate into the ground truth. This can be done — similar to the ideas of graph-based mapping presented in Chapter 2 — by considering the poses as masses and connections between them as springs. Thus, our metric is based on the *relative* transformation between poses. Instead of comparing $\mathbf{x}$ to $\bar{\mathbf{x}}$ we do the operation based on relative transformations. To this end, let $\delta_{ij} = \mathbf{x}_j \ominus \mathbf{x}_i$ be the relative transformation that moves the node $\mathbf{x}_i$ onto $\mathbf{x}_j$. Likewise, let $\bar{\delta}_{ij}$ be the transformation based on $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_j$. Hence, we can compare δ and $\bar{\delta}$ by

$$\varepsilon(\delta) = \frac{1}{|\mathscr{R}|} \sum_{\langle ij \rangle \in \mathscr{R}} (\delta_{i,j} \ominus \bar{\delta}_{ij})^2 \tag{4.2}$$

$$\tilde{\varepsilon}(\delta) = \frac{1}{|\mathscr{R}|} \sum_{\langle ij \rangle \in \mathscr{R}} trans(\delta_{ij} \ominus \bar{\delta}_{ij})^2 + rot(\delta_{ij} \ominus \bar{\delta}_{ij})^2, \tag{4.3}$$

where $\mathscr{R}$ is the set of relations and $trans(\cdot)$ and $rot(\cdot)$ are used to separate the translational and rotational components. We suggest to provide both quantities individually. In this case, the error in Figure 4.1 will be consistently estimated no matter where the error occurred.

Our error metric, however, leaves open which relative displacements $\bar{\delta}_{ij}$ are included in the summation in Eq. 4.3. Evaluating two approaches based on a different set of relative pose displacements will obviously result in two different scores. The set of reference relations $\mathscr{R}$ can be defined to highlight certain properties of an algorithm, for example, by including relations obtained from a blueprint of the building to be mapped.

It should be noted that the metric presented here also has drawbacks. First, the metric — as we defined it — only evaluates the mean estimate of the SLAM algorithm and does not consider its estimate of the uncertainty. Second, it misses a probabilistic interpretation as the Fisher information would realize, see, for example, the work by Censi [31] on the achievable accuracy for range finder-based localization.

4.1.2 Obtaining the Set of Reference Relations

In practice, the key question regarding Eq. 4.3 is how to determine the true relative displacements between poses. Obviously, the true values are only accessible in simulation, where it is trivial to derive the exact relations. We can, however, determine close-to-true values by using the information recorded by a mobile robot and the background knowledge of the human recording the data sets. This, of course, involves manual work, but from our perspective it is the best method for obtaining such relations if no ground truth is available.

While we in the remainder of this chapter focus on laser range finder, the metric is actually independent of the sensor. If the robot is currently not equipped with a laser scanner, there are two solutions. The relation can be obtained by either temporarily mounting a laser on the robot or by developing means for accurately matching the relative transformation between two poses with the sensor modality currently available on the robot.

In our work, we propose the following strategy. First, one seeks for an initial guess about the relative displacement between poses, for example, by scan-matching. Later a human can evaluate the initial guess and reject wrong relations due to his knowledge about the environment. Subsequently, we suggest to manually refine the relations. This manual processing, which is carried out with an appropriate user interface, allows us to remove outliers in the range data degrading the matching result.

In addition to the relative transformations added upon visibility and matching of observations, one can directly incorporate additional relations resulting from other sources of information, for example, aerial images, as we will present in Chapter 6.

4.2 Algorithms without Trajectory Estimates

Our goal is to compare our graph-based formulation with a particle filter and scan-matching as a baseline. Hence, we have a trajectory at hand for benchmarking. On the other hand, however, a family of approaches does only estimate the current location of the robot in each time step. For example, an approach based on an Extended Kalman filter typically drops the full trajectory to lower the computational load by marginalizing out the older poses [205]. Hence, it is worth to also discuss how to apply our metric in this case.

We see two solutions to overcome this problem: First, one can recover the trajectory by localizing the robot in the map. This step can exploit the data association estimated by the SLAM approach. Second, in some settings this strategy can be difficult and one might argue that a comparison based on the landmark locations is more desirable. In this case, one can apply our metric as well by operating on the landmark locations instead of on the poses of the robot. The disadvantage of this approach is that the data association between estimated landmarks and ground truth landmarks is not given. For artificial landmarks the data association could be performed by a human as done by Kurlbaum and Frese [126]. The same might be intractable for image features like SIFT [142] or SURF [16].

4.3 Experimental Evaluation

For the evaluation, we consider the following mapping approaches: First, we applied incremental scan-matching as a baseline approach. Scan matching, here using the approach of Censi [30], incrementally computes an open loop maximum likelihood trajectory of the robot by matching consecutive scans. Such a technique is often employed in more sophisticated approaches to

Trans. error $[m^2]$	Scan Matching	RBPF (50 part.)	Graph-Based
Intel	0.136 ± 0.277	0.011 ± 0.034	0.002 ± 0.004
MIT	19.85 ± 59.84	$0.164 \pm 0.814^{\ddagger}$	0.006 ± 0.029
CSAIL	0.117 ± 0.728	$0.005 \pm 0.013^{\ddagger}$	0.0001 ± 0.0005
FR 79	0.249 ± 0.687	$0.006 \pm 0.020^{\ddagger}$	0.005 ± 0.011

(a) translational error

Rot. error $[deg^2]$	Scan Matching	RBPF (50 part.)	Graph-Based
Intel	25.8 ± 170.9	36.7 ± 187.7	24.0 ± 166.1
MIT	25.4 ± 65.0	$0.9 \pm 1.7^{\ddagger}$	0.9 ± 0.9
CSAIL	22.3 ± 111.3	$1.9 \pm 17.3^{\ddagger}$	0.01 ± 0.04
FR 79	7.3 ± 14.5	$0.7 \pm 2.0^{\ddagger}$	0.7 ± 1.7

(b) rotational error

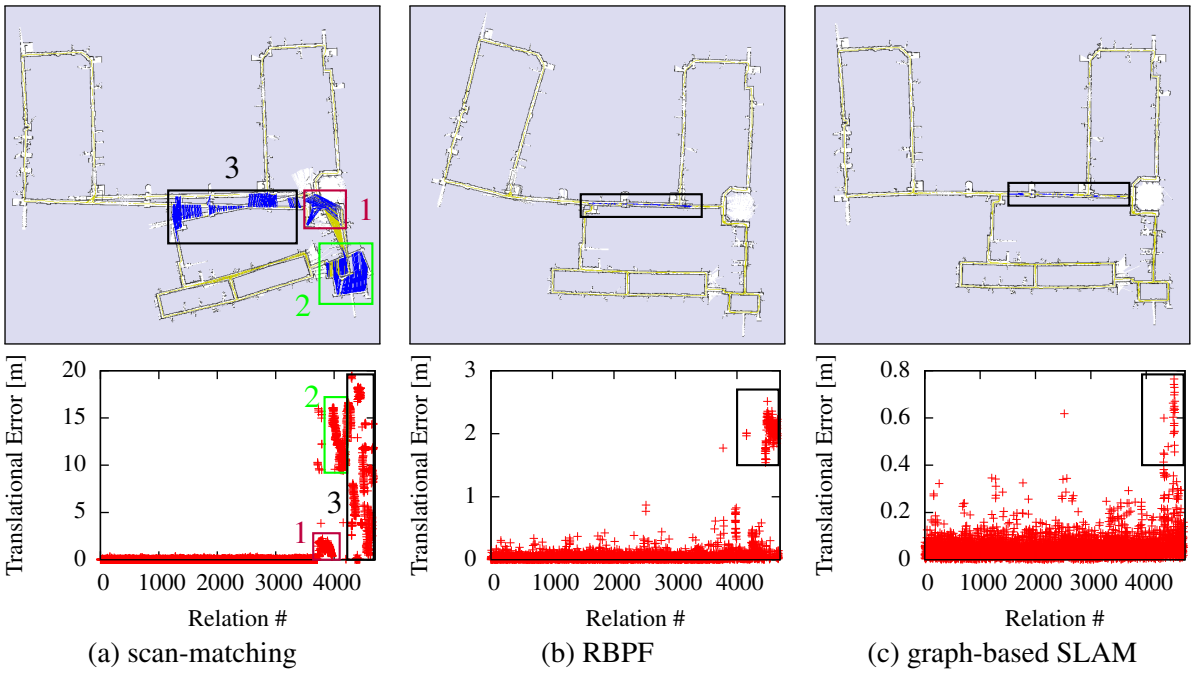
Table 4.1: Quantitative results of different approaches/datasets. For the instances marked with ‡ scan matching has been applied as a preprocessing step.

improve the odometry. Second, we used GMapping [77] which is a mapping system based on a Rao-Blackwellized Particle Filter (RBPF) for learning grid maps. It estimates the posterior over maps and trajectories by means of a particle filter, where each particle carries its own map. Third, we selected an approach that addresses the SLAM problem by its graph-based formulation as described in Chapter 2. Note that we did not include the hierarchical method proposed in Chapter 3 in the comparison because the standard graph-based SLAM method converges to the same solution on each of the data set considered here. The goal of the experiments is to highlight the advantages of the graph-based mapping formulation compared to other methods.

To compare the state-of-the-art methods, we selected a set of data sets which is representative for different kinds of environments. We extracted the relations between robot poses using the method described above by manually validating every single observation between a pair of poses. In detail, we take the following data sets into account. As a challenging indoor corridor-environment with a non-trivial topology including nested loops, we selected the MIT Killian Court data set. As typical office environments with a significant level of clutter, we selected the data set of building 079 at the University of Freiburg, the Intel Research Lab data set, and a data set acquired at the CSAIL at MIT.

We let each algorithm process all benchmark data sets mentioned above. A condensed view of the performance of each algorithm is given by the averaged error over all relations. In Table 4.1a, we provide an overview on the translational error of the various algorithms, while Table 4.1b shows the rotational error. As expected, it can be seen that the more complex algorithms (RBPF and graph-based SLAM) usually outperform scan matching. This is mainly caused by the fact that scan matching only optimizes the result locally and will introduce topological errors in the maps, especially when large loops have to be closed. On average, graph-based mapping seems to be slightly better than an RBPF for mapping.

In the following, we present a detailed evaluation for two of the data sets, where we highlight the errors in the map and the corresponding relations in our map. This enables us to better highlight the advantages of graph-based SLAM for estimating a model of large-scale environments.



(a) scan-matching (b) RBPF (c) graph-based SLAM

Figure 4.2: The MIT Killian Court data set. The reference relations are depicted in light yellow. The results of (a) scan-matching, (b) RBPF using 50 samples, and (c) a graph-based approach. The regions marked in the map (boxes and dark blue relations) correspond to regions in the error plots having high error.

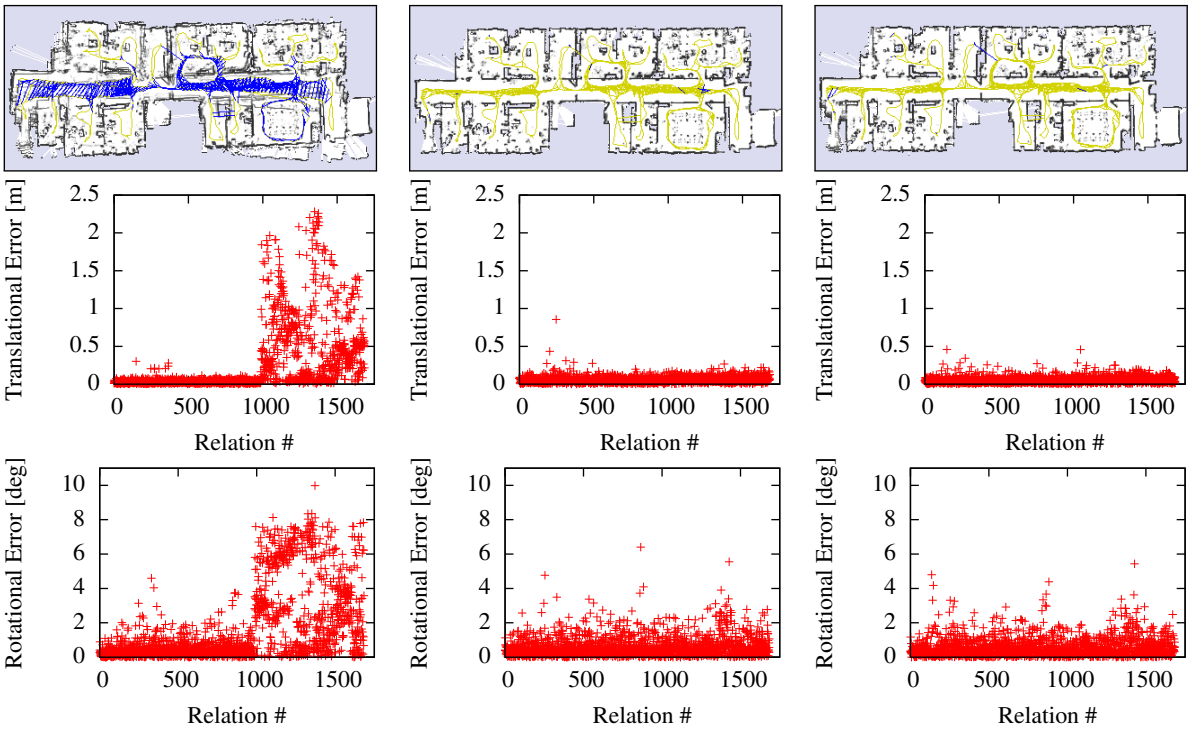
4.3.1 MIT Killian Court

In the MIT Killian Court data set (also called the infinite corridor data set), the robot mainly observed corridors with only few structures that support accurate pose correction. The robot traversed multiple nested loops – a challenge especially for the RBPF-based technique. We extracted close to 5,000 relations between nearby poses that are used for evaluation. Figure 4.2 shows three different results and the corresponding error distributions to illustrate the capabilities of our method. Regions in the map with high inconsistencies correspond to relations having a high error. The absence of significant structure along the corridors results in a small or medium re-localization error of the robot in all compared approaches. In sum, we can say the graph-based approach outperforms the other methods and that the score of our metric reflects the impression of a human about the quality of the map obtained by visually inspecting the mapping results. The visual inspection by a human who knows the structure of the building takes into account that the vertical corridors in the upper part are supposed to be parallel.

4.3.2 Freiburg Indoor Building 079

The building 079 of the University of Freiburg is an example for a typical office environment. The building consists of one corridor which connects the individual rooms. Figure 4.3 depicts the results of the individual algorithms (scan matching, RBPF, graph-based). In the first row of Figure 4.3, the relations having a translational error greater than 0.15 m are highlighted in dark blue.

In Figure 4.3a showing the scan matching result, the relations plotted in blue are generated when the robot revisits an already known region. These relations are visible in the corresponding error plots (Figure 4.3a, second and third row). As can be seen from the error plots, these relations with a number greater than 1,000 have a larger error than the rest of the data set. The



(a) scan-matching (b) RBPF (c) graph-based SLAM

Figure 4.3: The Freiburg Indoor Building 079 data set. The results of (a) scan-matching, (b) RBPF using 50 samples, and (c) a graph-based approach. Within each column, the top image shows the map, the middle plot is the translational error and the bottom one is the rotational error.

fact that the pose estimate of the robot is sub-optimal and that the error accumulates can also be seen by the rather blurry map and by double-occurrences of some walls. In contrast to that, the more sophisticated algorithms, namely RBPF and graph-based mapping, are able to produce consistent and accurate maps in this environment (see Figure 4.3a and b). This is confirmed by our metric since only very few relations show an increased error (illustrated by dark blue relations). Hence, there is no clear distinction between the performance of an RBPF and a graph-based approach on this data set.

4.4 Related Work

Activities related to performance metrics for SLAM methods, as the work described in this chapter, can roughly be divided into three major categories: First, competitions where robot systems are competing within a defined problem scenario (such as playing soccer), second, collections of publicly available data sets that are provided for comparing algorithms on specific problem, and third, related publications that introduce methodologies and scoring metrics for comparing different methods.

To perform comparisons between robots, numerous robot competitions have been initiated in the past. The most prominent one are probably RoboCup for robots playing soccer or rescuing victims after a disaster, and cars driving autonomously at the DARPA Urban Challenge. Nevertheless, competition settings are likely to generate additional noise due to different hardware and software. Depending on the competition, approaches are often tuned to the settings addressed in the competitions and it is unclear whether the SLAM part fulfills the requirements one seeks for. Furthermore, such competitions require to run a whole system and the goal is an

overall good performance, e.g., the result of the SLAM algorithm might be worsened because the approach has to cope with a limited processing time.

Some steps towards benchmarking navigation solutions have been presented in the past. Amigoni *et al.* [6] presented a general methodology for performing experimental activities in the area of robotic mapping. They suggest a number of issues that should be addressed when experimentally validating a mapping method. If ground truth data is available, they suggest to utilize the Hausdorff metric for map comparison.

Wulf *et al.* [216] proposed the idea of using manually supervised localization for matching 3D scans against a reference map. They suggest to generate the reference maps from independently created CAD data. The comparison between the generated map and the ground truth has been carried out by computing the Euclidean distance and angle difference of each scan, and plotting these over time. As we have argued above, comparing the absolute error between two tracks might not yield a meaningful assertion.

Balaguer *et al.* [12] utilize the USARSim robot simulator and a real robot platform for comparing different open source SLAM approaches and they propose that the simulator engine could be used for systematically benchmarking different approaches of SLAM. They do, however, not provide a quantitative metric for comparing generated maps with ground truth. Their comparisons were carried out by visual inspection.

Instead of utilizing a simulator, several researchers focused on obtaining (close to) ground truth estimates. For example, Ceriani *et al.* [34] describe a system for collecting ground truth for indoor laser-based SLAM. Sturm *et al.* [202] utilize a motion capture system to evaluate RGB-D SLAM approaches.

Evaluating the performance of optimization approaches can be done by analyzing the value of the objective function and the runtime of the approach, as we have seen, for example, in Chapter 2. Instead of considering the objective function, Olson and Kaess [167] suggested to evaluate the quality of the map based on the distance of the nodes to their ground truth location. The reason for this is the nonlinear shape of the objective function for graph-based SLAM. A small difference in the function value may correspond to a completely different map. As the method requires ground truth data, it can be utilized for evaluating simulated data, but how to apply it for real-world data sets remains unclear. Furthermore, the metric is geared towards evaluating optimization algorithms.

4.5 Conclusions

In this chapter, we presented a framework for comparing the results of SLAM approaches with the goal to create objective benchmarks. We proposed a metric for measuring the error of a SLAM system based on the estimate of the trajectory. Our metric uses only relative relations between poses and is motivated by the energy needed to transform an estimate into ground truth. This overcomes serious shortcomings of approaches using a global reference frame to compute the error.

Additionally, we provide an error analysis for three mapping systems using the metric and data sets. The results show that graph-based SLAM obtains accurate maps, whose quality is often superior to other state-of-the-art methods. To facilitate the evaluation for other researchers, we released the set of manually verified relations for a collection of publicly available data sets on the web. This collection of manually verified relations has been considered by several researchers as baseline for evaluating the performance of their SLAM algorithms or inspired their own evaluation for different data sets also featuring other sensor modalities, e.g., Carlone *et al.* [29], Strasdat *et al.* [199], Sünderhauf and Protzel [203], and Tipaldi *et al.* [208].

Chapter 5

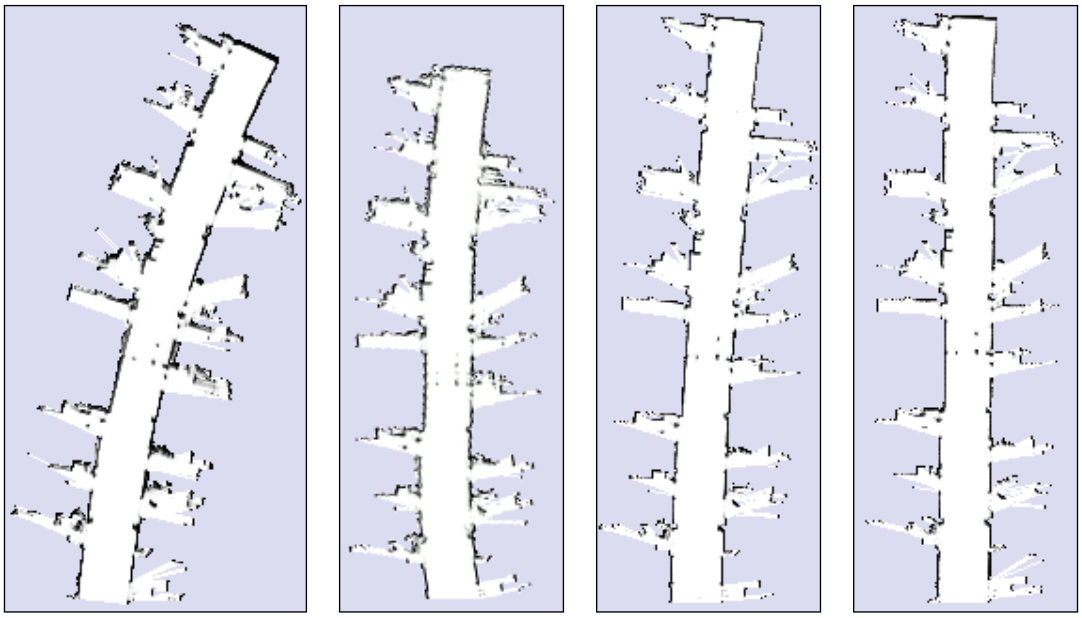
Simultaneous Parameter Calibration, Localization, and Mapping

The calibration parameters of a mobile robot play a substantial role in navigation tasks. Often these parameters are subject to variations that depend either on changes in the environment or on the load of the robot. Up to now, we assumed that we know the value of the parameters. In this chapter, we propose an approach to simultaneously estimate a map of the environment, the position of the on-board sensors of the robot, and its kinematic parameters. To this end, we will extend the graph-based mapping technique, which we have presented and evaluated in the previous chapters. Our method requires no prior knowledge about the environment and relies only on a rough initial guess of the parameters of the platform which are easy to obtain. The proposed approach is able to estimate the parameters on-line and it adapts to non-stationary changes of the configuration. We tested our approach in simulated environments and on a wide range of real-world data using different types of robotic platforms.

• • • • • • • • • • •

We so far considered approaches which allow a robot to estimate a model of an environment by means of a SLAM algorithm. Additionally, we compared our graph-based approach with other state-of-the-art methods highlighting the good performance of our approach. For realizing our approach to SLAM, however, we ignored that we rely on some underlying parameters of the robotic platform, which we assumed as known. Furthermore, many other approaches for navigation such as localization, path planning, and motion control also rely on the knowledge of specific parameters of the robot. These parameters typically include the position of the sensor on the platform or specific aspects of the kinematic model that translates encoder ticks into a relative movement of the mobile base. In the following, we present a method which allows us to estimate these parameters while the robot performs SLAM. Hence, in reference to SLAM we call our method Simultaneous Parameter Calibration, Localization, and Mapping.

The influence of the calibration parameters on the accuracy of state estimation processes can be substantial. For instance, an accurate calibration of the odometry can substantially improve the expected accuracy of the motion prediction by reducing the search space of the algorithms that provide the motion estimates. Figure 5.1 shows a motivating example. Here, we ran a



(a) (b) (c) (d)

Figure 5.1: (a) Map obtained based on the raw uncalibrated odometry of a robot with unevenly inflated tires traveling along a corridor. (b) The result of applying a scan-matching algorithm with a large search space to account for the uncalibrated odometry leads to the shortened map. (c) A restriction of the search space is not able to fully correct the errors. (d) Applying the correct calibration together with a small search space leads to an accurate estimate.

scan-matching algorithm given the odometry measurements of a robot that moves along a corridor. Since the corridor is not rich in features, the scan matcher yields solutions that are highly ambiguous along the direction of the corridor. As a result, scan-matching approaches tend to underestimate the length of corridors [197]. To limit this effect one can restrict the search space of the scan-matcher to a small region around the position predicted by odometry. This reduces the computational requirements but requires a highly accurate calibration of the odometry.

While 2D range scans do not provide highly distinguishable features, camera images allow us to extract discriminable features, e.g., Scale-Invariant Feature Transform (SIFT) [142] or Speeded Up Robust Feature (SURF) [16]. The best match of a feature is not necessarily the correct match, though. This leads to a set of feature matches which contains a substantial amount of outliers. Typically, an algorithm based on Random Sample Consensus (RANSAC) [60] is employed to robustly estimate the inlier set despite the large amount of outliers. If an initial guess of the camera motion is available, for instance, by the odometry of the robot, we are able to restrict the search for the match to the most likely area in the image. This in turn increases the likelihood of finding the correct match. Figure 5.2 visualizes the number of correct feature matches returned by the matching algorithm with and without access to prior information on the motion of the camera. As we can see, the number of correct matches using the prior information is substantially larger compared to an uninformed matching algorithm. A larger number of correct matches in principle yields a better estimate for the transformation between the two frames. To accurately predict the motion of the camera based on the odometry, the robot greatly benefits from knowing the parameters of the odometry and the position of the sensor.

To obtain these parameters it is common to either rely on the specifications of the platform, to manually measure them, or to run ad-hoc calibration procedures before the mission of the robot is started. The latter solution is typically the most accurate and robust since it can ex-

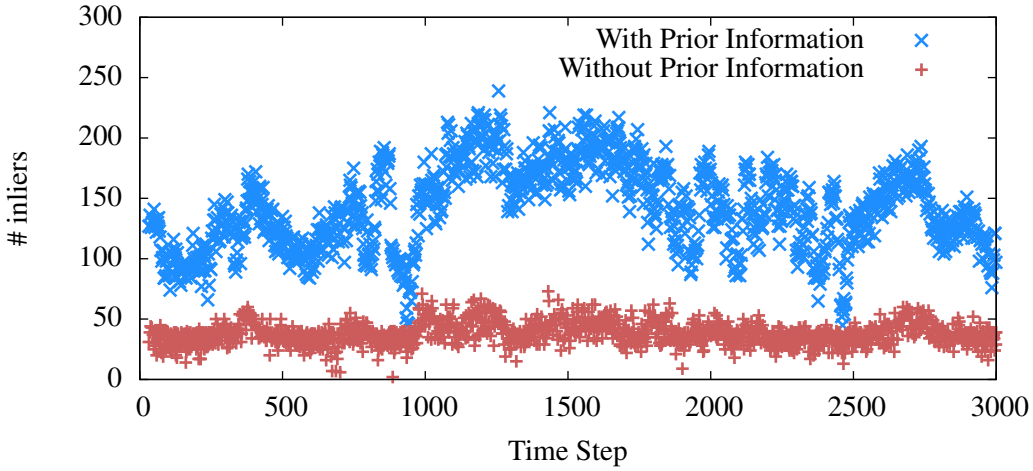


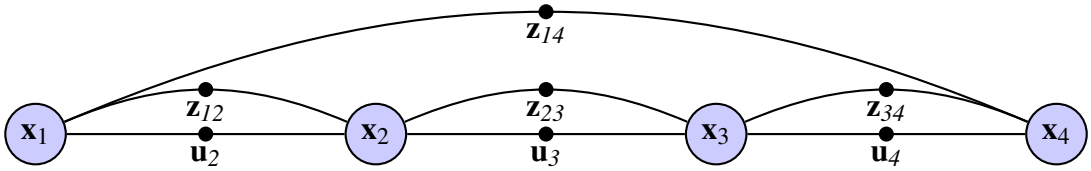
Figure 5.2: The number of inliers of a matching algorithm operating on feature matches. A matching algorithm that considers the prior information about the motion of the camera achieves a larger set of inliers compared to an algorithm which has no access to this information.

ploit the a priori knowledge of a calibration pattern to infer reasonable initial guesses for the parameters. Ad-hoc calibration procedures, however, suffer from two main drawbacks. They are not able to estimate non-stationary parameters and they need to be repeated whenever there is a potential change in the configuration of the robot. On the one hand, these changes will happen unavoidably during the lifetime of the robot as a consequence of the wearing of mechanical parts. On the other hand, we frequently observe sudden changes in these parameters as a consequence of particular events. As an example, the odometry parameters depend on both the distribution of the load on the platform and the type of surface the robot moves upon. These external quantities affect the effective wheel radii as well as the accuracy of the odometry prediction. When the robot carries a load the odometry will change, and similarly when it moves from carpet to concrete. One solution to dynamically estimate these parameters is to treat them as hidden state variables that have to be estimated together with the map and the position of the robot. To this end, we could use special equipment such as an external position tracking device. More promising, however, are calibration procedures that only rely on the data gathered by the robot and do not require any preparation or additional information.

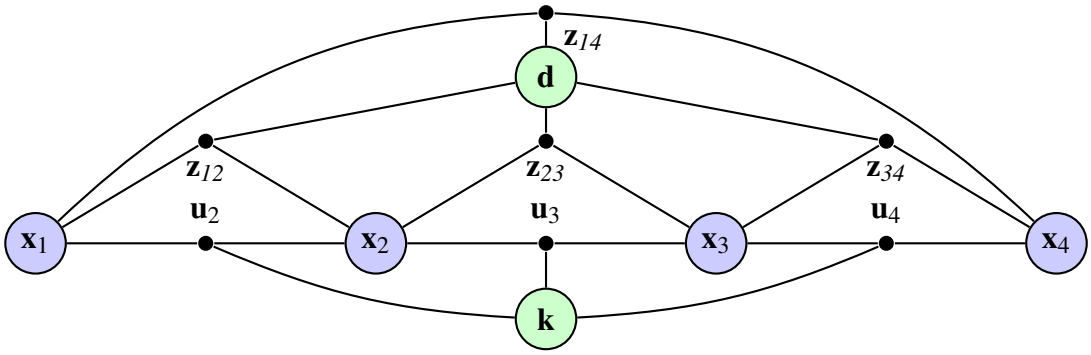
We present an approach to estimate the calibration parameters of a robot equipped with an on-board sensor and wheel encoders while it performs SLAM. The core idea of our method is to treat the map estimate as a calibration pattern and to constantly refine the estimates of the map, the trajectory, and parameters of the robot through a least squares procedure. If the map and the robot positions are known, our method behaves as a standard least squares approach for parameter calibration.

Note that when we augment the problem with the calibration variables it cannot be described anymore by a graph but instead requires a hyper-graph, as indicated in Figure 5.3. This is due to the fact that a measurement does not only depend on a pair of variables (the connected nodes) but rather on a triplet (the nodes and the calibration parameters). Therefore, we extend the standard graph-based SLAM optimization framework to handle this class of problems. Our approach is able to simultaneously estimate the map of the environment and calibrate the parameters of a robot in a continuous manner. We do not require any special preparation for the environment, such as an external tracking system or landmarks to be placed in the designated area.

Our approach allows us to determine these state variables on the fly (e.g., sensor positions



(a) standard graph-based SLAM



(b) simultaneous calibration

Figure 5.3: The factor graph shown in (a) represents a standard SLAM problem. Here, the robot has odometry measurements $\mathbf{u}$ which relate subsequent poses of the robot along with scan-matching estimates $\mathbf{z}$. In particular, a loop closure relates $\mathbf{x}_1$ and $\mathbf{x}_4$ which can be obtained by matching the respective observations. If we augment the SLAM problem by the calibration parameters, we obtain the factor graph depicted in (b). In fact, the graph is a hyper-graph, as the factors connect more than two nodes each.

and odometry calibration). To deal with temporal changes or more in general with interdependencies between the parameters and the other state variables, we estimate the parameters on the most recent data. This approach allows a mobile robot, for example, to estimate a different set of odometry parameters for different regions of the environment and to better model the motion of the robot in these areas. Our approach might additionally be beneficial in a variety of contexts including, for instance, terrain classification. We present evaluations of our approach in simulated and in a wide range of real-world experiments using several robot platforms moving on different types of ground.

The remainder of this chapter is structured as follows. After describing our approach for simultaneously estimating the calibration parameters and the poses of a navigating robot in Section 5.1, we present quantitative simulated and real-world experiments in Section 5.2. Finally, we discuss the related work in Section 5.3.

5.1 Simultaneous Calibration, Localization, and Mapping

Our system relies on the graph-based formulation of the SLAM problem to estimate the maximum-likelihood configuration. In contrast to the traditional SLAM methods we explicitly model that the measurements obtained by the robot are given in different coordinate frames. For example, the odometry of the robot is given by the velocity measurements of its wheels. Applying the forward kinematics of the platform transforms the velocities measured during a time interval into a relative displacement of the platform expressed in the odometry frame. Usually, the robot is equipped with a sensor that is able to perceives features in the environment, e.g., a laser range finder. This sensor is mounted on the robot and obtains measurements in its own coordinate

frame. Thus, a scan-matching algorithm, which aligns two range scans in a common coordinate frame to estimate the ego-motion of the scanner, has to project the computed motion estimate through the kinematic chain of the robot to obtain an estimate for the motion of the platform. As it is not always easy to accurately measure the offset transformation between the base of the robot and the sensor or to determine the parameters for the forward kinematics, we suggest to integrate those into the maximum likelihood estimation process. A rough estimate which serves as initial guess for the parameters is typically easy to obtain. Furthermore, the parameters of the forward kinematics are affected by the wear of the devices during the lifetime of the robot. Our approach is able to re-estimate those changes over the life-time of the robot. Figure 5.4 illustrates the involved parameters which we will describe in detail in the following.

5.1.1 Description of the Hyper-Graph

Whenever the robot obtains a measurement $\mathbf{s}_i$, we add a node $\mathbf{x}_i = (x_i, y_i, \theta_i)^\top$ to the graph. Each node represents the position of the robot at which the according measurement was obtained. Again, $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)^\top$ is the vector of poses up to time T. Furthermore, let $\mathbf{d}$ be the pose of the on-board sensor relative to the coordinate frame of the robot. Finally, let $\mathbf{k}$ be the parameters of the forward kinematics function and $\mathbf{u}_i$ and $\Omega_i^{\mathbf{u}}$ be respectively the motion command and the information matrix which translates the robot from node $i-1$ to i.

The error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{k}, \mathbf{u}_i)$ measures how well the parameter blocks $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, and $\mathbf{k}$ satisfy the odometry measurement $\mathbf{u}_i$. A value of $\mathbf{0}$ means that the constraint is perfectly satisfied by the parameters. For simplicity of notation, we will encode the involved quantities in the indices of the error function:

$$\mathbf{e}^{\mathbf{u}}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{k}, \mathbf{u}_i) \stackrel{\text{def.}}{=} \mathbf{e}^{\mathbf{u}}(\mathbf{x}_{i-1}, \mathbf{x}_i) \stackrel{\text{def.}}{=} \mathbf{e}_i^{\mathbf{u}}(\mathbf{x}). \tag{5.1}$$

The error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x})$ is defined as

$$\mathbf{e}_i^{\mathbf{u}}(\mathbf{x}) = (\mathbf{x}_i \ominus \mathbf{x}_{i-1}) \boxminus K(\mathbf{u}_i, \mathbf{k}), \tag{5.2}$$

where $K(\cdot)$ is the forward kinematics function converting from wheel velocities to a relative displacement of the vehicle. Furthermore, $\ominus$ is the inverse of the the usual motion composition operator $\oplus$ (see Eq. 2.57 and Eq. 2.58) and $\boxminus$ computes the difference while taking into account the different domains of the entities (see Section 2.3.2).

For a robot with a differential drive, which is one of the most common types of robots, the odometry $\mathbf{u} = (v_l, v_r)^\top$ consists of the velocities of the left and the right wheel. The wheel velocities are computed by counting the encoder ticks of the motors during the time step which are multiplied by the respective radii r_l and r_r of the wheels. Furthermore, the distance b between the two wheels has to be known to compute the circular arc on which the robot moves. The relative motion during the time interval Δt is given by

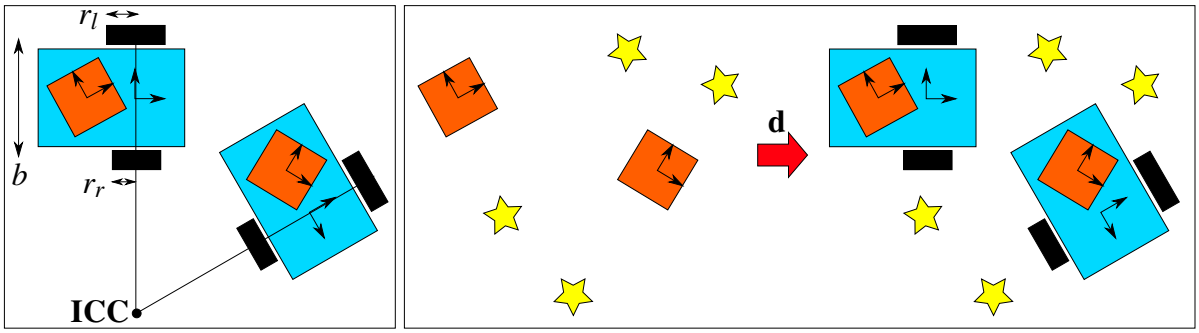
$$K(\mathbf{u}, \mathbf{k}) = \begin{pmatrix} R(\Delta t\, \omega) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} -\mathbf{ICC} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{ICC} \\ \Delta t\, \omega \end{pmatrix}, \tag{5.3}$$

where $R(\cdot)$ is the 2D rotation matrix of its argument,

$$\mathbf{ICC} = \left(0, \frac{b}{2} \frac{r_l v_l + r_r v_r}{r_l v_l - r_r v_r}\right)^\top, \tag{5.4}$$

and

$$\omega = \frac{r_l v_l - r_r v_r}{b}. \tag{5.5}$$



(a) forward kinematics (b) ego-motion of the sensor

Figure 5.4: (a) The parameters $(r_r, r_l, b)^\top$ used to compute the motion of the robot given the wheel velocities. (b) The sensor observations (indicated as yellow stars) allow us to estimate the ego-motion of the sensor and the motion of the robot *given* the offset $\mathbf{d}$ between the coordinate frame of the robot and the sensor.

Thus, the calibration parameter $\mathbf{k} = (r_r, r_l, b)^\top$ for the odometry is a three-dimensional vector. Figure 5.4a shows an illustration of the calibration parameters for the odometry.

Additionally, the error function $\mathbf{e}^{\mathbf{d}}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{d}, \mathbf{z}_{ij})$ measures how well the parameter blocks $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{d}$ satisfy the virtual measurement $\mathbf{z}_{ij}$, which is obtained as the scan-matching result of the observations $\mathbf{s}_i$ and $\mathbf{s}_j$ (see Figure 5.4b). If the three parameters perfectly satisfy the error function, then its value is $\mathbf{0}$. Here, we assume that the laser is mounted without inclination, which is the ideal condition. The error function $\mathbf{e}^{\mathbf{d}}_{ij}(\mathbf{x})$ has the following form:

$$\mathbf{e}^{\mathbf{d}}_{ij}(\mathbf{x}) = \big((\mathbf{x}_j \oplus \mathbf{d}) \ominus (\mathbf{x}_i \oplus \mathbf{d})\big) \boxminus \mathbf{z}_{ij}. \tag{5.6}$$

In Eq. 5.6 we applied the same simplifying notation as defined in Eq. 5.1.

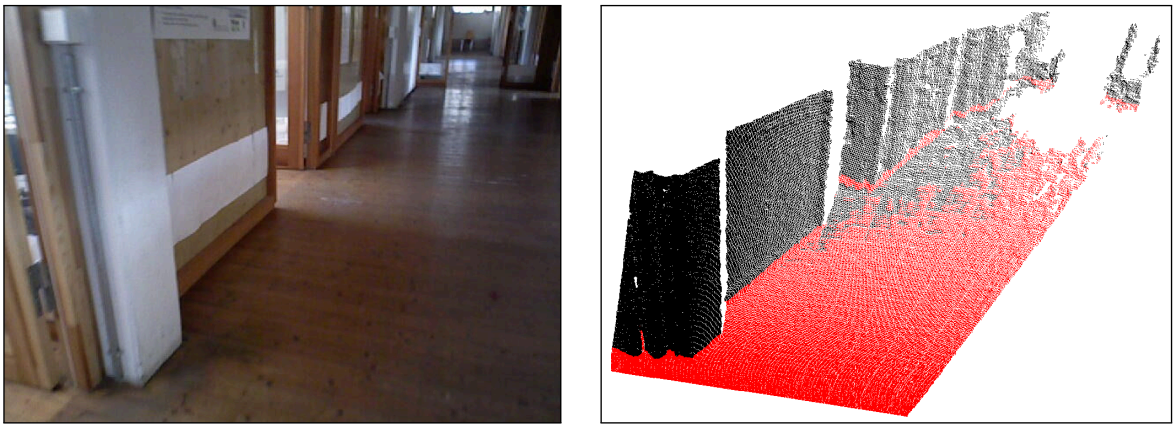
The goal of our maximum likelihood approach is to find the configuration of $\langle \mathbf{x}^*, \mathbf{d}^*, \mathbf{k}^* \rangle$ which minimizes the negative log-likelihood $F(\mathbf{x}, \mathbf{d}, \mathbf{k})$ given all the observations

$$F(\mathbf{x}, \mathbf{d}, \mathbf{k}) = \sum_{\langle i,j \rangle} \mathbf{e}^{\mathbf{d}}_{ij}(\mathbf{x})^\top \Omega^{\mathbf{z}}_{ij} \mathbf{e}^{\mathbf{d}}_{ij}(\mathbf{x}) + \sum_i \mathbf{e}^{\mathbf{u}}_i(\mathbf{x})^\top \tilde{\Omega}^{\mathbf{u}}_i \mathbf{e}^{\mathbf{u}}_i(\mathbf{x}), \tag{5.7}$$

where $\tilde{\Omega}^{\mathbf{u}}_i$ is the projection of $\Omega^{\mathbf{u}}_i$ through the forward kinematics function $K(\cdot)$ via the unscented transformation [96]. Since the projection depends on the estimate of $\mathbf{k}$, we update the projection if $\mathbf{k}$ changes substantially.

Given this formulation we may easily integrate prior knowledge, for example, the result of a previous calibration or the manually — thus non-precisely — measured transformation of the laser. This is possible as long as the prior information can be represented by a Gaussian distribution. Furthermore, state transitions observed by measurements, e.g., the robot actively rotates the laser scanner, can be incorporated.

To estimate the calibration parameters, the trajectory of the robot should introduce measurements that constrain all possible dimensions of $\mathbf{k}$ and $\mathbf{d}$. Clearly, a trajectory only consisting of straight line motions does not allow us to observe the position of the laser. The same holds for a perfectly circular trajectory since the laser could be anywhere on the circle. Both cases are pathologic and can easily be avoided by varying the wheel velocities of the robot. Recently, Censi *et al.* [32] gave a proof that our informal statement holds. They show that two linear independent velocity commands, which lead to two circular arcs with different radii, are sufficient to guarantee the observability of the parameters. In other words, we are able to observe the hidden parameters, for example, by commanding the robot to execute a straight line motion and



(a) camera image (b) extracted ground plane (red points)

Figure 5.5: The range measurements of a depth camera allow us to extract the ground plane. (a) The camera image provided by an RGB-D camera similar to the Microsoft Kinect. Assuming a rough initial knowledge about the orientation of the sensor is known, we are able to extract the ground plane by means of a RANSAC algorithm. (b) The points marked in red constitute the inlier set of the ground plane.

a turn on the spot behavior. Note that the command for a straight motion might in fact produce a circular arc, as we do not know the true wheel radii. Likewise, the velocities sent for turning on the spot may cause a small translation of the robot. Despite these concerns, those commands will allow us to observe the hidden calibration parameters. In our experiments, the parameters were observable all the time, although no special care was taken to guarantee the observability, for example, by modifying the controller setting the wheel velocities. We attribute this to small variations in the normal control algorithm of the robot, which lead to a robot that, for example, never executes a perfectly straight motion if requested to drive along a straight line.

5.1.2 3D On-Board Sensors

In the previous section, we described how to estimate the offset of an on-board 2D range scanner to the odometry center of the robot. With the advent of the Microsoft Kinect an inexpensive alternative exists which provides dense 3D range data along with an RGB image. Such depth cameras emit an infrared light pattern which is received by a camera. Computing the disparity of features in the images allows us — similar to a stereo camera pair — to obtain 3D depth information.

As with 2D range scanners, we are able to estimate the 3D ego-motion $\mathbf{z}_{ij}$ of the camera between two frames i and j by aligning the measurements. Along with the 3D offset of the sensor $\mathbf{c}$ in the coordinate frame of the robot we are able to constrain the motion of the robot:

$$\mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x}) = \big((\text{projectFrom2DTo3D}(\mathbf{x}_j) \oplus \mathbf{c}) \ominus (\text{projectFrom2DTo3D}(\mathbf{x}_i) \oplus \mathbf{c})\big) \boxminus \mathbf{z}_{ij}, \tag{5.8}$$

where projectFrom2DTo3D$(\cdot)$ projects its argument from 2D to 3D, i.e., the pose is extended with a z-coordinate and pitch/roll angles, which are all set to 0.

Eq. 5.8 does not observe the height of the sensor above the ground. Typically, a rough initial guess for the attitude of the sensor is easy to obtain manually. Assuming such an estimate is available, we are able to extract the ground plane observed by the sensor. Figure 5.5 depicts an example in which a robot equipped with an RGB-D camera is driving along a corridor. To extract the ground plane, a RANSAC algorithm samples three points from the point cloud to calculate a candidate plane. If the angle of the normal vector of the candidate lies within a

threshold (20 degrees in our current implementation) around the ground plane assumption given by the prior, we determine the points whose distance to the plane is smaller than 0.1 m. The candidate with the largest inlier set yields our estimate of the ground plane. The distance of the sensor to the extracted ground plane results in our observation for the height of the sensor. Additionally, the measurement $\mathbf{z}_i^p$ of the ground plane allows us to further constrain the attitude of the sensor, namely the rotation of the sensor with respect to the ground plane. Hence, we obtain the additional error term

$$\mathbf{e}_i^{\mathbf{c},p}(\mathbf{c}) = \mathbf{c} \boxminus \mathbf{z}_i^p, \tag{5.9}$$

which constrains the pitch/roll angles of the sensor to match the ground plane observation and the height of the sensor above the ground.

Replacing the sensor error term and adding Eq. 5.9 to Eq. 5.7, we obtain the negative log-likelihood $F(\mathbf{x},\mathbf{c},\mathbf{k})$ given all the observations as

$$F(\mathbf{x},\mathbf{c},\mathbf{k}) = \sum_{\langle i,j \rangle} \mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x})^\top \Omega_{ij}^{\mathbf{z}} \mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x}) + \sum_i \mathbf{e}_i^{\mathbf{u}}(\mathbf{x})^\top \tilde{\Omega}_i^{\mathbf{u}} \mathbf{e}_i^{\mathbf{u}}(\mathbf{x}) + \sum_i \mathbf{e}_i^{\mathbf{c},p}(\mathbf{c})^\top \Omega_i^p \mathbf{e}_i^{\mathbf{c},p}(\mathbf{c}), \tag{5.10}$$

where Ω_i^p represents the information matrix of the ground plane observation. Minimizing Eq. 5.10 yields the poses of the robot, the calibration parameters of the forward kinematics function, and the 3D offset of the range sensor.

5.1.3 Estimation via Least-Squares on a Hyper Graph

To obtain the optimal solution for either Eq. 5.7 or Eq. 5.10 we apply our general optimization framework g^{2}o presented in Chapter 2. This allows us to obtain a numerical solution in a fast end efficient manner. We only require an initialization, which we perform as follows. The poses of the robot, which are represented by $\mathbf{x}$, are initialized via the odometry measurements considering the specifications of the robot as initial values of $\mathbf{k}$. For the position $\mathbf{d}$ of the sensor a rough initial guess is required, which is easy to obtain manually.

From Eq. 5.7 we notice that each of the terms in the sum depends on at most three parameter blocks. More precisely, if the constraint arises from an odometry measurement, it will depend on the connected robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ and by the odometry parameters $\mathbf{k}$. Alternatively, if a constraint arises from a laser measurement, it will depend on the connected robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ and on the laser position $\mathbf{d}$. Accordingly, each of the Jacobians J_k will have only three non-zero blocks, in correspondence of the variables involved by the constraint k. Thus, each of the terms in the sum $\sum_k J_k^\top J_k$ will be a matrix with at most nine non-zero components. Furthermore, we will obtain a number of non-zero entries proportional to the number of constraints. This construction results in a sparse system that can be efficiently solved. Our g^{2}o toolkit, for instance, solves one iteration of a calibration problem having 3,000 nodes in less than 0.01 s using one core of an Intel i7@2.8 GHz.

5.1.4 Monitoring the Convergence

Some calibration parameters may be constant while others change. For example, the laser position $\mathbf{d}$ is constant if the robot has no actuator to move its sensor. Therefore, it is of interest to decide whether enough data has been collected such that one can stop calibrating and reduce the computational demands. To this end, we can consider the approximated Hessian H and compute the marginal covariance of the calibration parameters given the collected measurements. The

(a) (b) (c)

Figure 5.6: The robots used to acquire the real-world data sets: (a) MobileRobots PowerBot, (b) a custom made platform for the EUROPA project, and (c) Pioneer I.

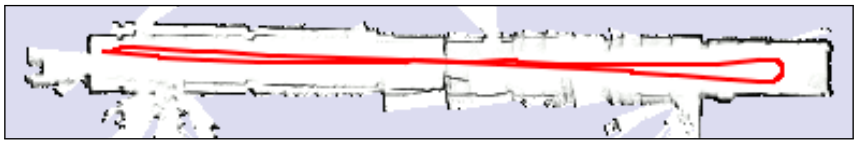
marginal covariance $\Sigma_{\mathbf{d}}$ of the calibration parameter is given by extracting the corresponding block of H^{-1}. As we have previously mentioned in Section 2.3.5, we are able to exploit the properties of H, namely that it is sparse, symmetric, and positive definite. This allows us to compute the desired elements of H^{-1} given the Cholesky factor without computing all elements of the typically dense inverse. As we will show in the experiments, this information enables us to access the quality of the estimate of the parameters.

5.2 Experiments

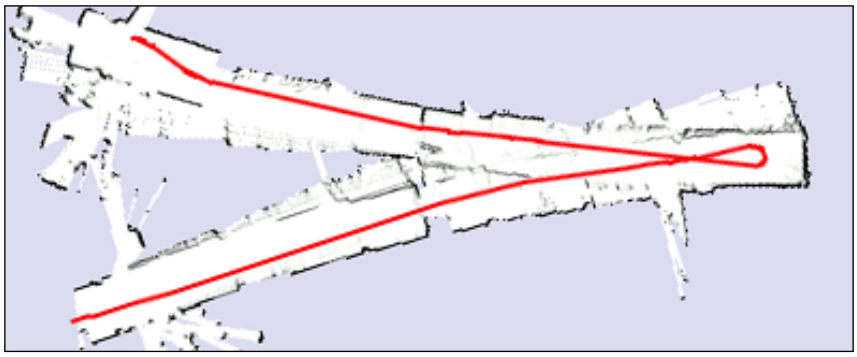
The approach described above has been implemented and evaluated on both simulated and real-world data acquired with a heterogeneous set of robots equipped with laser range finders. Figure 5.6 visualizes the robots we used to collect the real-world data considered in the experiments.

The laser-based SLAM front-end for processing the data is an own implementation of the framework described by Olson [163]. The front-end employs a correlative scan-matcher to estimate the transformation of the laser along with the 3×3 covariance matrix representing the uncertainty of the estimated transformation. The correlative scan-matcher performs an exhaustive search to determine the best fitting alignment for two laser scans within a given search radius. We add a new node to the graph whenever the ego-motion of the laser estimated by the scan-matcher is larger than 0.1 m in translation or 10° in rotation, whichever occurs first.

For estimating the ego-motion of a Kinect depth sensor or a stereo rig, we implemented a pipeline for visual odometry which works as follows. First, we extract visual features [16] from the images. Second, the transformation between images is estimated by a standard 3-point RANSAC algorithm [160]. Finally, a least squares estimate which minimizes the re-projection error of the feature correspondences is performed along with removing feature projections having large errors from the system for an increased robustness against outliers. Again, a node is inserted into the graph whenever one of the above mentioned thresholds is reached.



(a) the odometry of the robot in the standard configuration



(b) the odometry when the robot carries a load

Figure 5.7: Robot driving up and down a corridor. Top: Applying the calibration corresponding to the current configuration of the robot leads to a good odometry estimate. Bottom: If the robot is carrying a load, the same calibration parameters results in a severe drift in the odometry.

5.2.1 Online Odometry Calibration

In real-world scenarios the odometry is affected by different factors. For example, if the robot is carrying a load, the additional weight compresses inflated tires and results in reduced wheel radii. To this end, we used the PowerBot platform (see Figure 5.6a) which has a maximum payload of 100 kg to carry a load of approximately 40 kg. The wheels of the PowerBot are inflated tires whose radii are affected by both the air-pressure of the tires and the total weight of the platform. The load in this set of experiments was intentionally placed on the left hand side of the robot. In a first experiment we recorded data sets in which the robot was either carrying the load or it was operating in its normal configuration. We used one data set for estimating the parameters and a different one for evaluating the odometry calibration parameters. Our approach estimated wheel radii of $r_r^* = 0.1251$ m, $r_l^* = 0.1226$ m for the normal configuration of the robot and $r_r^* = 0.1231$ m, $r_l^* = 0.1223$ m while carrying the load. The difference seems to be small, although it has a substantial effect on the forward kinematics of the robot. Figure 5.7 shows the outcome of applying the estimate of the normal configuration to the robot carrying the load. Applying the wrong calibration parameter has a crucial effect on the trajectory as it is estimated by the odometry. Since the weight of the load is mutable and can be placed in an arbitrary position on the robot, the best performance can be obtained by calibrating the odometry parameters while the robot is operating.

By considering the 50 most recent measurements within a sliding window around the current node we are able to estimate the wheel radii online also when they are subject to change due to external factors. The size of the sliding window was determined empirically. Too few measurements do not allow us to recover the calibration parameters accurately and a large sliding window has an increased latency if the parameters change. For old odometry measurements outside the sliding window we change the error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x})$ to employ a fixed value for $\mathbf{k}$, i.e., we only estimate the physical parameters on the recent data and use the previously estimated parameters to model the odometry error term for older measurements. Figure 5.8 visualizes the estimated wheel radii during an experiment in which the robot had to carry a load placed

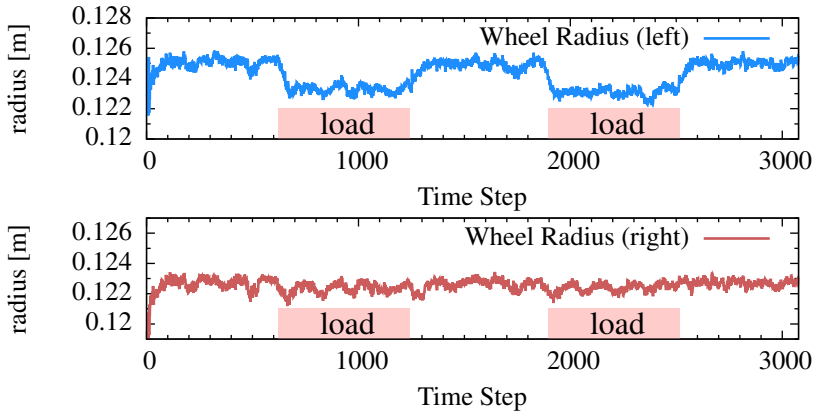
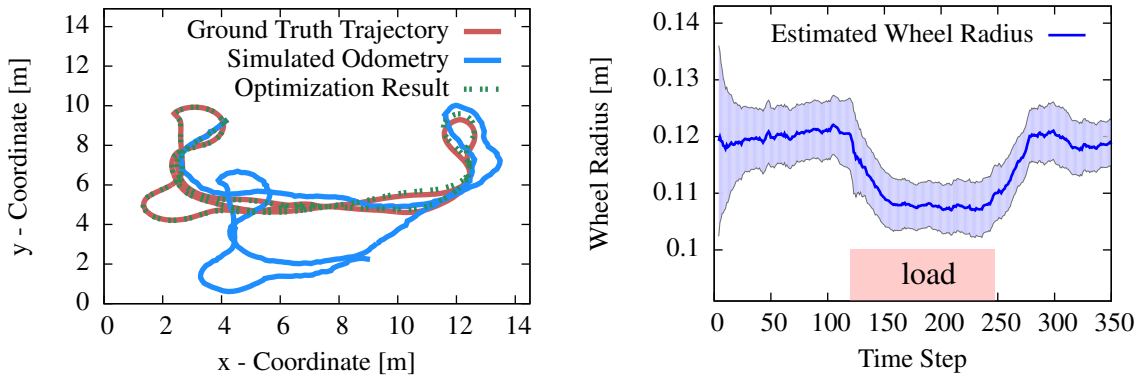


Figure 5.8: Results of the online estimation of the wheel radii. The robot had to carry a load twice which was placed on the left hand side of the platform leading to a compression of the left wheel.



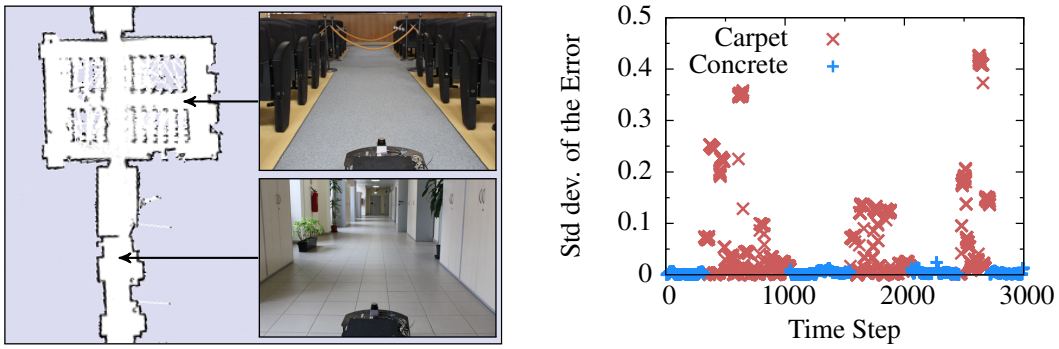
(a) simulated trajectory

(b) online calibration result

Figure 5.9: (a) The simulated robot trajectory. (b) Estimating the wheel radii online based on the most recent observations. Here, the robot was carrying a load during the time interval $[120, 240]$ which leads to compressed wheels having smaller radii.

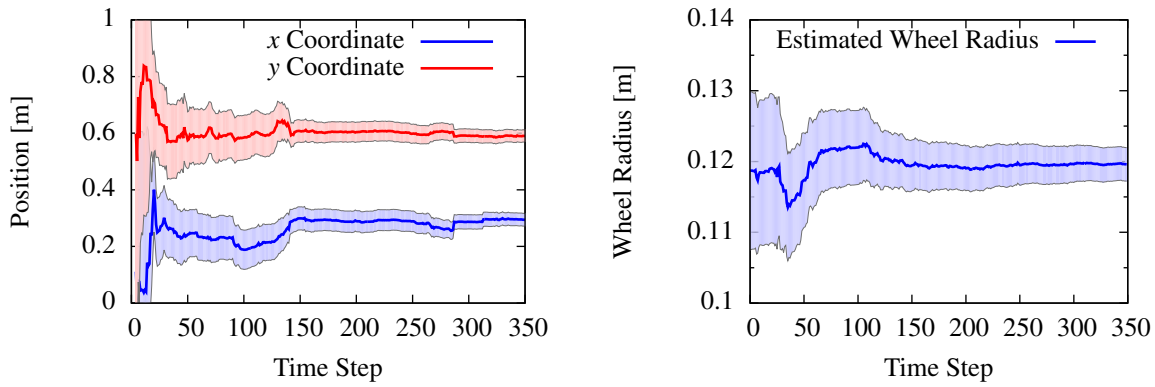
on the left hand side of the platform. The robot was carrying the load during the intervals $[600, 1250]$ and $[1865, 2530]$. Using our approach we are able to correctly estimate the wheel radii independent of the load carried by the robot along with the maximum likelihood map of the environment.

As it is hard to obtain ground truth data for real-world data sets, we simulated a robot traveling along the trajectory depicted in Figure 5.9a. The environment in which the robot operates consists of two rooms connected by a hallway. Additionally, a laser range finder similar to a SICK LMS is simulated by performing ray-casting operations in the simulation environment. The simulator allows us to directly judge the quality of the calibration results. The odometry measurement and the range measurements obtained by the robot are perturbed by Gaussian noise. Within a simulation run we modeled a robot carrying a weight which we simulated having the effect of a reduction of the wheel radius from 0.12 m to 0.108 m. The robot carries the load during the time interval $[120, 250]$. Figure 5.9b depicts the results of the online calibration based on the most recent measurements. As we can see, the system is able to represent the compressed wheels and the estimate corresponds well to the ground truth given by the simulator. Furthermore, the trajectory as it is estimated by our approach also matches well to the ground truth as shown in Figure 5.9a.



(a) (b)

Figure 5.10: (a) In indoor environments a robot may encounter different floor types. (b) The standard deviation of the error of the odometry edges for the sliding window at each time step.



(a) x and y position of the laser (b) wheel radius

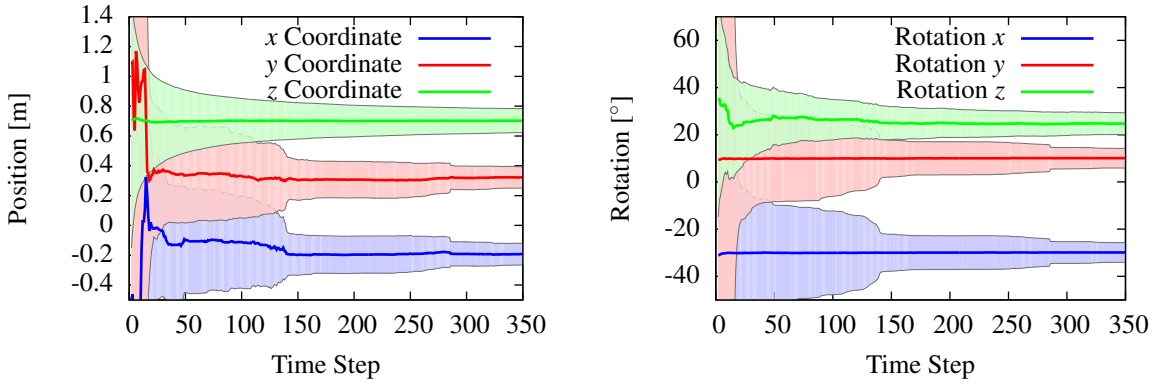
Figure 5.11: (a) The evolution of the x and y coordinate of the laser transformation as it is estimated by our approach. The true value of the x and y coordinate is 0.3 m and 0.6 m, respectively. (b) The estimate for the wheel radius having a true value of 0.12 m.

5.2.2 Influence of the Ground Surface

Within real-world indoor environments a robot may encounter different floor types, e.g., tiling, PVC flooring, wooden floor, or different kind of carpet. To test the influence of the floor type, we recorded data sets in which the robot drives on a soft carpet and on concrete tiling floor, as shown in Figure 5.10a. In this experiment we estimated the odometry parameters online. On both floors the estimated wheel radii were the same. By analyzing the standard deviation (see Figure 5.10b) in the error of the odometry edges $\mathbf{e}_i^{\mathbf{u}}$ for the sliding window around the current node we observe a higher noise in the odometry due to slippage on the carpet. This information can be stored in the map so that the robot can use it to adjust the motion model noise in a localization task.

5.2.3 Simulation Experiments

In a first experiment we simulated the 2D laser having a transformation of $(0.3, 0.6, 30°)^\top$ with respect to the odometry frame of the robot. Here, we optimized after inserting every node and monitored the evolution of the laser transformation as it is estimated by our approach in each time step. Figure 5.11 visualizes how the estimate for the x and y coordinate of the relative laser transformation and the radius of the left wheel along with their estimated uncertainty evolves.



(a) estimate of the position

(b) estimate of the attitude

Figure 5.12: The evolution of (a) estimate of the position and (b) the estimate of the attitude for a simulated depth camera as it is estimated by our approach. The ground truth values are $(x, y, z) = (-0.2, 0.3, 0.7)$ and $(-30°, 10°, 25°)$ for the rotation around the axes.

As we can see the estimate converges quickly to the correct transformation. By monitoring the marginal covariance of the estimated laser transformation we are able to judge the quality of the estimate.

Furthermore, we estimated the odometry parameters of the simulated robot whose left wheel has a radius of $r_l = 0.12\,\text{m}$, whereas the right wheel has a radius of $r_r = 0.125\,\text{m}$. The distance between the wheels is $b = 0.6\,\text{m}$. The output of the calibration is $r_l^* = 0.1207\,\text{m}$, $r_r^* = 0.1264\,\text{m}$, and $b^* = 0.607\,\text{m}$.

In a second set of experiments we simulated a robot equipped with a depth camera which estimates its own ego-motion in 3D. The translational offset was simulated as $(-0.2, 0.3, 0.7)^\top$, whereas the rotation around the axes was set to $(-30°, 10°, 25°)^\top$. The simulated wheel radii were the same as in the previous simulation experiment. Additionally, the ground plane observations yielding the rotation of the sensor around the x and y axes and its height above the ground were simulated and perturbed with Gaussian errors, which are sampled from $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 1^2)$ for the height estimate and the rotation angles, respectively. Figure 5.12 visualizes the outcome of our approach. As we can see, the calibration converges quickly to the true values. Furthermore, the estimated values of the kinematics parameters are $r_l^* = 0.1211\,\text{m}$, $r_r^* = 0.1263\,\text{m}$, and $b^* = 0.603\,\text{m}$.

We carried out further simulation experiments in which we randomly sampled the transformation of the on-board sensor with respect to the odometry frame and also modified the true wheel radii and the distance between the wheels. The calibration parameters as they are estimated by our approach did in all cases correspond well to the true values and the error was in the same range like in the particular examples reported above.

5.2.4 Real-World Experiments

To evaluate our approach on real-world data we processed data of a heterogeneous set of robots depicted in Figure 5.6. Table 5.1 summarizes the parameters of the platforms. To collect the data, we steered each robot twice through the environment. The front-end again processed the data to estimate the motion of the laser for each time step. When the robot revisits an already known region, the loop closure, which are detected by front-end, are added as constraints to the graph. Note that the estimation of the calibration parameters does not require to detect loop closures. Such a constraint, however, reduces the residual error in the trajectory as it is estimated

	PowerBot	EUROPA	Pioneer
wheel radius [m]	0.125	0.16	0.065
wheel distance [m]	0.56	0.7	0.35
ticks per revolution	22835	20000	1970
laser offset [m, m, °]	(0.22, 0, 0)	(0.3, 0, 0)	(0.1, 0, 0)
laser scanner model	Sick LMS291	Sick LMS151	Hokuyo URG

Table 5.1: The parameters of the robots used for our experiments.

	laser offset $\mathbf{d}^*$ (m, m, °)	wheel radii (r_l^*, r_r^*) (m, m)	distance b^* m
PowerBot - 1	(0.2258, 0.0026, 0.099)	(0.1263, 0.1275)	0.5825
PowerBot - 2	(0.2231, -0.0031, 0.077)	(0.1243, 0.1248)	0.6091
EUROPA - 1	(0.3067, -0.0051, -0.357)	(0.1603, 0.1605)	0.6969
EUROPA - 2	(0.3023, -0.0087, -0.013)	(0.1584, 0.1575)	0.7109
Pioneer - 1	(0.1045, 0.009, -0.178)	(0.0656, 0.065)	0.3519
Pioneer - 2	(0.1066, -0.0031, -0.28)	(0.0658, 0.0655)	0.3461

Table 5.2: Calibration results for different robot data sets.

by our approach. Table 5.2 summarizes the calibration results. As we can see, the result for the laser transformation are within a few millimeters of the manually measured position. The same holds for the radii of the wheels and their distance to each other.

Additionally, we mounted a Microsoft Kinect on the PowerBot platform and considered the stereo data captured by the EUROPA robot with its Bumblebee stereo camera to evaluate our approach for calibrating the position of those sensors. To this end, we again recorded two data sets with each platform. For the Kinect we manually measured a translation of $(0.28, 0.04, 1.0)$ and a rotation of $(0°, 21°, 15°)$, whereas the translation of the Bumblebee is $(0.21, 0.06, 1.19)$ and the rotation is $(0°, 20°, 0°)$ according to the CAD drawings of the robot. The parameters as they are estimated by our approach are summarized in Table 5.3. The obtained results indicate that our approach is able to accurately estimate the position of an on-board 3D range sensor and simultaneously calibrate the odometry parameters.

	sensor offset $\mathbf{d}^*$ (m, m, m, °, °, °)	wheel radii (r_l^*, r_r^*) (m, m)	distance b^* m
PowerBot - 1	(0.273, -0.049, 1.005, -0.842, 21.820, 15.673)	(0.1263, 0.1260)	0.5895
PowerBot - 2	(0.284, -0.035, 0.999, -0.645, 21.224, 15.749)	(0.1254, 0.1252)	0.5952
EUROPA - 1	(0.214, 0.061, 1.187, -0.912, 21.402, -0.842)	(0.1559, 0.1557)	0.7039
EUROPA - 2	(0.212, 0.057, 1.185, -0.523, 21.919, -0.612)	(0.1558, 0.1558)	0.7146

Table 5.3: Calibration results for different robot data sets with a 3D on-board sensor.

5.3 Related Work

The traditional approaches to calibrate a mobile robot and its sensors involve to accurately measure the trajectory of the robot while recording odometry and sensor measurements, for example, by external cameras or lasers [34]. A pioneering work in this area is the work by Borenstein and Feng [21]. Here, an external camera measures the error in the final location of the robot that is repeatedly driven back and forth along a square. This procedure recovers two out of the three parameters. Methods to calibrate the sensor position, which rely on an external data, for example, match the measurements against a known map to recover the trajectory of the sensor and use a least squares estimator to determine the relative transformation between the robot and its sensor. In a similar way the odometry parameters can be estimated via another independent least squares estimator given the knowledge of the reference trajectory [190]. For example, Antonelli *et al.* [9, 10] consider an external camera to track the position of a robot for calibrating the odometry parameters of a differential drive. Their method employs a least squares estimator which exploits the linear relation between the measurements and the unknown parameters.

In the context of computer vision, the idea to calibrate the intrinsic camera parameters while performing structure from motion is commonly known [86]. This problem has a structure which is very similar to the one addressed by our method. The main difference lies in the kind of parameters that are estimated. One of the first approaches to determine the stationary parameters of a mobile robot and to determine the error of the motion was proposed by Martinelli and Siegwart [150]. The idea behind this work is to extend the state of a Kalman filter used for localization with the kinematic parameters of the odometry. Whereas this approach can operate on-line, it requires an a priori known map. Subsequently, Jones *et al.* [95] and Kelly and Sukhatme [104] extended the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) algorithm to include calibration parameters. Despite their increased complexity, in these nonlinear problems smoothing approaches outperform filtering methods in terms of accuracy [200].

Eliazar and Parr [52] proposed to use an Expectation-Maximization approach to learn the motion model for a mobile robot. Their method is able to accurately estimate the parameters of a simplified odometry model and it does not require to know the map in advance. Their approach, however, requires to be run off-line due to its high computational requirements introduced by considering a particle filter for mapping. Based on a localization algorithm Roy and Thrun [180] estimate online the systematic error in the odometry. They treat the error in translation and rotation independently. Both approaches model the calibration as a linear function of the odometry measurement, whereas our approach estimates the physical parameters of the robot.

Gao and Spletzer [69] presented an approach to determine the extrinsic calibration parameters between two laser range finders. Underwood *et al.* [213] proposed a method to determine the 3D position of a laser within the body frame of the robot. Compared to our method those approaches either rely on establishing feature correspondences between the individual observations by preparing the environment with laser-reflective tape or assume a simple and partially known geometric environment for calibrating the sensors. Assuming a known configuration of landmarks in the environment, Antonelli *et al.* [8] described an approach to calibrate the odometry together with the extrinsic and intrinsic parameters of a camera mounted on the robot. By just estimating the ego-motion of a pair of sensors in an arbitrary environment, Brookshire and Teller [23, 24] are able to calibrate the offset between the sensors either in 2D [23] or 3D [24]. Maddern *et al.* [145] perform an extrinsic calibration of LIDAR sensors by optimizing the point

cloud quality as the vehicle traverses an unmodified and unknown environment. Their metric for measuring the quality of the calibration is derived from the Rényi Quadratic Entropy.

Censi *et al.* [32, 33] proposed a technique similar to our method. They construct a least squares calibration problem that estimates both the kinematic parameters and the sensor position. Their approach does not need to know the map in advance, but it is restricted to the estimation of stationary parameters. Furthermore, since it relies on scan-matching to estimate the ego-motion of the sensor, this method does not provide an accurate map in large and loopy environments. Such a pure calibration method or one of the techniques described above, however, are the best choice if no simultaneous mapping of the environment is required. Here, calibration methods may provide guarantees for the convergence or the optimality of the parameters. Thus, these techniques do not require suitable initial values for the parameters.

Our work can be seen as an extension of traditional graph-based SLAM algorithms. We refer to Section 2.6 for a detailed discussion of the related work in that area.

5.4 Conclusions

In this chapter, we presented an approach to estimate the calibration parameters while performing SLAM. To this end, our approach extends the graph-based formulation of the SLAM problem to handle the calibration parameters. The overall approach is accurate and designed for online operation, which allows us to handle changes in the parameters. For example, placing a load onto the robot affects the wheel diameters while the robot is in operation. Furthermore, compared to ad-hoc calibration methods our approach solely relies on the on-board sensors of the robot and does not require external information.

Additionally, our approach has the potential to provide useful information about the ground surface which affects the uncertainty of the odometry measurements. This information may in the future be exploited for terrain classification and might also be considered by localization algorithms.

While an accurate calibration of the parameters allows us to tune the performance of matching algorithms, as we have shown in the motivating examples, it most importantly avoids to include systematic noise and thus improves the quality of the estimate. In the following two chapters, we provide additional methods for increasing the accuracy of the model.

Chapter 6

Using Aerial Images as Prior Information for Graph-Based SLAM

After we have discussed the estimation of underlying parameters while learning a model of the environment, we will now focus on improving the quality of the model estimated by our approach. Most existing solutions to the SLAM problem learn the map from scratch and have no means for incorporating prior information. On the other hand by exploiting priors, we obtain more information leading to a better result. In this chapter, we present a novel SLAM approach that achieves global consistency by utilizing publicly accessible aerial photographs as prior information. It inserts correspondences found between the sensor measurements and the aerial image as constraints into our graph-based optimization framework. We evaluate our algorithm based on large real-world data sets acquired in mixed indoor and outdoor environments by comparing the global accuracy with state-of-the-art SLAM approaches and GPS. The experimental results demonstrate that the maps acquired with our method show increased global consistency.

• • • • • • • • • • • •

Up to this point, we have considered methods for solving the optimization problem that arises in SLAM and we also introduced a method to estimate the underlying calibration parameters while performing SLAM. The simultaneous calibration presented in the previous chapter improves the fusion of measurements of the odometers of the vehicle and the on-board range sensor. Thus, the robot is able to reduce the systematic noise in the estimation process also when the parameters are subject to change. Furthermore, as we have shown in Chapter 4, our approaches yield accurate models of the environment, whereas the quality of the maps estimated with our approach actually exceeds the quality of the maps learned with other state-of-the-art methods. Albeit these good results, in this chapter and in the next one we will focus on techniques to increase the accuracy of the map. First, we present an approach that allows the robot to incorporate prior information into the estimation process for improving the overall quality of the estimate, whereas we in the subsequent chapter concentrate on the fine details of the map.

Originally, the SLAM problem has been formulated independently of any specific prior about the environment and most SLAM approaches seek to determine the most likely map and trajectory taken by the robot given a sequence of observations without taking into account special priors. Priors can, however, greatly improve solutions to the SLAM problem. Consider, for

(a) standard SLAM (b) localization (c) our approach

Figure 6.1: Motivating example comparing (a) standard SLAM, (b) localization using aerial imagery as prior information visualized in, and (c) our combined approach. Note the misalignment relative to the outer wall of the building in (a). Whereas the localization applied in (b), which relies on aerial images, yields proper alignments, it cannot provide accurate estimates inside the building. (c) Combining the information of both algorithms achieves the best result (aerial image © Google).

example, a scenario in which a globally consistent map is required or in which the robot has to navigate to a target location specified in global terms such as a specific GPS coordinate. Corresponding applications include rescue or surveillance missions in which one requires specific areas to be covered. Unfortunately, GPS typically suffers from outages so that a robot only relying on GPS information might encounter substantial positioning errors. At the same time, even sophisticated SLAM algorithms cannot fully compensate for these errors as there still might be a lack of constraints between certain observations combined with large odometry errors. Even in situations with substantial overlap between consecutive observations, the matching processes might result in errors that linearly propagate over time and lead to substantial absolute errors. Consider, for example, a mobile robot mapping a linear structure, such as a corridor of a building or the passage between two parallel buildings. Typically, this corridor will be slightly curved in the resulting map. Whereas this is not critical in many applications, as the computed maps are in general locally consistent [91], it might be sub-optimal in application scenarios in which global consistency is required, such as those discussed above.

In this chapter, we present an approach that overcomes these problems by utilizing aerial photographs for calculating global constraints within a graph-representation of the SLAM problem. In our approach, these constraints are obtained by relating the data from the sensor of the robot to the aerial image. In particular, we consider 3D point clouds obtained by a laser range finder and the images provided by a stereo camera.

Compared to traditional SLAM approaches, the use of a global prior enables our technique to provide more accurate solutions by limiting the error when visiting unknown regions. In contrast to approaches that seek to directly localize a robot in an outdoor environment, our approach is able to operate reliably even when the prior is not available, for example, because of the lack of appropriate matches. Therefore, it is suitable for mixed indoor/outdoor operation. Figure 6.1 shows a motivating example and compares the outcome of our approach with the ones obtained by applying a state-of-the-art SLAM algorithm and a pure localization method using aerial images. While the localization aligns well with the aerial image, it cannot estimate the pose of the robot driving indoors. Thus, integrating this information into a SLAM algorithm improves the quality of the map.

The approach proposed in this chapter applies the graph-based approach of the SLAM problem as discussed in Chapter 2. In this variant of the SLAM problem, every node of the graph

represents a robot pose. Edges in the graph encode relative transformations between nodes computed from overlapping observations. Additionally, our system computes its global position for every node for which the prior is available by employing a variant of Monte Carlo localization (MCL). In particular, our approach considers 3D laser scans or stereo images as observations and aerial images as reference maps. The use of 3D information allows our system to determine the portions of the image and of the 3D scene that can be reliably matched. It computes these matches by detecting structures that potentially correspond to intensity variations in the aerial image. If the sensors also provide us with images, we additionally take into account the visual information for matching. Note that our system preserves the flexibility of traditional SLAM approaches and can also be used in absence of any prior information. When the prior is available our system provides highly accurate solutions also in pathological data sets (i.e., when no loop closures take place), though.

GPS is a popular device for obtaining position estimates. Whereas it has also been used to localize mobile vehicles operating outdoors, we found that the accuracy of this estimate is in general not sufficient to obtain precise maps by only considering GPS data, especially when the robot moves close to buildings or in narrow streets. This is also acknowledged by other researchers. For example, Levinson and Thrun [134] report localization errors with a high-end GPS system such that a map-based localization algorithm is required for robust navigation.

The remainder of this chapter is organized as follows. In the next section, we describe how to obtain the prior information from aerial images by means of a localization algorithm, which applies two different sensor modalities. The first one applies a 3D laser range finder and the second one considers the visual information by a stereo camera. Subsequently, we present the corresponding optimization problem in Section 6.2, which integrates the prior information derived from the aerial image. After presenting the experiments in Section 6.3, we discuss the related work in Section 6.4.

6.1 Prior Information from Aerial Images

Our system relies on a graph-based optimization back-end for solving the optimization problem that originates from SLAM. It operates on a sequence of sensor measurements and odometry. Every node of the graph represents a position of the robot at which a sensor measurement was acquired. In addition to direct constraints between poses obtained by matching the respective sensor data, it can integrate prior information whenever such information is available. Here, we consider the prior given in form of an aerial image. Such images are publicly available from various sources.

This prior information is introduced to the graph-based SLAM framework as global factors on the nodes of the graph. These global constraints are absolute locations of the robot obtained by MCL on a map inferred from the aerial images. As these images are captured from a viewpoint which is significantly different from the one of the robot, we extract corresponding 2D features from the 3D measurements obtained from a laser scanner or a stereo camera which are more likely to be consistent with the ones visible in the image. In this way, we can prevent the system from introducing inconsistent prior information.

In the following we explain how we adapted MCL to operate on aerial images and how to select the points in the 3D measurements to be considered in the observation model. After describing how to utilize the data of a 3D range finder, we present a method which employs a stereo camera to extract the features for localizing the vehicle. In the following section, we describe our extension to the optimization back-end for solving the extended SLAM framework including the prior information.

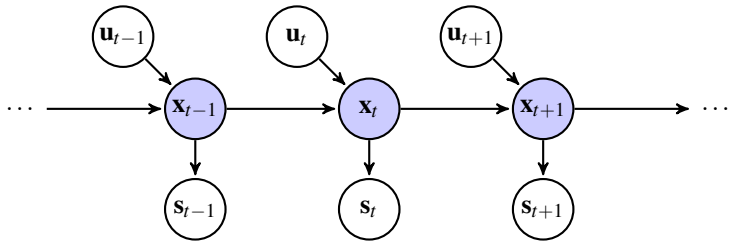


Figure 6.2: Monte Carlo localization represented as a Bayesian network, which visualizes the Markov assumption.

6.1.1 Monte Carlo Localization

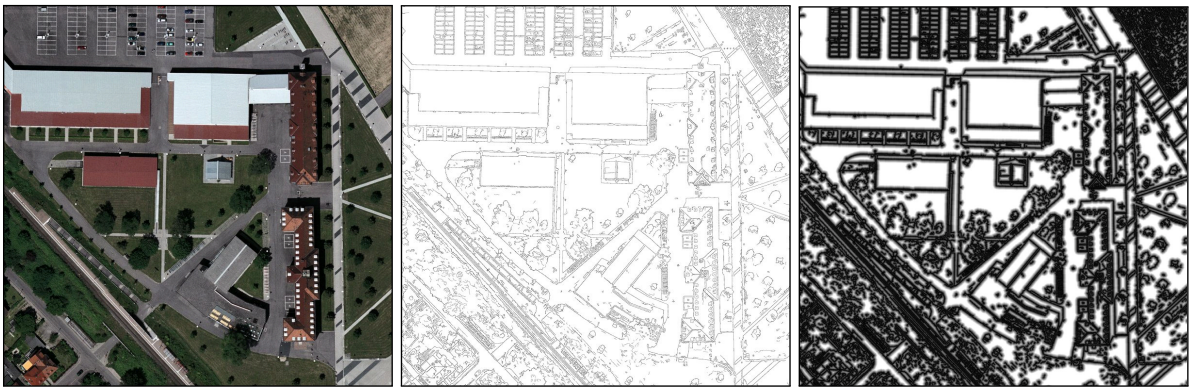
To estimate the current pose $\mathbf{x}_t$ of the robot in an environment, we consider probabilistic localization, which follows the recursive Bayesian filtering scheme [205]. The key idea of this approach is to maintain a probability density $Bel(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t})$ of the location $\mathbf{x}_t$ of the robot at time t given all observations $\mathbf{s}_{1:t}$ and all control inputs $\mathbf{u}_{1:t}$. Again, we assume that the state $\mathbf{x}_t$ encodes all relevant information up to time t (Markov assumption). This allows us to obtain this posterior by a recursive formula as follows:

$$
\begin{aligned}
Bel(\mathbf{x}_t) &= p(\mathbf{x}_t \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t}) && (6.1)\\
&\overset{\text{Bayes}}{=} \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t, \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t})\; p(\mathbf{x}_t \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) && (6.2)\\
&\overset{\text{Markov}}{=} \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t)\; p(\mathbf{x}_t \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}) && (6.3)\\
&\overset{\text{total prob.}}{=} \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t}, \mathbf{x}_{t-1})\; p(\mathbf{x}_{t-1} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t})\, d\mathbf{x}_{t-1} && (6.4)\\
&\overset{\text{Markov}}{=} \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})\; p(\mathbf{x}_{t-1} \mid \mathbf{s}_{1:t-1}, \mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} && (6.5)\\
&= \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})\; Bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}. && (6.6)
\end{aligned}
$$

Here, η is a normalization factor which results from applying Bayes' rule but omitting the denominator. The terms in Eq. 6.6 are the prediction model $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ and the sensor model $p(\mathbf{s}_t \mid \mathbf{x}_t)$, see also the Bayesian network in Figure 6.2.

For the implementation of the described filtering scheme, we employ a sample-based approach which is commonly known as Monte Carlo localization (MCL) [42]. Each sample corresponds to a possible pose of the robot and has an assigned weight and we consider 1,000 samples in our current implementation. For realizing the prediction step, we implemented a standard model operating on the odometry $\mathbf{u}$ of the robot [52, 42]. The model samples a motion command for each particle individually and assumes Gaussian noise for modeling the uncertainty in the odometry.

The remaining part to be described for implementing MCL on aerial images is the sensor model $p(\mathbf{s} \mid \mathbf{x})$. To this end, we need to determine the likelihood for obtaining the measurement $\mathbf{s}$ given the robot is at position $\mathbf{x}$ in an aerial image. In our current system, we apply the so-called endpoint model or likelihood fields [205]. Let $\mathbf{p}^k$ be the 2D point corresponding to the k-th beam $\mathbf{s}^k$ of the range measurement $\mathbf{s}$ projected into global coordinates assuming the robot is located at $\mathbf{x}$. The endpoint model computes the likelihood of $\mathbf{s}^k$ based on the distance between the scan point $\mathbf{p}^k$ and the point $\mathbf{o}^k$ in the map which is closest to $\mathbf{p}^k$. Furthermore, we assume



(a) aerial image (b) Canny image (c) likelihood field

Figure 6.3: (a) Aerial image (© Google) of the campus of the University of Freiburg, (b) the corresponding Canny image, and (c) the corresponding likelihood field computed from the Canny image. Note that the structure of the buildings and the vertical elements is clearly visible despite of the considerable amount of clutter.

that the beams are independent and the sensor noise is normally distributed. Thus, we obtain

$$p(\mathbf{s} \mid \mathbf{x}) \propto \prod_{k=1}^{|\mathscr{S}|} \exp\left(-\frac{\|\mathbf{p}^k - \mathbf{o}^k\|^2}{2\sigma^2}\right), \tag{6.7}$$

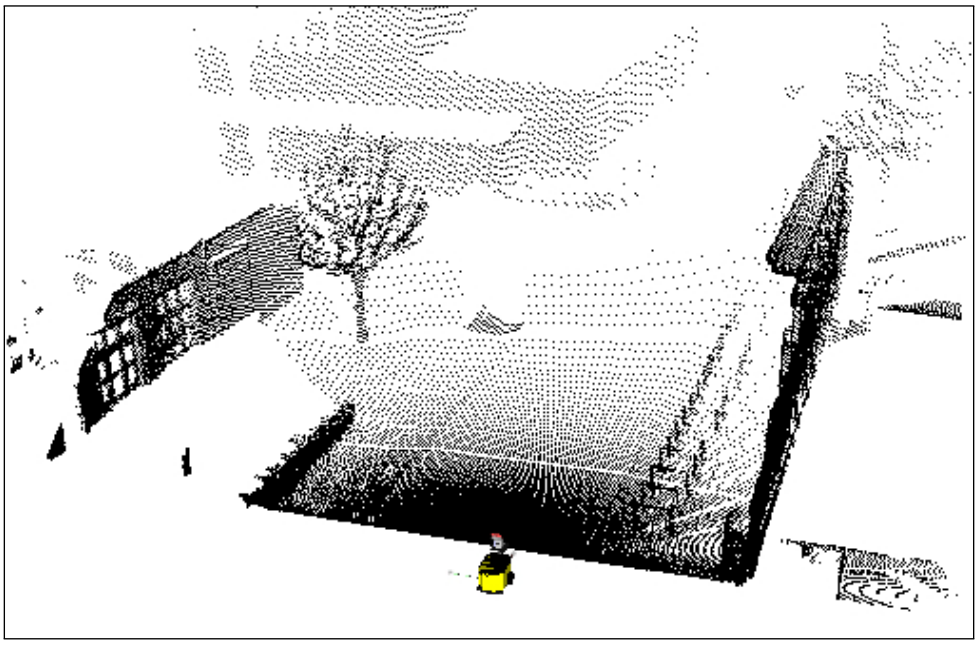
where $\sigma = 2.0\,\text{m}$ in our system. Since the aerial image only contains 2D information about the scene, we need to select a set of 2D points from a 3D observation, which is either given by the 3D range finder or the stereo camera. The extracted 2D points should correspond to structures that can be identified and matched with the aerial image. In other words, we need to transform both the scan and the image to a set of 2D points which can be applied in Eq. 6.7.

To extract candidate points from the aerial image, we employ the standard Canny edge extraction procedure [28]. The idea behind this is that a height gap in the world is likely to coincide with a change in intensity, which is detectable by the edge extraction on the aerial image. In an urban environment such edges are typically generated by borders of roofs, trees, or fences. Of course, the edge extraction procedure returns a lot of false positives that do not represent any actual 3D structure. Such false positives include street markings, grass borders, shadows, and other flat intensity variations. All these aspects have to be considered by the sensor model. Figure 6.3 shows an aerial image and the extracted Canny image along with the likelihood-field.

After we have presented how to evaluate the likelihood of a position given a set of 2D points, the remaining point is how to convert the 3D range scan or a stereo image pair into a set of 2D locations that are likely to be visible in the canny image. Below we will describe two variants for implementing such a pre-processing step operating on different sensor modalities. Whereas the first one, described below, operates on 3D range data obtained with a sweeping laser scanner, the second one, presented in Section 6.1.3, is designed for a a stereo camera.

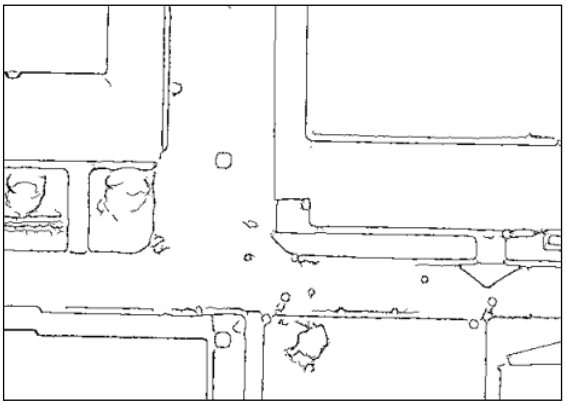
6.1.2 Extracting Height Variations in 3D Range Scans

The objective of our pre-processing of the 3D scan is to obtain a subset of 2D features that should contain all the points which may be visible in the reference map. To this end, we compute the z-buffer [62] of a scan from a bird's eye perspective. This allows us to discard points which are occluded in the bird's eye view from the 3D scan. Such occlusions might be caused, for

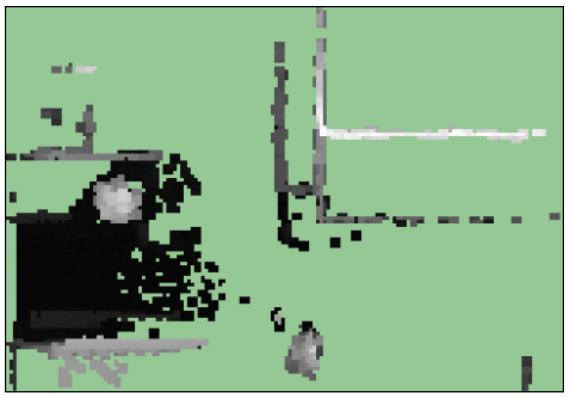


(a) 3D range scan

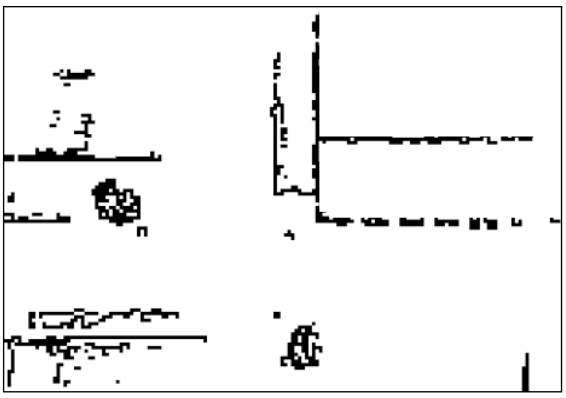
(b) aerial image



(c) Canny image



(d) z-buffer



(e) height variations

Figure 6.4: (a) A 3D scan represented as a point cloud; (b) the aerial image of the corresponding area (© Google); (c) the Canny edges extracted from the aerial image; (d) the 3D scene from (a) seen from the top (gray values represent the maximal height per cell, the darker a pixel the lower the height; and the green area was not visible in the 3D scan); (e) the positions extracted from (d), where the variation in height is above a threshold.

example, by overhanging roofs, where the house wall is occluded and therefore is not visible in the aerial image. Given the z-buffer, we construct a set of 2D points to be plugged into Eq. 6.7 by considering the 2D projection of the points exhibiting a variation in height with respect to their local neighborhood. These points are likely to coincide with an intensity change in the aerial image and thus are presumably visible in the Canny image. In our experiments, we considered variations in height of 0.5 m and above as possible positions of edges that could also be visible in the aerial image. Figure 6.4 shows an example for this procedure.

An implementation purely based on a 2D scanner (like the approach proposed by Frueh *et al.* [68]) would not account for occlusions due to overhanging objects, such as a roof. An additional situation, in which our approach is more robust, is in the presence of trees. In this case a 2D view would only sense the trunk, whereas the whole treetop is visible in the aerial image.

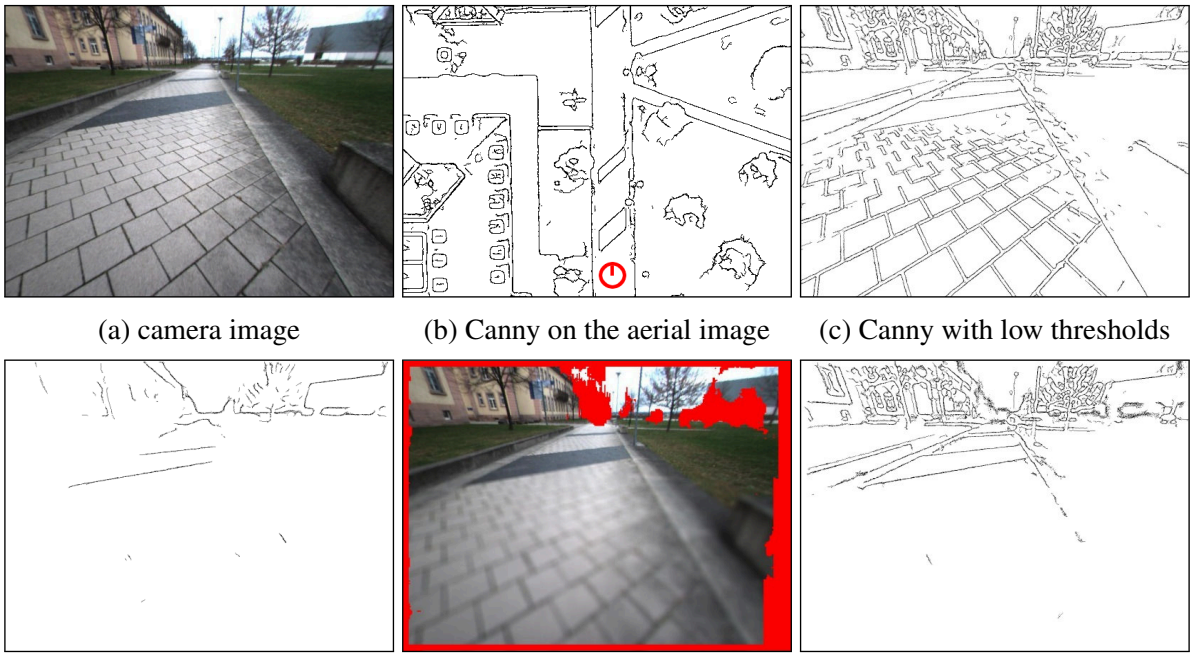
6.1.3 Features for Stereo Images

After having described a method for extracting height variations from a 3D laser scan, we will now focus on utilizing a stereo camera to extract the relevant information for localizing the robot. We can obtain 3D data from the stereo data by computing a disparity image. In addition to utilizing this 3D data with the procedure described above, we consider the color information to enable the robot to take advantage of flat structures. The color image allows us to extract street markings or borders of different ground surfaces that cannot be detected by a range-only device at all or without further post-processing, e.g., curb detection, detection of lanes, and road boundaries in general.

To extract the visual information, we proceed as follows. First, we process the stereo image to obtain 3D information by means of a disparity image. Second, we apply a variant of the Canny edge detector to the camera image. This is motivated by the fact that the same edges that are visible in the aerial image might also be visible in the camera image obtained by the robot. We, however, have to take into account that the image acquired by the robot and the aerial image differ substantially in resolution. Consider, for example, the paving tiles illustrated in Figure 6.5a, which are clearly identifiable in the camera image. Their pattern, however, is not present in the Canny image obtained from the aerial image, as we can see in Figure 6.5b. A straightforward application of an edge extraction either extracts the fine structures that are not visible in the aerial image (see Figure 6.5c) or misses important features if run with an increased acceptance threshold (see Figure 6.5d). Instead of that, we exploit the range data provided by the stereo processing for blurring the camera image. The kernel size of the blur is thereby inversely proportional to the range, which results in the image shown in Figure 6.5e. Figure 6.5f illustrates the result of the edge extraction utilizing the distance-based blur. Here, most of the structures that are also visible in the aerial image are extracted, but the ones which are too fine to be visible in the aerial image are neglected. Finally, since the aerial image is an orthogonal view, we discard features that are not obtained from the ground plane. Accordingly, we project the 3D points onto the ground plane to obtain a set of 2D points, which is finally applied in Eq. 6.7.

6.1.4 Discussion on the Sensor Model

Both sensor modalities have some limitations. The evaluation of $p(\mathbf{s} \mid \mathbf{x})$ is susceptible to visually cluttered areas in the aerial image since it can find random correspondences between the sensor data and the Canny edges in these areas of the aerial image. There is also the possibility



(a) camera image (b) Canny on the aerial image (c) Canny with low thresholds

(d) Canny with high thresholds (e) distance-based blur (f) Canny applied on (e)

Figure 6.5: Example for the distance-based blurring of the images as a preprocessing for the Canny edge detection. (a) The original camera image. (b) The corresponding part of the Canny edges with the position of the robot marked in red. (c) Standard Canny on the original camera image using low thresholds extracts a lot of edges not visible in the aerial image. (d) Standard Canny on the original camera image using thresholds that are high enough so that the pattern of the ground directly in front of the robot is not recognized as edges. (e) The blurred image, where the amount of blur is inversely proportional to the depth information. The area marked in red indicates that no range information is available. (f) Standard Canny on the dynamically blurred image with the same thresholds as in (c). Here, most of the important edges, i.e., those on the ground that are also visible in the aerial image were extracted correctly. Yet, the ground pattern in front of the robot was not extracted.

of systematic errors, when a wrong line is used for the localization, e.g., in the case of shadows. In our practical experiments we could not find evidence that this leads to substantial errors when one applies position tracking and as long as the robot does not move through such areas for a longer period of time. The main advantages of the endpoint model in this context are that it ignores possible correspondences outside of a certain range and that it implicitly deals with edge points that do not correspond to any 3D structure.

The model based on the 3D range finder data and the stereo data cover different aspects. While the stereo camera extracts visual features, like road markings, which are also present in the aerial image and which cannot be perceived by range only sensors, the 3D range data provided by a laser scanner yields more accurate depth information. Furthermore, the maximum detectable range is much larger for the laser compared to the range of our stereo camera. This is due to the small baseline of 0.12 m. As we will see in the experiments, a combination of the two models allows us to take advantage of both models to robustly localize the robot in difficult situations.

Our method, of course, also depends on the quality of the aerial images. Perspective distortions in the images could easily introduce errors. For the data sets used to carry out our experiments, however, we could not find evidence that this is a major complicating factor.

Finally, both presented sensor models are not applicable all the time. To this end, we employ a heuristic to detect when the prior is not available, i.e., when the robot is inside of a building

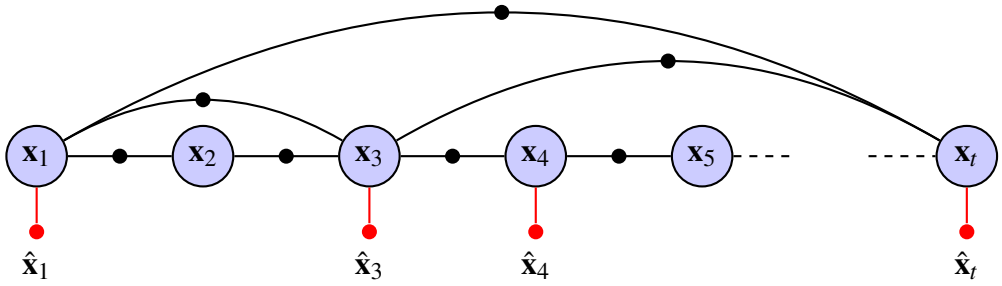


Figure 6.6: The graph representation of our method. In contrast to the standard approach, we additionally integrate global factors (shown in red) reflecting the prior information, which is obtained by localizing the robot in an aerial image. As indicated in the example, the prior information might not be available for all nodes, for example, if the robot is indoors.

or under structures that occlude the robot the bird's eye view. This heuristic is based on the 3D perception of the robot. If there are range measurements whose endpoints are directly above the robot, we do not integrate any global constraints from the position estimate since we assume that the area the robot is sensing is not visible in the aerial image. To this end, we build a local 3D map in which we accumulate the measurements of the robot. We rely on this local map to decide whether the robot is indoors. While a more profound solution regarding place recognition is clearly possible, this conservative heuristic turned out to yield sufficiently accurate results.

6.2 Priors in Graph-Based Maximum Likelihood SLAM

We apply a graph-based SLAM technique to estimate the most-likely trajectory, i.e., we seek for the maximum-likelihood (ML) configuration like the majority of approaches to graph-based SLAM. Let us briefly recall the main components of such a optimization framework. The details can be found in Chapter 2.

The poses of the robot are given by the vector $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_T)^\top$, where $\mathbf{x}_i$ describes the pose of node i. Furthermore, $\mathbf{s}_i$ is the range scan taken at the location of node i. Let $\mathbf{z}_{ij}$ and Ω_{ij} be respectively the mean and the information matrix of an observation of node j seen from node i, perturbed by Gaussian noise. The virtual measurement $\mathbf{z}_{ij}$ may arise from odometry or from matching sensor measurements and it yields a constraint in the optimization problem. For example, we obtain the virtual measurement $\mathbf{z}_{ij}$ by scan-matching the observations $\mathbf{s}_i$ and $\mathbf{s}_j$ in our front-end. Furthermore, let $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ be a function that computes a difference between the expected observation of the node $\mathbf{x}_j$ seen from the node $\mathbf{x}_i$ and the constraint $\mathbf{z}_{ij}$ gathered by the robot. Let $\mathscr{G}$ be the set of pairs of indices for which a constraint (virtual measurement) $\mathbf{z}$ exists. Assuming that the constraints are independent, we obtain the negative log likelihood $F(\mathbf{x})$ of all the observations

$$F(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathscr{G}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \Omega_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}). \tag{6.8}$$

To account for the residual error in each constraint, we can additionally consider the prior information $\mathscr{P}$ by incorporating the position estimates of our localization approach. To this end, we extend Eq. 6.8 as follows:

$$\tilde{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathscr{G}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \Omega_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) + \sum_{i \in \mathscr{P}} \mathbf{e}(\mathbf{x}_i, \hat{\mathbf{x}}_i)^\top \Omega_i \mathbf{e}(\mathbf{x}_i, \hat{\mathbf{x}}_i), \tag{6.9}$$

where $\hat{\mathbf{x}}_i$ denotes the position as it is estimated by the localization using the bird's eye image and Ω_i is the information matrix of this constraint. In our approach, we compute Ω_i based on the distribution of the samples in MCL. This leads us to the factor graph shown in Figure 6.6, where the prior information is indicated as red factors. The function $\mathbf{e}(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ is defined as

$$\mathbf{e}(\mathbf{x}_i, \hat{\mathbf{x}}_i) \stackrel{\text{def.}}{=} \mathbf{x}_i \ominus \hat{\mathbf{x}}_i. \tag{6.10}$$

Note that the prior might not be available in each time step, for example, if the robot is operating indoors, it has no access to the prior.

The result of the optimization is a set of poses that maximizes the likelihood of all the individual observations and accordingly minimizes the negative log-likelihood of all observations, i.e., we obtain $\mathbf{x}^*$ as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \tilde{F}(\mathbf{x}). \tag{6.11}$$

The optimization also accommodates the prior information about the environment to be mapped whenever such information is available. In particular, the objective function $\tilde{F}(\mathbf{x})$ encodes the available pose estimates as given by our MCL algorithm described in the previous section. Intuitively the optimization deforms the solution obtained by the relative constraints to maximize the overall likelihood of all the observations, including the priors. The optimization results in a consistent estimate, as long as the MCL gives the correct position of the vehicle. It is worth to mention that including the prior information about the environment yields a globally consistent estimate of the trajectory even in situations where no loop closures occur.

6.3 Experiments

The approach described above has been implemented and evaluated on real data acquired with a MobileRobots Powerbot equipped with a SICK LMS laser range finder mounted on an Amtec wrist unit. The 3D data used for the localization algorithm has been acquired by continuously tilting the laser up and down while the robot moves. The maximum translational velocity of the robot during data acquisition was 0.35 m/s. This relatively low speed allows our robot to obtain 3D data that is sufficiently dense to perform scan matching without the need to acquire the scans in a stop-and-go fashion. During each 3D scan the robot moved up to 2 m. We used the odometry to account for the distortion caused by the movement of the platform. Additionally, we utilize a Point Grey Bumblebee2 stereo camera to acquire the vision data. Although the robot is equipped with an array of sensors, in the experiments we only used the devices mentioned above.

6.3.1 Comparison to GPS

This first experiment aims to show the effectiveness of the localization on aerial images compared with the one achievable with a consumer grade GPS. We manually steered our robot along a 890 m long trajectory through our campus, entering and leaving buildings. The robot captured 445 3D scans that were utilized for localization. We also recorded the GPS data for comparison purposes. Figure 6.7 compares the GPS estimate with the one obtained by MCL on the aerial view. The higher error of the GPS-based approach is clearly visible. Note that GPS, in contrast to our approach, does not explicitly provide the orientation of the robot.

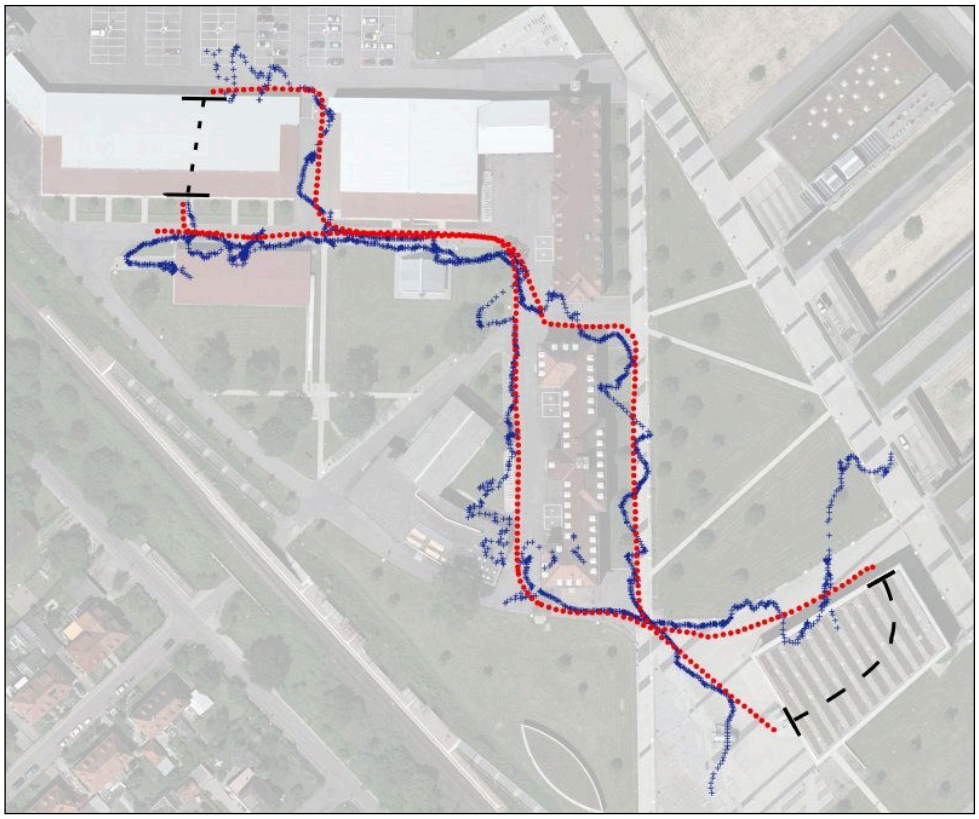


Figure 6.7: Comparison between GPS measurements (blue crosses) and global poses from the localization in the aerial image (red circles). Dashed lines indicate transitions through buildings, where GPS and aerial images are unavailable (aerial image © Google).

6.3.2 Comparison of a 3D Laser and a Stereo Camera for Localization

The proposed localization based on 3D laser data relies on the extraction of variations in height that are matched with the aerial image. In contrast, we can match the visual data provided by the stereo camera to visual features obtained from the ground plane with the aerial image. Some features, e.g., curbs, cannot be reliably detected by a 3D range sensor using height variations without introducing a lot of false positives that are not visible in the Canny image of the aerial photograph. Furthermore, other flat features such as road markings are not perceivable at all in range-only data. This experiment is designed to evaluate the performance of using just the data provided by a stereo camera for localizing the robot.

To compare the two proposed sensor modalities, we steered the robot along a 680 m long trajectory on our campus. While driving the robot again collected 3D scans like in the experiment described above. Additionally, the robot recorded stereo vision data. The stereo camera is mounted approximately 1.2 m above the ground and tilted downwards by 30 degrees. This setup allows the robot to observe the ground surface which is also visible in the aerial image. Using this data we analyzed the position estimate of MCL using the two different sensors described in Sections 6.1.2 and 6.1.3. Note that we set the update rate of the two approaches to the same frequency. Therefore, both approaches integrate the same number of sensor readings, i.e., we discard stereo images which are available at higher rates than the 3D laser scans generated by our platform. Figure 6.8 shows the trajectory estimate of the two approaches. As can be seen from the image, the estimate using vision is more accurate in this case. In the particular area, the robot is driving on the foot path going through a vegetated area, whereas the estimate using only 3D laser data is off the foot path due to the lack of a sufficiently dense 3D structure. In the

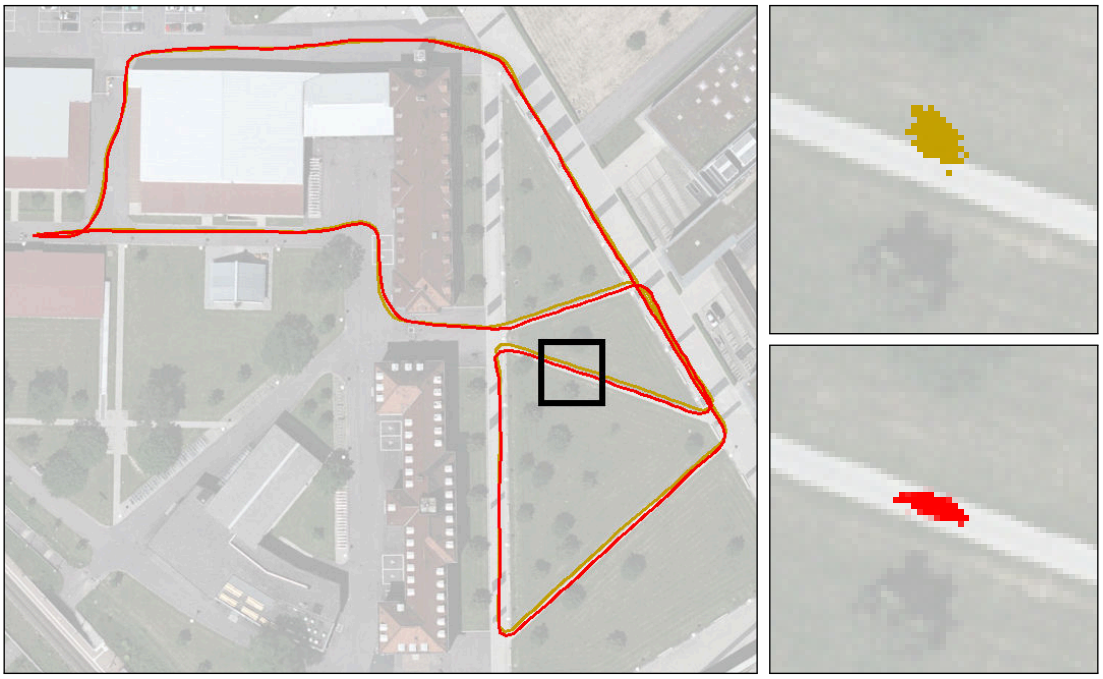


Figure 6.8: Comparison between MCL using 3D laser scans and stereo vision data. The trajectory as it is estimated based on the 3D laser and vision data is shown in yellow and red, respectively. The trajectory estimate using vision localizes the robot on the foot path, whereas the laser based localization is slightly off. The column on the right shows a magnified view of the black rectangle and it depicts the particle cloud for MCL using 3D laser scans (top) and stereo data (bottom) (aerial image © Google).

other parts of the trajectory, the estimate of the two approaches overlay with each other, i.e., we could not observe a substantial difference in the position estimate.

6.3.3 Global Map Consistency

The goal of this set of experiments is to evaluate the ability of our system to create a consistent map of a large mixed indoor and outdoor environment. We compare our approach against a graph-based SLAM approach, which does not have access to the prior data. The implementation is similar to the one proposed by Olson [163]. This approach applies scan-matching and identifies previously visited places by matching the respective observations while outliers are detected with a clustering technique. For evaluating the global map consistency, we recorded data in two different environments, our campus and a residential area. These two areas differ substantially. The campus area contains only a few large buildings, whereas the residential area consists of several rather small houses along with front gardens surrounded by fences and hedges. Additionally, cars, which were not present at the time the aerial image was taken, are parked on the narrow streets and vice versa. Figure 6.3a and Figure 6.12 show aerial images of the two test sites. First we describe the experiment carried out in the campus environment, followed by a description of the experiment in the residential area.

We evaluate the global consistency of the generated maps obtained with both approaches. To this end, we recorded five data sets by steering the robot through our campus area. In each run the robot followed approximately the same trajectory. Figure 6.9 depicts the trajectory of one of these data sets as it is estimated by our approach and a standard graph-based SLAM method. For each of the two approaches (our method using the aerial image and the graph-based SLAM technique that uses no prior information) we calculated the maximum likelihood

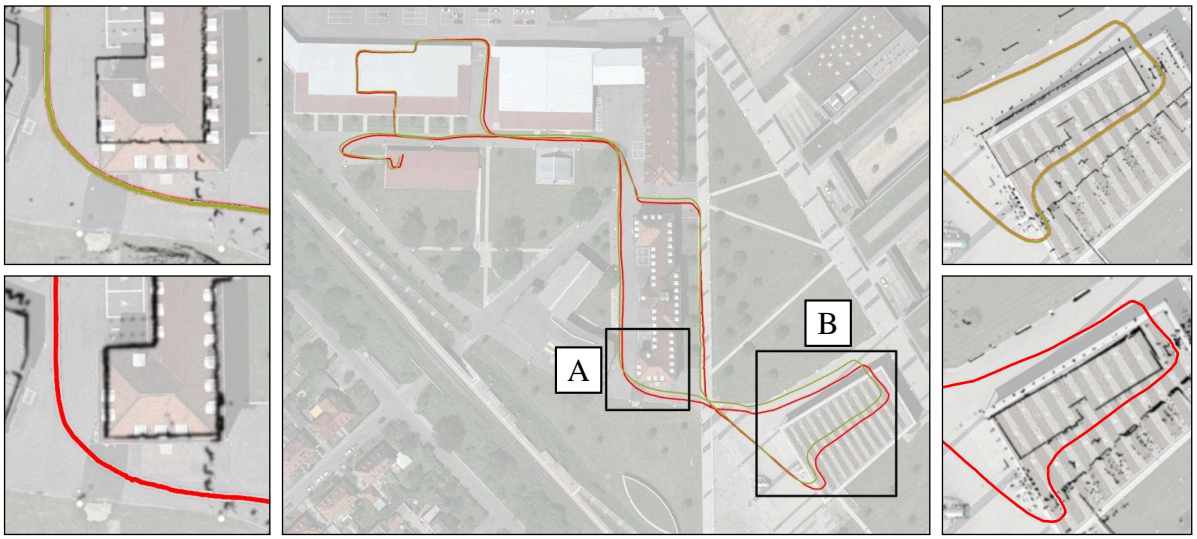


Figure 6.9: Comparison of our approach to a standard SLAM approach in a complex indoor/outdoor scenario. The center image shows the trajectory estimated by the SLAM approach (green) and the trajectory generated by our approach (red) overlaid on the Google Earth image used as prior information. On the left and right side, detailed views of the areas marked in the center image are shown, each including the trajectory and map. The upper images show the results of the standard SLAM approach; detail A on the left and B on the right. The lower images show the results of our system (A on the left side and B on the right). It is clearly visible that, in contrast to the SLAM algorithm without prior information, the map generated by our approach is accurately aligned with the aerial image (aerial image © Google).

map by processing the acquired data of each run.

For each of the five data sets we evaluated the global consistency of the maps by manually measuring the distances between six easily distinguishable points on the campus (see Figure 6.10). As ground-truth we considered the so-called *Automatisierte Liegenschaftskarte*, which we obtained from the German land registry office. It contains the outer walls of all buildings where the coordinates are stored in a global reference frame. This allowed us to compute the true distance between the chosen points. We compared these distances to the corresponding distances in the maps. We computed the average error in the distance between these points. The result of this comparison is summarized in Figure 6.11. As we can see, the errors of our approach are substantially smaller than the ones of the standard approach.

An additional experiment was carried out in a residential area. An aerial image of this area is depicted in Figure 6.12. We steered our robot five times on the streets along an approximately 710 m long trajectory. The data sets were recorded at different times and on several days, i.e., parts of the environment were subject to change. For example, the position of shadows changed and cars were parked in different locations. This environment is less structured than our campus environment. In particular, the parts of the environment which are marked in Figure 6.12 impose challenges for the MCL. The area marked on the right is dominated by vegetation along a railway embankment resulting in cluttered 3D range measurements. In this area, our approach using only 3D laser data is unable to accurately localize the robot and the MCL is likely to diverge. In the area marked on the left, the street is partially occluded due to overhanging trees. Here, the localization using stereo vision data is unable to robustly localize the vehicle. Using both sensors, the 3D laser data and the stereo images, our approach, however, is able to localize the robot also in these two challenging areas. The vision data provides useful information about the road borders that are not observed by the 3D laser close to the railway, whereas the 3D laser measures the trees and building structures in the other problematic region. Fusing the infor-

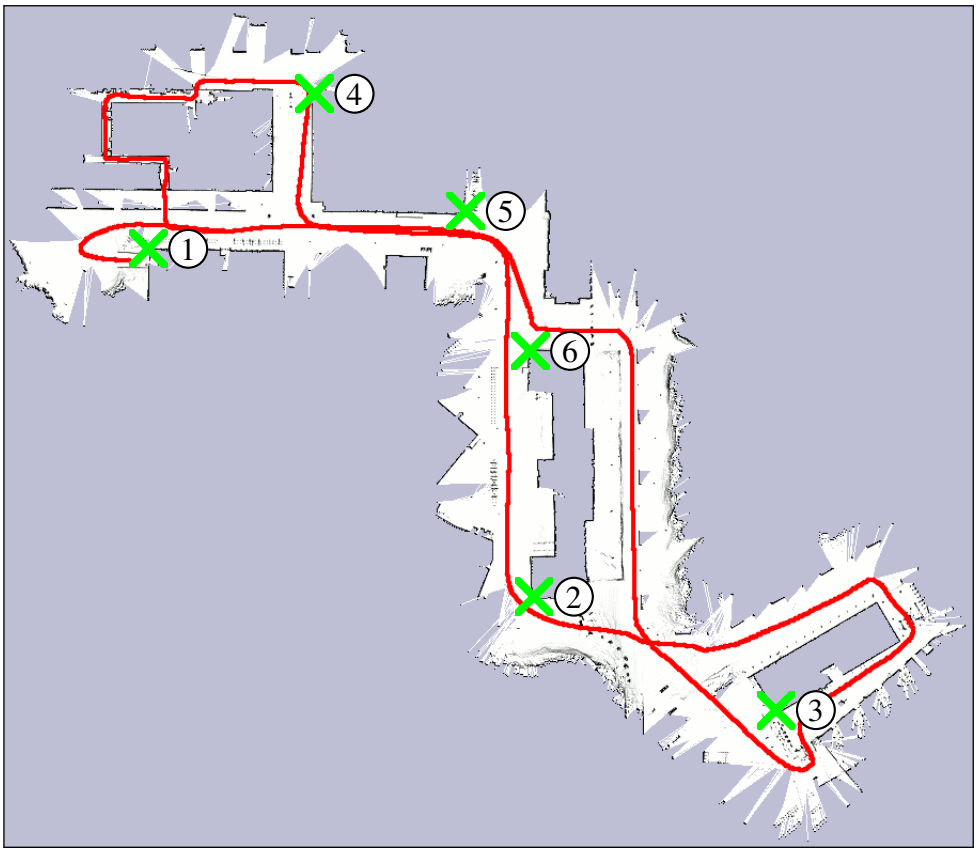


Figure 6.10: The six points (corners of the buildings) we used for evaluation are marked as crosses on the map.

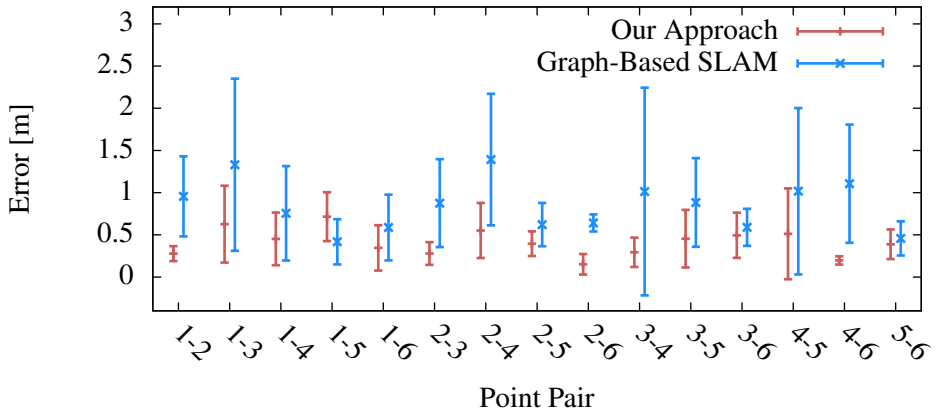


Figure 6.11: Error bars (95 % confidence interval) for the estimated distances between the six points used for evaluating the map consistency.

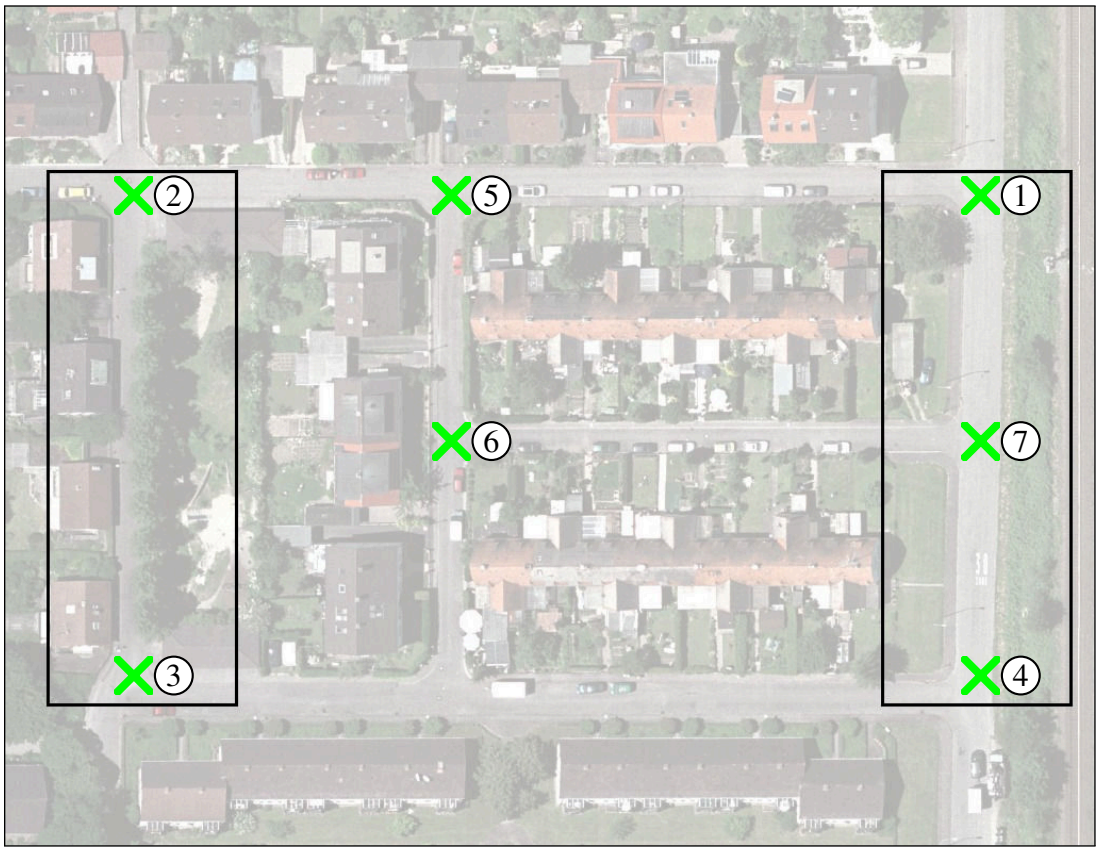


Figure 6.12: Aerial image of a residential area. The two areas marked with rectangles impose challenges to the localization algorithm. In the region marked on the left the localization using stereo vision fails. Within the area marked on the right the localization using 3D laser data is inaccurate. Using both sensors as input results in a accurate localization for the whole environment. The seven points we used for evaluation are marked as crosses on the aerial image (aerial image © Google).

mation of both sensors allows the robot to reliably track its position in the whole environment. For each run we computed the maximum likelihood estimates of the map for our approach and standard graph-based SLAM without prior information.

Unfortunately, an evaluation based on the ground truth map is not possible for this environment since most of the houses are not observable due to the fences and hedges along the street. We therefore have to rely on a highly accurate GPS receiver which achieves a sub-meter accuracy. Accumulating GPS data for a longer time period allows us to obtain even more accurate position estimates. To evaluate the output of our approach and the standard graph-based SLAM algorithm, we recorded the GPS information in each run. Additionally, we selected seven positions for which an accurate GPS estimate was available and we steered the robot over the same positions in each run. We measured the distance between the locations as determined by GPS and compared the distances with the maximum likelihood estimates of our approach and standard graph-based SLAM for each run. Figure 6.13 summarizes the results. While our approach is able to achieve an average error of 0.85 m the graph-based SLAM algorithm without prior information achieved an average error of 1.3 m.

As these two experiments reveal, SLAM without prior information results in a larger error than obtained with our approach in both environments. Additionally, the standard deviation of the estimated distances is substantially smaller than the standard deviation obtained with a graph-based SLAM approach that does not utilize prior information. Our approach is able to estimate a globally consistent map on each data set. Note that similar accuracies with respect

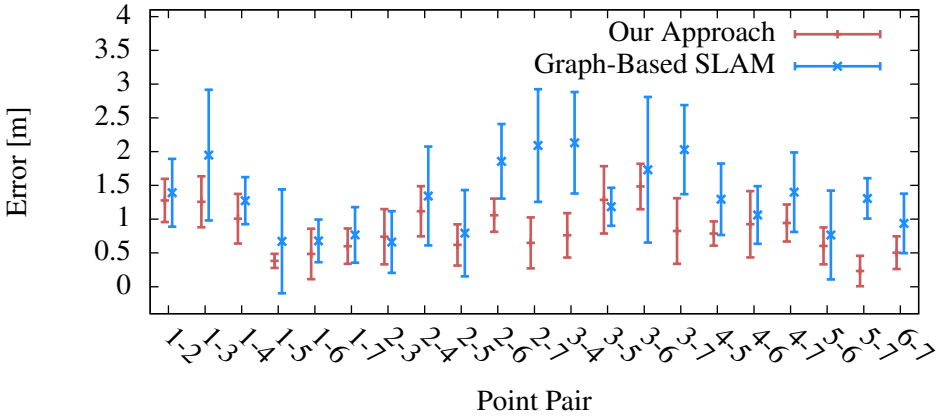
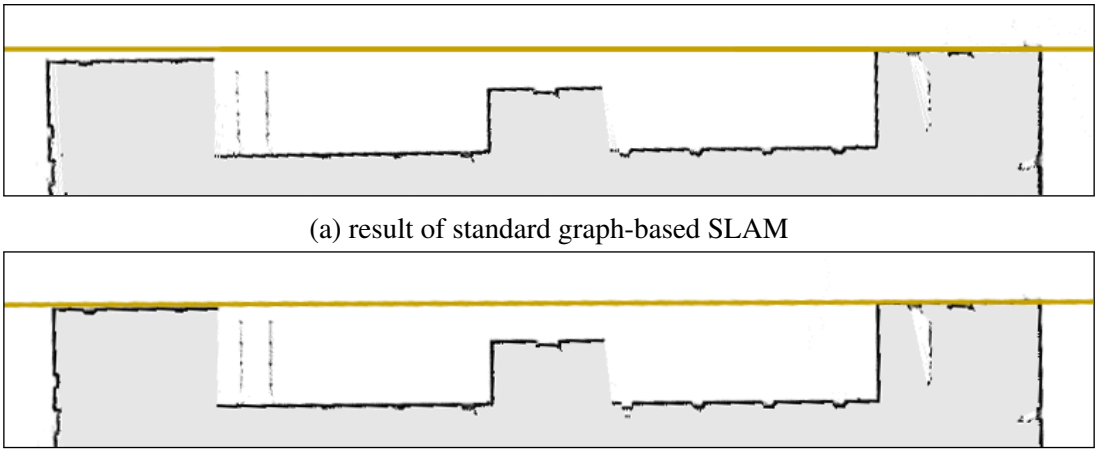


Figure 6.13: Error bars (95 % confidence interval) for the estimated distances between the seven points used for evaluating the map consistency.



(a) result of standard graph-based SLAM

(b) result of our approach

Figure 6.14: Close-up view of an outer wall of a building as it is estimated by graph-based SLAM (top) and our method with prior information (bottom). In both images a horizontal line visualizes the true orientation of the wall. As can be seen from the image, graph-based SLAM bends the straight walls of the building more than our approach.

to global consistency might be obtained with a standard SLAM procedure if the data contained more loop closures. This indicates an additional advantage of our method, namely that it in principle does not require loop closures to achieve global consistency, at least when the prior is available for most of the time.

6.3.4 Local Alignment Errors

Ideally, the result of a SLAM algorithm should perfectly correspond to the ground truth. For example, the straight wall of a building should lead to a straight structure in the resulting map. The residual errors in the scan matching process, however, typically lead to a slightly bended wall. We investigated this in our five data sets for both SLAM algorithms by analyzing an approximately 70 m long building on our campus. This building corresponds to the longest straight structure in this environment and was therefore chosen for evaluation. To measure the accuracy, we approximated the first part of the wall by a line and extended this line to the other end of the building. In a perfectly estimated map, both corners of the building are located on this line. Figure 6.14 depicts a typical result. On average the distance between the horizontal

line and the corner of the building for standard graph-based SLAM is 0.5 m, whereas it is 0.2 m for our approach in the five data sets.

6.4 Related Work

The SLAM methods discussed in Chapter 2 do not take into account any prior knowledge about the environment. Those approaches typically start to estimate a model of the environment from scratch. On the other hand, several authors addressed the problem of utilizing prior knowledge to localize a robot outdoors. For example, Korah *et al.* [113] use image processing techniques to extract roads on aerial images. This information is then applied to improve the quality of GPS paths using a particle filter by calculating the particle weight according to its position relative to the streets. Leung *et al.* [133] present a particle filter performing localization on aerial photographs by matching images taken from the ground with a monocular vision system. The approach detects line features to find correspondences between the aerial and ground images. It applies a Canny edge detector, progressive probabilistic Hough transform to find lines in aerial images, and furthermore performs a vanishing point analysis for estimating building wall orientations in the monocular vision data. The approach achieves an average positioning accuracy of several meters. Ding *et al.* [44] use a vanishing point analysis to extract 2D corners from aerial images and inertial tracking data. They also extract 2D corners from LIDAR generated depth maps and apply a multi-stage process to match these corners with those from the aerial image. The corresponding matches finally yield a fine estimation of the camera pose that is used to texture the LIDAR models with the aerial images. Chen *et al.* [35] use an energy minimization technique to merge prior information from aerial images and mapping. They perform mapping by constructing sub-maps consisting of 3D point clouds, that are constrained by relations. Using a Canny edge detector, they compute a vector field from the image that models force towards the detected edges. The sum of the forces applied to each point corresponds to the energy measure in the minimization process, when placing a sub-map into the vector field of the image.

While our approach and the related work mentioned above operate on aerial images, several researchers considered manually drawn maps for navigation. For example, Parekh *et al.* [168] establish correspondences with a sketched map, which requires to deal with different scale and shape given in the sketch. Recently, Matsuo and Miura [151] presented a variant of FastSLAM which utilizes a hand-drawn map as initial estimate.

Incorporating the available prior information was also considered by Parsley and Julier [172, 173]. They demonstrate how to incorporate a heterogeneous prior map into an extended Kalman filter [172] or graph-based SLAM [173], respectively. They show that such a prior bounds the error while the robot travels in open-loop. Lee *et al.* [131] use the road graph from a given prior map for SLAM. Under the assumption that the vehicle follows only roads they can constrain the probabilistic model to the roads and thus achieve higher accuracy than traditional FastSLAM. Dogruer *et al.* [45] utilized soft computing techniques for segmenting aerial images into different regions such as buildings, roads, and forests. They applied MCL on the segmented maps. Compared to the approach presented in this chapter, their technique strongly depends on the color distribution of the aerial images since different objects on these images might share similar color characteristics.

Frueh *et al.* [68] described the generation of edge images from aerial photographs for 2D laser-based localization. As they state in their paper, localization errors might occur if rooftops seen on the aerial image significantly differ from the building footprint observed by the 2D scanner. Our approach proposed in this chapter computes a 2D structure from a 3D observation, which is more likely to match with the features extracted from the aerial image. This leads to an

improved robustness in finding corresponding locations. Additionally, our system is not limited to operate in areas where the prior is available. When no prior is available, our algorithm operates without relevant performance loss compared to standard SLAM approaches which do not utilize any prior. Our system furthermore allows a robot to operate in mixed indoor/outdoor scenarios.

Sofman *et al.* [196] introduced an online learning system predicting terrain travel costs for unmanned ground vehicles (UGVs) on a large scale. They extract features from locally observed 3D point clouds and generalize them on overhead data, such as aerial photographs, allowing the UGVs to navigate on less obstructed paths. Montemerlo and Thrun [153] presented an approach similar to the one presented in this chapter. The major difference to our technique is that they use GPS to obtain the prior. Due to the increased noise which affects the GPS measurements this prior can lead to larger estimation errors in the resulting maps.

It is worth to mention that our approach to generate a map out of the aerial image for localization is a straightforward application of the Canny edge extraction. Despite that the localization with our edge image worked well, an improved map can be obtained, for example, by the method described by Pink and Stiller [176]. Their approach applies distinctive image features to automatically generate a map, which contains considerably less clutter than our method, given the aerial image.

Furthermore, our approach was subsequently extended by Kleiner and Dornhege [105] to assist first-aiders in urban search and rescue scenarios. They focus on assisting a human who remotely operates the robot and hence the user assists the robot in determining the location in the aerial image. The pose provided by the user is then refined by applying our localization algorithm.

6.5 Conclusions

In this chapter, we presented an approach to incorporate the prior knowledge about the structure of an environment into the state estimation of an graph-based SLAM system. To incorporate the prior given by the aerial images into the graph-based SLAM procedure, we utilize a variant of Monte Carlo localization with a novel sensor model for matching 3D laser scans to aerial images. Additionally, we suggested a sensor model for using a stereo camera to localize the robot given an aerial image. A combination of the two sensor modalities achieves the best performance by overcoming the drawbacks of the individual sensors. Given the prior our approach can achieve accurate global consistency without the need to close loops.

Our method has been implemented and tested in a complex mixed indoor and outdoor setting. Practical experiments carried out on data recorded with a real robot demonstrate that our algorithm outperforms state-of-the-art approaches for solving the SLAM problem that have no access to prior information in terms of accuracy of the resulting map. In situations, in which no global constraints are available, our approach is equivalent to standard graphical SLAM techniques. Thus, our method can be regarded as an extension to existing solutions of the SLAM problem.

Chapter 7

Highly Accurate Maximum Likelihood Laser Mapping

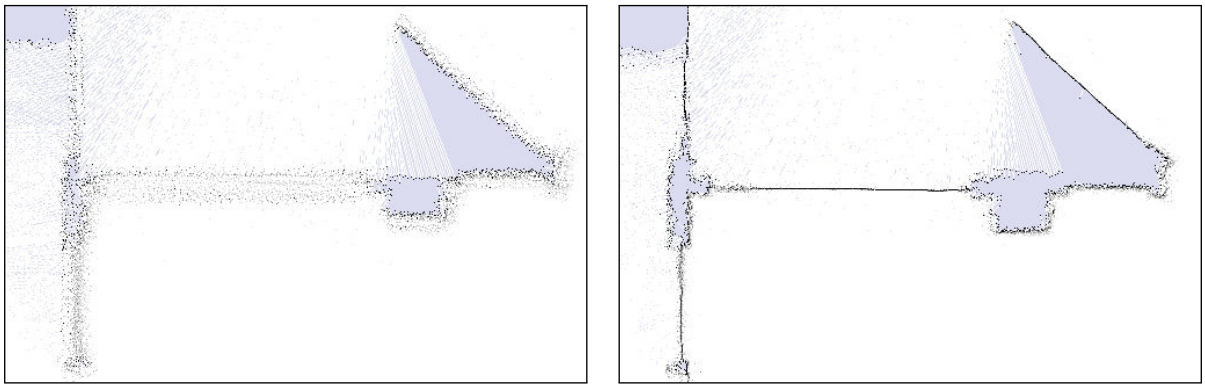
Whereas we in the last chapter illustrated one technique to improve the accuracy of the results of our SLAM algorithm, we now present an approach for obtaining highly accurate laser-based occupancy grid maps. Our approach is a post-processing step for SLAM to further improve the quality of the map that is obtained by integrating the laser readings into an occupancy grid. Thereto, we represent the laser readings as samples of the measured surface and we assume that the surface is locally smooth. Consequently, we approximate the surface by a Gaussian. Our approach in particular models the conic shape of the laser beam and considers the incidence angle of the beam. This allows us to construct a joint optimization problem that includes the poses of the robot and the laser readings. We evaluated our approach on a collection of real-world and simulated data sets. Furthermore, we demonstrate how the highly accurate maps greatly improve the localization performance.

• • • • • • • • • • •

In the previous chapter, we have discussed one way to improve the quality of the maps. We boosted the quality by including prior information about the environment to obtain a globally consistent map. The map was given by the configuration of the robot poses that best explains the constraints and the prior. While this technique yields accurate maps of large-scale environments, as we have shown, it treats the scan as rigid body and disregards the noise in the range reading itself. Furthermore, the standard method does not take into account the result of the optimization to further enhance the data association.

In this chapter, we will present an additional technique for improving the quality of the maps on a more fine-grained level. So far, we modeled the optimization problem as a graph, in which each pose of the robot is represented by a node and the edges between the nodes encode the constraints arising from the pair-wise matching of the respective observations along with a Gaussian covering the noise in the matching process. The map is rendered in a decoupled step by integrating the range scans at their maximum likelihood location [205].

In contrast to the standard technique, the approach presented in this chapter constructs an optimization problem that includes the position of the robot and the points of the range scans. In particular, we model the points obtained by the range finder as samples. To this end, we assume



(a) result of standard graph-based SLAM (b) result of our approach

Figure 7.1: Example for the accuracy of the maps generated by our approach. (a) A magnified view of a map obtained with a state-of-the-art traditional SLAM algorithm. (b) Same view of a map rendered from the result of the joint optimization of the robot poses and the individual scan points.

that the scanner samples points from a locally smooth surface. Since each sample is influenced by the local characteristics of the surface and the properties of the laser beam, we explicitly take into account the conic shape of the beam and the incidence angle of the beam to the surface. These two quantities allow us to better explain the noise in the range measurements. Out of the positions of the robot and the range measurement we construct a joint optimization problem which incorporates the above mentioned properties. By building upon the sparse optimization technique presented in Chapter 2, we are able to efficiently solve this problem, despite the large number of variables involved.

In Figure 7.1 we illustrate the advantages of our method compared to the standard technique. The figure shows a zoomed view of a map obtained by the standard SLAM technique and a map that has been optimized with our approach. Our result appears to be substantially more accurate. Within the experimental evaluation, we will present a quantitative evaluation for measuring the accuracy of the maps. Furthermore, we will illustrate the practical benefits of our techniques for localizing the robot in the map.

The remainder of this chapter is organized as follows. In Section 7.1 we present the underlying models for the laser and the map which leads us to a joint optimization problem. Afterwards, in Section 7.2 we discuss the differences of our method to the Iterative Closest Point algorithm (ICP) and Bundle Adjustment. In Section 7.3 we present an evaluation of our approach based on a set of real-world data sets. Finally, in Section 7.4 we discuss the related work.

7.1 Estimating Accurate Environment Models

As mentioned above our approach tries to maximize the accuracy of the models of the environment obtained by a SLAM algorithm. Our approach assumes that we can locally approximate the surface, which reflects the laser beam, by a line segment. This assumption typically holds in man-made indoor environments, as they are made out of smooth structures. In the remainder of this chapter we will refer to this locally smooth structures as *surface patches* or simply *patches*.

The key idea of our approach is to jointly optimize the positions of the surface patches and the positions of the robot. We achieve this by constructing a sparse optimization problem. Let us briefly sketch the individual steps of the algorithm, before we describe the specific parts in detail. Given the current range measurements, we estimate the local properties of each surface

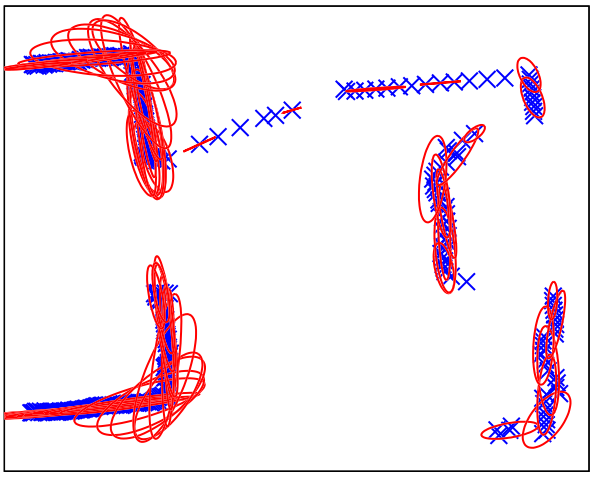


Figure 7.2: The Gaussian distributions for the points computed on a range scan. For better visibility only every fifth ellipse is drawn.

patch. The local properties of each patch allow us to find a nearby patch that is likely to be sampled from the same surface. The correspondences are the basis for setting up an optimization problem that seeks to minimize the distance between corresponding surface patches while taking into account the properties of the sensor which sampled the surface. After the optimization converged, we update the local properties of each patch and we also re-estimate the data association. This leads to a new optimization problem. We repeat the procedure until a convergence criterion is matched.

In the following we will describe in detail the underlying model of the surface patches and the model of the sensor, which is motivated by the physical mode of operation of the range finder. Afterwards, we present our strategy for establishing the optimization problem and the objective function of the problem itself. We will furthermore illustrate the differences to standard matching by Iterative Closest Point (ICP) [19, 188] and Bundle Adjustment (BA) [210] in the following section.

7.1.1 Model for the Surface Patches

As mentioned above, we assume that the surface is locally smooth. Hence, we can represent the patch by a Gaussian distribution. Let us denote the beam of an individual range scan as $\mathbf{s}_i^k$ which is measured by the robot at position $\mathbf{x}_i$. We approximate the Gaussian of the patch by considering the measurements in the local neighborhood. To this end, we accumulate the points within a certain radius around the projection of $\mathbf{s}_i^k$ into global coordinates and compute the mean μ_i^k and the covariance Σ_i^k of this set of points. The surface patch is then given by the Gaussian $\langle \mu_i^k, \Sigma_i^k \rangle$. Figure 7.2 depicts the Gaussian distributions extracted on a range scan.

Furthermore, the Gaussian $\langle \mu_i^k, \Sigma_i^k \rangle$ allows us to estimate the normal vector of the surface. The estimate of the normal vector $\hat{\mathbf{n}}_i^k$ is the Eigenvector corresponding to the smallest Eigenvalue of Σ_i^k. We additionally orient $\hat{\mathbf{n}}_i^k$ towards the range scanner to obtain a unique approximation of the normal vector. By considering the ratio of the largest and the smallest Eigenvalue, we can decide whether the estimate of the normal vector is well-defined. In particular, a straight wall will lead to small ratio. Hence, we regard a normal as well-defined if the ratio is below a threshold, otherwise we do not estimate a surface patch for this beam.

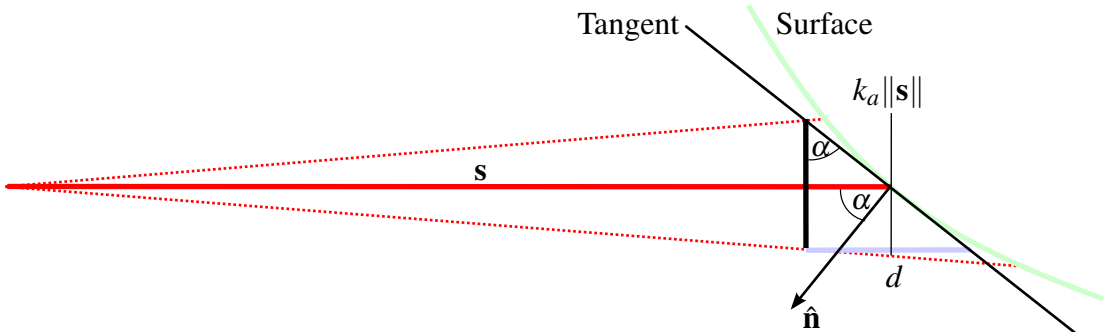


Figure 7.3: Laser beam hitting a surface. The figure illustrates the quantities which are involved in determining the sensor noise along the direction of the beam: $\mathbf{s}$, the laser beam; $\hat{\mathbf{n}}$, the normal of the surface; α, incidence angle; $k_a\|\mathbf{s}\|$: diameter of the beam $\mathbf{s}$ with a length of $\|\mathbf{s}\|$; and d, the length of the projection of the spot on the surface onto the line of the beam.

7.1.2 Sensor Model for Laser Range Finders

A laser range finder measures the range to the closest obstacle by emitting a laser beam and measuring the time until the reflection of the beam is received. Due to the optics involved the laser beam is actually a cone whose diameter increases with the distance traveled. Hence, the reflection of the beam the scanner receives is an aggregate corresponding to different distances. Typically, we can neglect this effect for rendering low-resolution maps.

Let us consider the case that the laser beam is perpendicular to a planar surface. In this situation, the area covered by the beam corresponds to a circle. If, however, the beam is not perpendicular to the surface, the area covered by the beam corresponds to an ellipse. The length of the semi axis of the ellipse is related to the angle between the beam and the normal vector of the surface. Thus, the incidence angle of the beam to the surface affects the area covered by the beam. On the other hand, the area covered by the beam has an influence on the error in the range measurement since the averaging is performed over the whole area. Consequently, our model of the noise in the range measurement has to include the incidence angle.

We have so far identified two sources for the error in a range measurement, namely the conic shape of the laser beam and the incidence angle. Typically, there is another source of (systematic) error in the range measurement. The distances returned by the range finder have a certain resolution, which leads to a quantization error ξ. The resolution of the range measurements is given in the data sheet of the scanner. A typical value for a laser scanner is ± 1 cm.

In Figure 7.3 we illustrate a laser beam hitting a surface. For a better readability we dropped the indices from the laser beam. The beam will result in a spot on the surface. Given our local approximation $\hat{\mathbf{n}}$ of the surface we are able to approximately compute the length d of the projection of this spot along the direction of the beam as

$$d \approx k_a\|\mathbf{s}\| \tan(\alpha), \tag{7.1}$$

where α is the incidence angle to the surface and k_a is the aperture of the cone of the laser beam. Let $\Gamma = \mathrm{diag}(\gamma_{11}^2, \gamma_{22}^2)$ be the covariance of the beam. The standard deviation γ_{11} along the beam $\mathbf{s}$ is proportional to the length d and we additionally account for the quantization error in that direction. Thus, we obtain

$$\gamma_{11} = k_{11} d + \xi. \tag{7.2}$$

Furthermore, the standard deviation γ_{22} in the direction orthogonal to the beam $\mathbf{s}$ is proportional

to the diameter of the beam. This yields

$$\gamma_{22} = k_{22}\|\mathbf{s}\|. \tag{7.3}$$

The parameters k_{11} and k_{22} are constant factors which depend on the laser range finder.

7.1.3 Data Association

As initial state for our method we consider the solution of a standard SLAM algorithm. This means the map estimate is consistent, i.e., patches corresponding to the same surface are close to each other but their location is affected by noise. To establish the correspondence between two patches $\langle \mu_i^k, \Sigma_i^k \rangle$ and $\langle \mu_j^l, \Sigma_j^l \rangle$, we utilize the "normal shooting" [36] method. The idea is to start the search from a surface patch $\langle \mu_i^k, \Sigma_i^k \rangle$, which has a well-defined normal. The search walks along the normal to find the closest patch $\langle \mu_j^l, \Sigma_j^l \rangle$ that has a similar normal. We restrict the search to 0.2 m around μ_i.

Furthermore, we enforce an additional sparsity heuristic. We only introduce a constraint for those pairs found by the above stated heuristic for which $i < j$. The association is updated after the optimization to be described in the next section converged.

7.1.4 Objective Function and Optimization

As mentioned above the overall goal of our approach is to create a joint optimization problem for the robot poses $\mathbf{x}$ and the map $\mathscr{M}$. The map in our case is a set of surface patches $\langle \mu_i^k, \Sigma_i^k \rangle$, where μ_i^k is measured by the beam $\mathbf{s}_i^k$ of robot pose $\mathbf{x}_i$. This relation is expressed by the term

$$\mathbf{e}_{meas}(\mathbf{x}_i, \mu_i^k, \mathbf{s}_i^k) = \left[h\left(\mathbf{x}_i, \mu_i^k\right) - \mathbf{s}_i^k \right]^\top \Gamma_{ik}^{-1} \left[h\left(\mathbf{x}_i, \mu_i^k\right) - \mathbf{s}_i^k \right], \tag{7.4}$$

where $h(\cdot)$ computes the expected measurement and Γ is the covariance (see Section 7.1.2).

Subsequent poses of the robot are constrained by the odometry measurement $\mathbf{u}_i$. As we assume Gaussian noise having the covariance Λ_i, we obtain the additional term

$$\mathbf{e}_{odom}(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i) = \left[g\left(\mathbf{x}_{i-1}, \mathbf{u}_i\right) - \mathbf{x}_i \right]^\top \Lambda_i^{-1} \left[g\left(\mathbf{x}_{i-1}, \mathbf{u}_i\right) - \mathbf{x}_i \right], \tag{7.5}$$

where $g(\cdot)$ is a function computing the pose given the previous pose and the odometry $\mathbf{u}_i$.

As a last element, we can exploit the data association of patches that are close to each other. The association yields a correspondence for surface patches $\langle \mu_i^k, \Sigma_i^k \rangle$ and $\langle \mu_j^l, \Sigma_j^l \rangle$ observed in different range scans taken from $\mathbf{x}_i$ and $\mathbf{x}_j$. We can define the error of the pair as the vector difference between the means of the Gaussians. For constructing the optimization, we need to weight this error. The idea is to take into account the local properties of each surface patch. Consider the situation in which both patches correspond to a straight wall. In this case, we would like to assign a larger weight to the error along the normal vector of the wall, whereas the error perpendicular to the normal should get a smaller weight. We are able to achieve this by weighting the distance between the patches according to sum of the inverses of the covariance matrices of the surface patches:

$$\Omega_{ij}^{kl} = \left(\Sigma_i^k\right)^{-1} + \left(\Sigma_j^l\right)^{-1}. \tag{7.6}$$

This leads us to the quadratic error for the pair which is given as

$$\mathbf{e}_{surf}\left(\mu_i^k, \mu_j^l\right) = \left(\mu_i^k - \mu_j^l\right)^\top \Omega_{ij}^{kl} \left(\mu_i^k - \mu_j^l\right). \tag{7.7}$$

With these error functions at hand we can setup the least squares estimation problem that seeks the robot poses $\mathbf{x}^*$ and the map $\mathscr{M}^*$ which best explains the set of constraints:

$$\langle \mathbf{x}^*, \mathscr{M}^* \rangle = \underset{\mathbf{x},\mathscr{M}}{\operatorname{argmin}} \sum_i \mathbf{e}_{odom}(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i) + \sum_{ik} \mathbf{e}_{meas}(\mathbf{x}_i, \mu_i^k, \mathbf{s}_i^k) + \sum_{ijkl} \mathbf{e}_{surf}(\mu_i^k, \mu_j^l). \tag{7.8}$$

For optimizing Eq. 7.8 we apply the Gauss-Newton algorithm as described in Chapter 2. Despite the large number of variables involved in the optimization, we can exploit the sparsity. The sparseness arises from the limited range of the sensors. Thus, only variables that are spatially close to each other have a constraint between them. This allows us to solve the problem efficiently, for example, our current system carries out one iteration of nonlinear optimization of a system consisting of 172,522 surface patches acquired from 616 robot positions, 996,451 surface constraints, and 671,550 constraints between surface patches in less than 5 seconds using one core of an Intel Core Quad running at 2.6 Ghz.

In addition to the sparseness, we do not update the local properties of the surface patches, i.e., the normal vectors and the covariances stay the same and are only update before the next optimization problem is constructed. As we store the covariances of the surface patch with respect to the observing robot pose, we can account for an update in the pose by rotating the covariance matrix accordingly.

7.2 Comparison with ICP and Bundle Adjustment

Our method is closely related to Bundle Adjustment (BA). Within BA, however, the landmarks are associated to each other considering the rich descriptiveness of visual features, such as SIFT [142] or SURF [16]. In contrast to this, ICP [19] registers two set of points $\mathscr{S}$ and $\mathscr{S}'$. To this end, it defines the data association by a nearest neighbor method given the currently estimated transformation. In our application, the initial transformation arises from the odometry of the robot. The data association yields the set $\{\langle \mathbf{l}_1, \mathbf{l}'_1 \rangle, \ldots, \langle \mathbf{l}_N, \mathbf{l}'_N \rangle\}$ of corresponding points from the two sets. Let $\langle R, \mathbf{t} \rangle$ describe the motion that moves the local frame of $\mathbf{x}_2$ into $\mathbf{x}_1$, then ICP determines the best motion *given* the correspondences as

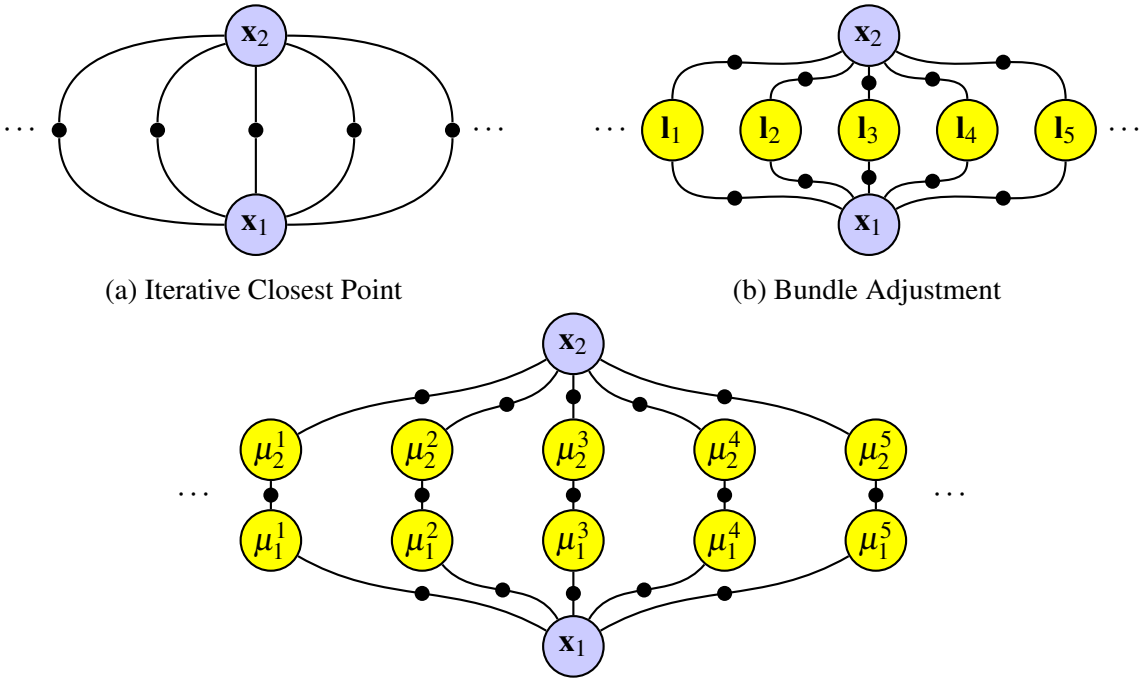
$$\langle R^*, \mathbf{t}^* \rangle = \underset{R,\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{N} \| \mathbf{l}_i - \left(R\mathbf{l}'_i + \mathbf{t} \right) \|^2, \tag{7.9}$$

which can be solved in closed form [212] and its graphical model is given in Figure 7.4a. Using the estimate $\langle R, \mathbf{t} \rangle^*$ the correspondences are updated and the whole process is iterated. Since the points itself are not updated, the merged cloud accumulates the noise contained in the individual range measurements.

On the other hand, BA estimates the motion — up to scale in a monocular setting — between two cameras utilizing the features visible in both images. In Figure 7.4b we illustrate the graphical model of BA. Here, the features are part of the optimization and their position is updated. The optimization, however, does typically not consider the surface properties. In addition to that, BA relies on a projective geometry to model the error in the pixel location of the observed feature in the camera image. More formally, BA can be defined as

$$\langle \mathbf{x}^*, \mathbf{l}^* \rangle = \underset{\mathbf{x},\mathbf{l}}{\operatorname{argmin}} \sum_{ij} \| \mathbf{h}(\mathbf{x}_i, \mathbf{l}_j) - \mathbf{z}_{ij} \|^2, \tag{7.10}$$

where $\mathbf{x}$ and $\mathbf{l}$ represent the positions of the cameras and the landmarks, respectively, $\mathbf{h}(\cdot)$ calculates the expected pixel location of a feature $\mathbf{l}_j$ in a camera image taken at location $\mathbf{x}_i$, and



(a) Iterative Closest Point

(b) Bundle Adjustment

(c) our approach

Figure 7.4: Comparison of the graphical models between Iterative Closest Point, Bundle Adjustment and our approach. For convenience we dropped the factor which relates the subsequent poses by the odometry of the robot.

$\mathbf{z}_{ij}$ is the measured pixel location. For laser based mapping as done by our method, we have to consider that 2D range data is not as descriptive as visual data. Thus, we cannot commit ourselves easily to a data association. We therefore choose to update the correspondences based on the nearest neighbor method in each iteration similarly to ICP.

Figure 7.4c depicts the graphical model underlying our method. Note that we dropped the sequential odometry factor for convenience. Here, a landmark corresponds to a single surface patch and it is associated to its nearest neighbor in the subsequent range scan. After solving the current graph, we update the associations as outlined above and iterate the whole process. Consequently, our method can be regarded as following the Expectation-Maximization algorithm. Moreover, opposed to BA our approach does not enforce an equality constraint on the corresponding surface patches. We instead allow them to slide along the tangent direction of the surface. Furthermore, we are able to determine the maximum likelihood transformation between two range scans as done by ICP. We, however, also reduce the noise in the range readings instead of accumulating it in the merged point cloud.

7.3 Experiments

The approach described above has been implemented and in this section we will present experiments to highlight the performance of our approach. For our experimental evaluation we considered a set of publicly available real-world data sets. In particular, the Freiburg building 079, the Intel Research Lab, the MIT CSAIL building, and the Aces building data set. This set is heterogeneous as it was recorded with different range finders. Hence, we can demonstrate that our approach is applicable to different range finder models. Please note that the approach presented in this chapter is not meant as a solution to the entire SLAM problem. It is rather

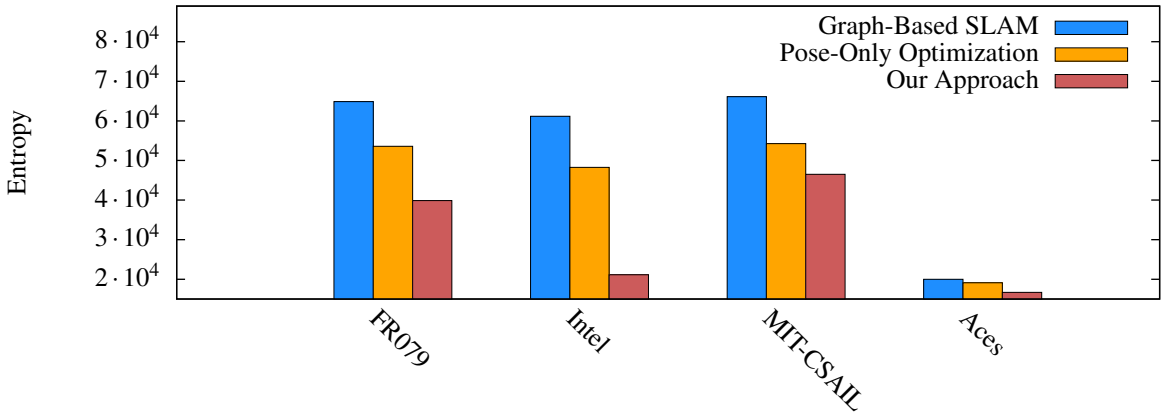


Figure 7.5: Entropy of the maps. On each data set the optimization results in a reduction of the entropy, whereas our method results in the lowest value on each data set (see [182]).

a post-processing step to be applied on-top of a SLAM solution for obtaining highly accurate models of an environment. To this end, the input in all our experiments is given by running the graph-based mapping discussed in Chapter 2.

7.3.1 Map Accuracy

In a first set of experiments we evaluate the accuracy of the maps obtained by jointly optimizing the poses of the robot and the range readings collected by the robot. For evaluating the accuracy of the maps generated by our approach, we consider the maps obtained by a standard graph-based SLAM algorithm as a baseline. The maps are obtained by the standard occupancy grid algorithm [205]. We measure the accuracy of the map by the entropy of the maps generated by different techniques. The entropy of the map measures the uncertainty about the state of the environment. By exploiting the assumption that the cells of an occupancy map $\mathbf{m}$ are independent, we can determine the entropy of a map, which is a grid of binary random variables, as

$$H(\mathbf{m}) = -\sum_{c \in \mathbf{m}} \log_2 p(c) \cdot p(c) + \log_2 (1 - p(c)) \cdot (1 - p(c)). \tag{7.11}$$

In addition to the baseline given by the standard SLAM algorithm, we consider the following two optimization results generated by our approach. First, we consider the scans as rigid and only optimize the poses of the robot. We refer to this method as pose-only in the following. The outcome of this method demonstrates how much the standard graph-based SLAM solution could be improved by updating the data associations and as a consequence improve the matching between the individual range scans. Second, the joint optimization of robot poses and the laser readings. In contrast to the pose-only optimization the joint formulation additionally accounts for the noise in the range scans. For all three methods we generated grid maps having a resolution of 5 mm. The impact of the optimization on the entropy of the maps can be seen in Figure 7.5. Our approach results in a substantial reduction of the entropy and hence the uncertainty in the maps. The decrease of the entropy in the map corresponds to the reduction of the noise in the range readings by jointly optimizing their position. The optimization takes into account that each point corresponds to a sample of the piece-wise smooth surface. This further reduces the uncertainty compared to the pose-only optimization.

Figure 7.6 and Figure 7.7 show the maps obtained by the three methods described above. As we can see from the magnified views in the lower parts of the figures, the maps generated by our method show an accuracy that is superior to both other approaches.

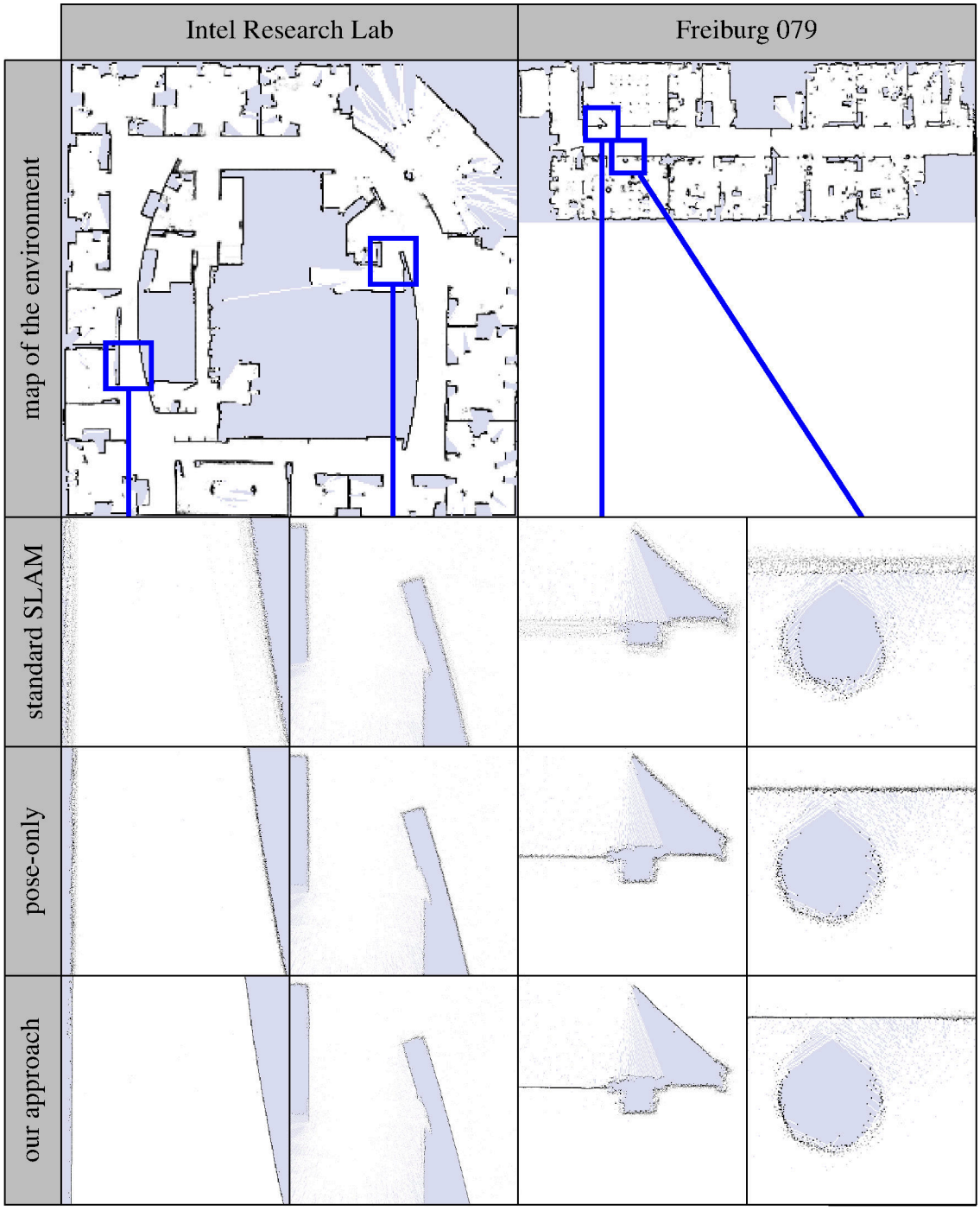


Figure 7.6: Maps generated by each approach. Within each column, the image on the top shows the full map. The three rows underneath show magnified partial views of the map on the top: Standard SLAM (second row), pose-only optimization (middle row), our approach (bottom row). The visual inspection reveals the high accuracy of the maps generated by our method (see [182]).

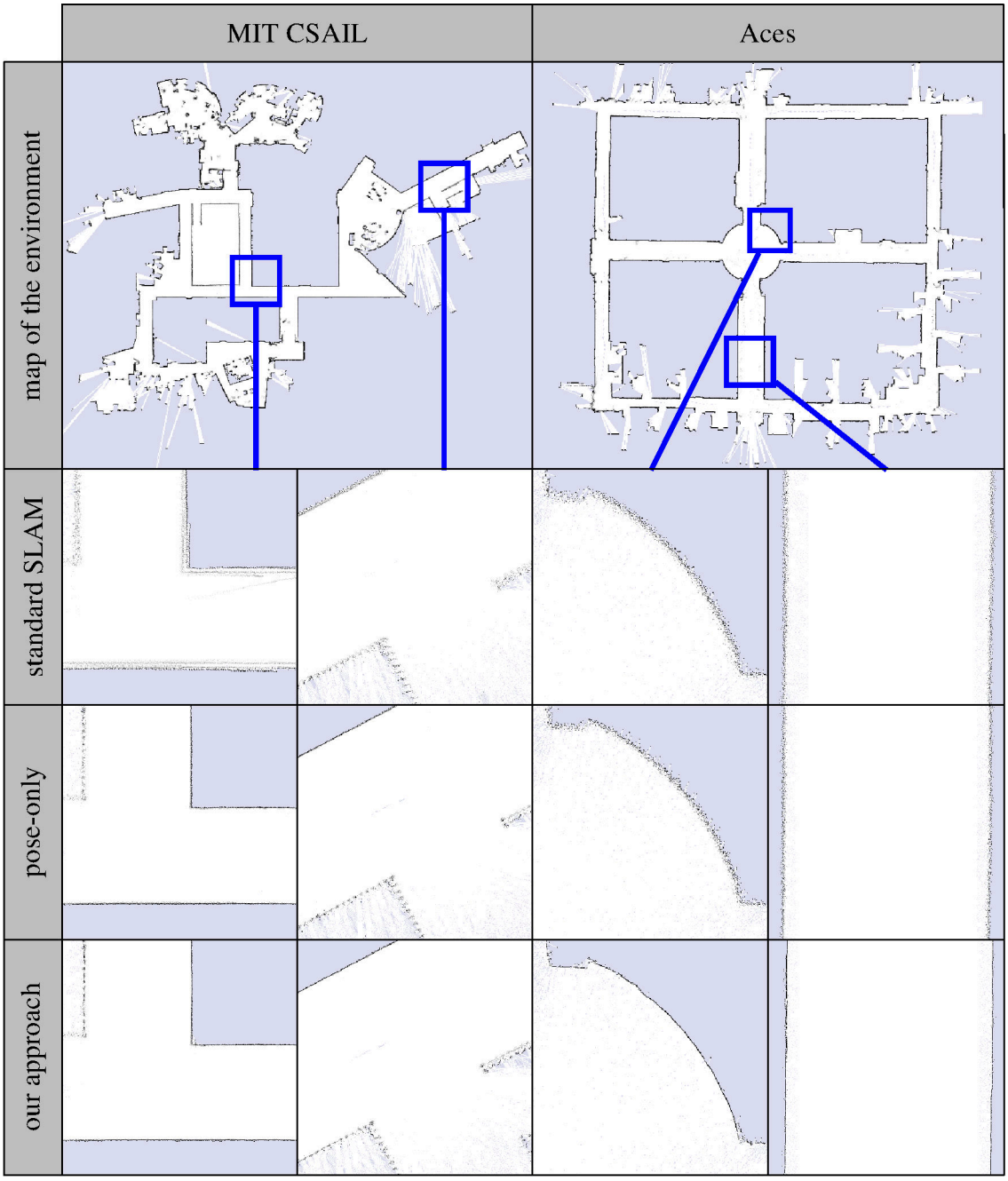
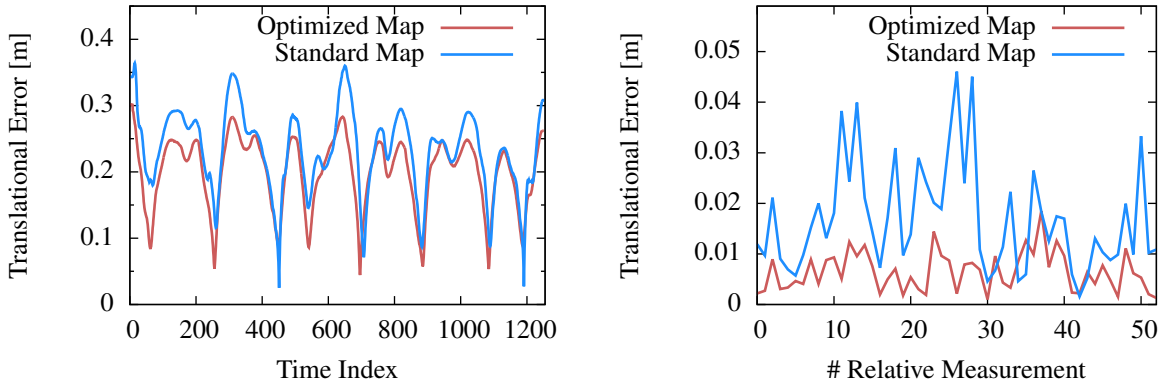


Figure 7.7: Maps generated by each approach. Within each column, the image on the top shows the full map. The three rows underneath show magnified partial views of the map on the top: Standard SLAM (second row), pose-only optimization (middle row), our approach (bottom row). The visual inspection reveals the high accuracy of the maps generated by our method (see [182]).



(a) absolute position error (b) relative position error

Figure 7.8: Localization error on the map generated by standard SLAM and our approach, respectively. (a) The absolute error in each time step. (b) The relative position error for subsequent poses with a distance of 1 m in between (see [182]).

7.3.2 Impact of the Map Accuracy on Localizing the Robot

To further illustrate the practical benefits of the highly accurate maps generated by our approach, we evaluated the position accuracy of a localization algorithm utilizing the map of a standard SLAM algorithm and our approach, respectively. As ground-truth data is not available for the public data sets considered in the former experiment, we utilized a simulator for collecting data. To this end, we created a map having a cell size of 5 mm $\times$ 5 mm which we used as environment for the simulator.

Utilizing the simulator we recorded two data sets. The first one served as input for generating a map with a standard SLAM algorithm and our approach. We conducted a Monte Carlo localization [42] run on both maps taking the second data set as input. To quantitatively evaluate the accuracy of the localization on the respective map, we considered two different metrics. The metric suggested by Olson and Kaess [167] evaluates the Root Mean Squared error (RMS) of the absolute location of the robot, whereas as second measure we utilize the metric based on the relative position error (RPE) as suggested in Chapter 4. Figure 7.8a shows the translational error in time step of the localization run with respect to the ground truth position of the robot. Thus, this error captures the error in the position as it is estimated by MCL and also inaccuracies in the map. The RMS for the MCL on the standard map is 0.0636 m, while on the map generated by our approach it is 0.0469 m. In contrast to this, Figure 7.8b depicts the RPE of pairs of poses that lie sequentially on the trajectory of the robot. Each of this pairs spans approximately one meter. As we can see, the relative error is substantially smaller for MCL operating on the map generated with our approach. The higher accuracy of our maps is also confirmed by the overall lower RMS. Taking all this into account, the maps generated by our approach allow a robot to more accurately localize itself compared to the position estimates obtained with a standard map.

7.4 Related Work

Several approaches have been proposed in the past to characterize the error affecting the result of scan-matching algorithms and thus to accurately estimate the constraints of the optimization problem. Bengtsson *et al.* [17], proposed to analyze the residual of the re-projection error around the minimum by either sampling the error function or by approximating it by a quadratic form. Subsequently, Censi [31] presented a closed form minimization algorithm to determine

this covariance. The majority of matching algorithms relies on point-to-point correspondences.

In contrast, Olson [165] proposed a hierarchical correlative algorithm derived from the method suggested by Konolige and Chou [109]. This matching algorithm computes the histogram of all possible robot positions around the minimum. Since a grid is considered to build a histogram approximation of the likelihood function, the algorithm does not rely on specific point-to-point correspondences. Furthermore, it can also estimate the uncertainty of the registration.

Segal *et al.* [188] presented a variant of 3D ICP scan-matching that minimizes the matching error between corresponding planar patches extracted from the input scans. This is in particular useful to model the decrease in the density of data points which coincides with an increase in the range. Biber *et al.* [20] proposed to solve the scan-matching problem by approximating the reference scan by a set of Gaussians. In contrast to the point-to-point criterion minimized by traditional ICP scan matchers [144, 19], the Gaussians weight the error along different directions based on the shape of the matched surfaces. Subsequently, Magnusson *et al.* [147] extended this approach to 3D.

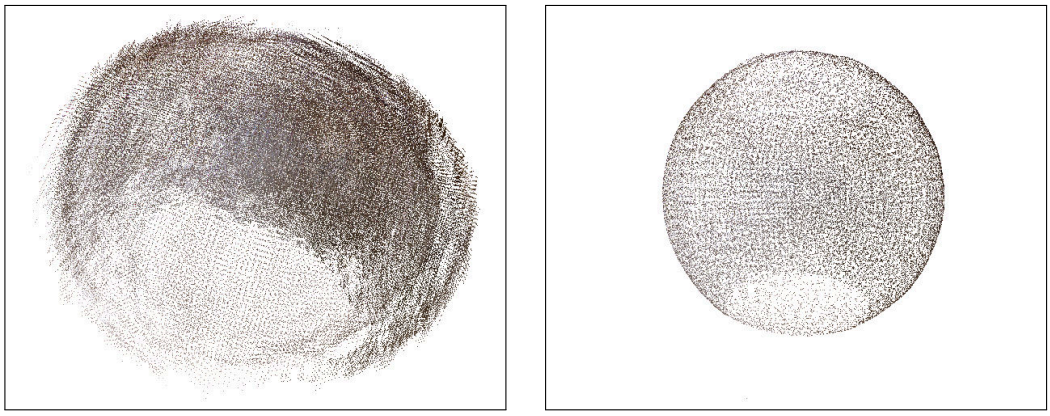
The methods mentioned above focus on accurately registering two scans, but they treat them as rigid bodies. Consequently, those methods do not attempt to refine the points of the individual scans. As we have seen in the experiments, jointly optimizing the range readings boosts the accuracy of the maps since it explicitly takes into account the noise in the range data itself. Our approach shares aspects with traditional Bundle Adjustment problems in computer vision [210] (see also Section 7.2). In contrast to Bundle Adjustment, we do not explicitly rely on feature correspondences, but we improve the data associations after each iteration.

Several researchers extended ICP for non-rigid objects. For example, Haehnel *et al.* [85] presented an extension to the classical ICP algorithm for matching non-rigid bodies. The non-rigid ICP variant proposed by Li *et al.* [135] is able to deform all points on a surface graph (data) onto a second surface graph (model) by introducing a deformation model and applying least square optimization to find the optimum for deformation and registration. Since this method uses a weighted least square approximation for the model, it is not easily transferable to the case of multiple scans and does not account for systematic sensor errors, while our method does not explicitly consider deformations.

Fleishman *et al.* [61] presented Robust Moving Least Squares, a smoothing technique based on robust regression. They assign the points to piecewise smooth surfaces and are able to obtain accurate models. In a similar context, Andersen *et al.* [7] propose a Markov Random Field formulation that optimizes the parallelism between neighboring surface elements and their overlap. This smooths out noise while maintaining sharp features. Both methods can be used to obtain accurate surface models, but ignore the model of the sensor that has been used to generate the point cloud and do not optimize over the sensor poses.

7.5 Conclusions

In this chapter, we presented an approach to improve the accuracy of maps generated by standard SLAM methods. We achieved the highly accurate maps by constructing a joint optimization problem which incorporates the position of the robot as well as the laser readings. For modeling the noise in the distance measured by the laser scanner, we considered the underlying principle of the laser beam, namely its conic shape and the influence of the incidence angle on the surface. This together with interpreting the range measurements as samples allowed us to better model the noise in the range data.



(a) initial state (b) result of our approach

Figure 7.9: This figure illustrates the application of an extension of our method to the data given by a Kinect sensor. The image depicted in (a) shows the input data obtained by moving around a sphere. The image in (b) depicts the resulting point cloud after optimization (see [184]).

Our experimental evaluation on a set of publicly available data sets illustrates the high accuracy of the maps obtained by our method. Our approach furthermore improves the achievable localization accuracy if the localization algorithm operates on a map generated by our approach compared to utilizing the map of a standard SLAM algorithm. Furthermore, we successfully extended the method to operate on the data of a Kinect depth sensor as well as 3D range sensors as illustrated in Figure 7.9. The details of the extension can be found in our joint work [184].

Part II

Applications

Chapter 8

Navigation with a Car in Complex Urban Environments

Recently, the problem of autonomous navigation of automobiles has gained substantial interest. Autonomous cars have been shown to robustly navigate over extended periods of time through complex desert courses or through dynamic urban traffic environments. The robots in these tasks, however, typically rely on GPS traces to follow pre-defined trajectories so that only local planners are required. In this chapter, we present an approach for autonomous navigation of cars in urban structures, such as parking garages, where GPS is not available. Our approach builds upon the state estimation techniques presented in this thesis to estimate a model of the environment, which is considered for path planning and localization in the GPS-denied part of the environment. It furthermore employs a local path planner for controlling the vehicle. In a practical experiment carried out with an autonomous car in a real parking garage we demonstrate that our approach allows the car to autonomously park itself in a large-scale multi-level structure.

• • • • • • • • • • • •

In the second part of this thesis, we will build upon and extend the techniques which have been presented throughout the previous chapters to realize autonomous navigation with robots in complex and challenging settings. In particular, we will first present a navigation system for an autonomous car and subsequently we will elaborate a system for a robotic pedestrian assistant. Both envisioned applications impose different challenges for the navigation system. The realized systems demonstrate what can be achieved by the techniques presented in this thesis in challenging real-world scenarios.

In recent years, the problem of autonomous navigation of automobiles has gained substantial interest in the robotics community, particularly due to the two "Grand Challenges" organized by DARPA. There also is a wide range of civilian applications, for example, in the area of driver assistance systems that enhance the safety by performing autonomous maneuvers of different complexities including adaptive cruise control or emergency breaking. During the two Grand Challenges, cars have been shown to navigate reliably along desert courses and in dynamic urban traffic scenarios. Most of these scenarios rely on GPS data to provide an estimate about the pose of the vehicles on pre-defined trajectories, though. Therefore, only local planners [46, 137] were needed to control the vehicles. Autonomous navigation in environments

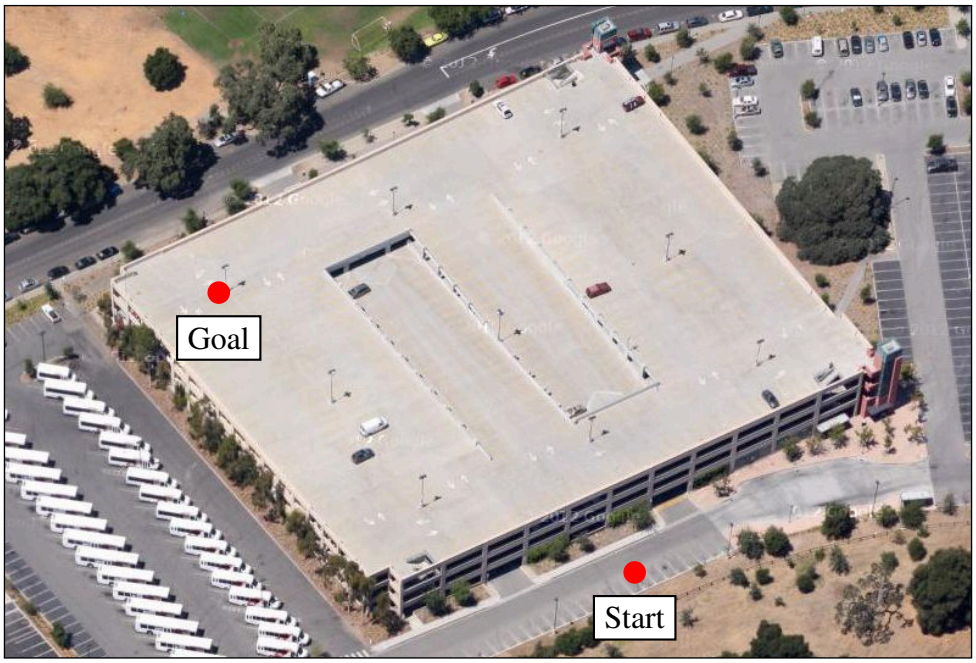


Figure 8.1: Multi-level parking garage used for the experiment. The garage has four levels. The start point was close to the entrance, the goal point is on the upper level (aerial image © Google).

without sufficiently accurate GPS signals, such as in parking garages, and the connection to the navigation in GPS-enabled settings, however, have not been sufficiently well targeted in these challenges. Here, the state estimation techniques presented in this thesis bridge the gap by providing means to robustly and accurately map the GPS-denied environment the vehicle has to operate in.

In this chapter, we present an approach to autonomous automotive navigation in large-scale GPS-denied structures with potentially multiple levels. This problem is relevant for a variety of situations. for example, for autonomous parking behaviors or for navigation systems that provide driver assistance even within buildings and not only outdoors where a sufficiently accurate GPS signal and detailed road network information is available. Even state-of-the-art inertial navigation systems are not sufficient to accurately estimate the position of the robot in large-scale indoor structures, such as the one depicted in Figure 8.1, which is the parking garage used for our experiments. Particularly, the techniques developed in this thesis provide us with the ability to acquire accurate maps of such large mixed outdoor and indoor environments. To enable a mobile vehicle to park itself at an arbitrary, pre-defined position in that garage given it starts at the entrance of the building at the lowest level, several requirements need to be met. First, the vehicle needs an appropriate representation of the building to calculate the path to be taken. Second, it needs to be able to localize itself in this three-dimensional building. Third, it needs to be able to follow this path, so that it safely arrives at the designated target location.

Our approach considers multi-level surface maps [209] to compactly model the environment. We apply the graph-based optimization procedure presented in this thesis to establish the consistency of this map. The map allows us to plan a global path from the start to the goal position and to robustly localize the vehicle based on laser range measurements. We furthermore employ a local planner to follow this path and to avoid unforeseen obstacles. As a result, the vehicle can autonomously navigate in such multi-level environments without any additional information provided by a user. Our approach has been implemented and tested with a real Volkswagen Passat Wagon.

Figure 8.2: Local MLS-map example. MLS-maps provide means to efficiently represent the drivable surface and the vertical objects, which are useful for localization. Analyzing the height variations allows us to detect curbs, walls, and other obstacles, which are visualized in red, while the drivable surface is shown in yellow.

The remainder of this chapter is organized as follows. First, in Section 8.1 we describe the underlying map data structure and the mapping algorithm itself that allowed us to map the complex multi-level parking structure. After presenting the localization algorithm in Section 8.2, we briefly describe the planning framework in Section 8.3. The experimental results illustrating the abilities and advantages of our approach are outlined in Section 8.4. Finally, in Section 8.5 we discuss the related work.

8.1 Mapping of the Parking Garage

In this section, we describe our approach for mapping large urban structures like a multi-level parking garage. We will discuss the data structure used to model the complex environment. The SLAM solution is obtained by a graph-based optimization technique similar to the one presented in Chapter 2. Furthermore, we propose a heuristic for detecting the level of the environment. The information about the level is an important information as it allows us to reset the content of certain data structures of our system which are implemented on 2D data structures for efficiency reasons.

8.1.1 Map Representation

We consider Multi-level Surface (MLS) maps [209] to model the environment in 3D. An MLS map is based on a 2D grid, where each cell c_{ij} with $i, j \in \mathbb{Z}$ stores the vertically occupied space in a list $P_{ij}^1, \ldots, P_{ij}^L$ of so-called surface patches. Each surface patch models a vertically occupied volume of space. To this end, it saves a height estimate μ_{ij}^l along with the variance σ_{ij}^l to model the uncertainty in the height of the surface. Additionally, the vertical extend of the surface patch is given by a depth value. In particular, such a patch indicates the possibility to traverse the cell at a specific height. In Figure 8.2 we see one example of an MLS map. This map was acquired by the vehicle by merging 15 point clouds obtained by the Velodyne LIDAR mounted on the roof of the vehicle. The map depicts one of the entrances of the multi-story car park in which

we carried out the experiments. In addition to the 3D representation of the world, the MLS map allows us to identify the drivable surface and obstacles, which are indicated in yellow and red in Figure 8.2, respectively. We obtain this classification by analyzing the height variation between adjacent cells. Despite the rich set of information encoded in the MLS map, the representation of complex environments can be achieved in a compact manner. For example, the data set shown in the experiments consists of approximately 4 GB of raw data, while the complete MLS-map with a cell size of 20 cm requires only around 120 MB. This amount of data easily fits into the main memory of modern computers and could in practical applications easily be downloaded from the information system of the garage to the navigation system of the car.

8.1.2 Mapping with Graph-Based SLAM

Our mapping system addresses the SLAM problem by its graph-based formulation. As we have previously outlined, a node of such a graph represents a 6DoF pose of the vehicle and an edge between two nodes models a spatial constraint between them. To compute the spatial configuration of the nodes that best satisfies the constraints encoded in the edges of the graph, we apply the graph-based SLAM formulation presented in Chapter 3, which allows us to estimate the maximum likely trajectory online while the robot explores the environment. Performing this optimization procedure reduces the uncertainty in the pose estimate of the robot whenever constraints between non-sequential nodes are added, i.e., a loop is closed.

In detail, the graph is constructed as follows: whenever a new observation $\mathbf{s}_t$ has to be incorporated into the system, we create a new node at position $\mathbf{x}_t$. We then create a new edge $\mathbf{e}_{t-1,t}$ between the current position $\mathbf{x}_t$ and the previous one $\mathbf{x}_{t-1}$. This edge is then labeled with the transformation $\mathbf{z}_{t-1,t}$ between the two poses $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$. We determine the virtual measurement $\mathbf{z}_{t-1,t}$ by scan-matching the 3D measurements $\mathbf{s}_{t-1}$ and $\mathbf{s}_t$. To this end, we use a variant of the Iterative Closest Point algorithm (ICP) [19, 188] to obtain a maximum likelihood estimate of the robot motion between the subsequent observations. In our implementation, we perform ICP on local MLS-maps instead of raw 3D point clouds. Additionally, we exploit the classification of points into walls and flat surface as given by the MLS-map data structure. We constrain the correspondence search within the ICP algorithm to comply with the classification.

Whereas this procedure significantly improves the estimate of the trajectory, the error of the current robot pose tends to increase over time due to the accumulation of small residual errors. This effect becomes visible as alignment errors in the map when the vehicle revisits already known regions. To solve this problem, we need to re-localize the robot in a region of the environment which has been visited long before. To resolve these errors, i.e., to close the loop, we apply our scan-matching technique on our current scan $\mathbf{s}_t$ and a former scan $\mathbf{s}_i$, where $i \ll t$. This yields the virtual measurement $\mathbf{z}_{i,t}$ which describes the motion between the poses $\mathbf{x}_i$ and $\mathbf{x}_t$. Consequently, we add the resulting constraint to the graph. To detect a potential loop closure, we identify all former poses that are within a constant ellipsoid and whose observations overlap with the current observation. If a match is found, we augment the graph by adding a new edge between $\mathbf{x}_i$ and $\mathbf{x}_t$ labeled with the relative transformation which is computed by matching the corresponding observations $\mathbf{s}_i$ and $\mathbf{s}_t$.

This procedure works well as long as the relative error between the two poses lies within the error ellipsoid. In general, the longer the path between the two nodes $\mathbf{x}_t$ and $\mathbf{x}_i$ is, the higher the relative error becomes. So that our strategy might fail because ICP converges to a wrong alignment of the scans. In the absence of a globally consistent position estimate like GPS, which would allow us to restrict the search of loop closures, one might use more sophisticated techniques [39, 84, 198] to robustly identify the places. While we acknowledge this drawback,

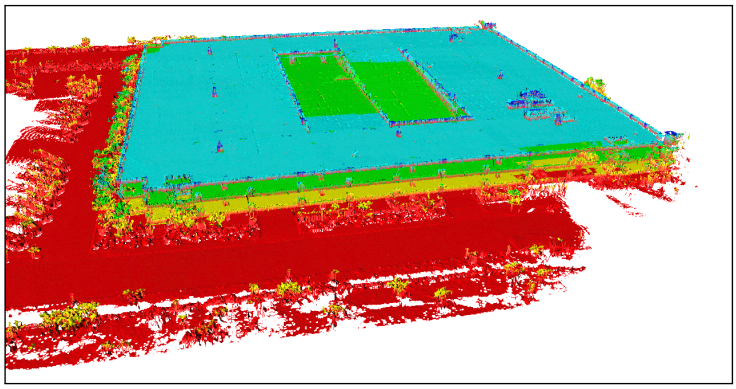


Figure 8.3: This image shows a partial view of the map of the parking garage in which color encodes the level assigned to each surface patch. As we can see, our heuristic correctly estimates the level of most patches, while only some very small areas obtain a wrong level assignment.

during our experiments our strategy never introduced a wrong constraint and the visual inspection of the resulting map always revealed that the map is consistent.

8.1.3 Level Information

As mentioned above, the MLS-map stores information about the surface which encloses the occupied volume of the environment. We can exploit this fact to assign progressive IDs to the different drivable levels in the map by a heuristic. We refer to a multi-level environment if the robot has the possibility to reach a specific cell c_{ij} of the map at least at two different height levels.

To label the environment, we proceed as follows: First, we generate a connectivity graph $\mathscr{C}$ of the surface described by the MLS map. This is done by connecting all the patches $P_{ij}^1, \ldots, P_{ij}^L$ of the cell c_{ij} with their neighboring patches of the cells $\{c_{i+k,j+l} \mid k,l \in \{0,1\}, k+l \geq 1\}$ with an undirected edge. The edge is only introduced if the height difference between the two surface patches is below a threshold, which is given by the characteristics of the robot.

This connectivity graph is then used to initiate a region growing procedure. The initial frontier consists of the lowest surface patch. We keep the frontier sorted according to the surface height. The algorithm keeps track on how often each cell has been visited at different height levels. This counter is initialized to zero and it is incremented whenever the cell is reached. Additionally, we verify that the counter of the predecessor is not larger than the counter of the current patch. In such a situation, we update the counter to the value of the predecessor. This is required as the current cell may contain surface patches that do not belong to the drivable surface. By this procedure, we walk along the surface modeled in the map and each time the frontier reaches a cell, the counter of the cell corresponds to the level, which we assign to the respective surface path. As we can see in Figure 8.3, our heuristic works well on the map of the parking garage and provides us the required information. We correctly identified the four drivable levels of the parking garage, while our heuristic only fails in small parts of the map.

In our approach, the level information is needed for triggering the update of local two-dimensional data structures of our system. Whenever the car moves to a different level, we need to recompute these structures according to the current level. An example of such a two-dimensional data structure which needs to be updated upon a change of the level is the two-dimensional obstacle map, which is considered by the planning components of the system.

8.2 Localization

Whenever the GPS signal is lost, even high-end integrated navigation units as the one considered in our experiments are not able to accurately localize the robot. Such navigation systems combine GPS, wheel odometry via a distance measurement unit, and inertial measurements to obtain a global position expressed in latitude, longitude, and altitude. In this case, the lack of GPS measurements prevents the system to compensate for the drift in the (x, y, z) position. This drift results from the integration of small errors affecting the relative measurements obtained by the encoders and the inertial sensors. In particular, we observed a significant error along the z component of the pose vector, which represents the height of the vehicle. This high error does not allow us to directly use the z estimate of the inertial system to estimate the level in case of multi-level indoor environments.

To this end, we instead consider probabilistic localization on a map following the Bayesian filtering scheme. We apply a technique similar to the one presented in Chapter 6. Here, we, however, have to estimate the position of the robot in 3D. Thus, we briefly recapitulate the underlying concepts. The key idea is to maintain a probability density $Bel(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{s}_{1:t}, \mathbf{u}_{1:t})$ of the position of the robot $\mathbf{x}_t$ at time t given all the so far collected sensor measurements $\mathbf{s}_{1:t}$ and all control inputs $\mathbf{u}_{1:t}$. This posterior is updated as follows:

$$Bel(\mathbf{x}_t) = \eta\, p(\mathbf{s}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})\, Bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}, \tag{8.1}$$

where η is a normalization factor. For implementing the scheme given in Eq. 8.1, we have to specify the sensor model $p(\mathbf{s}_t \mid \mathbf{x}_t)$ and the prediction model $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$.

The inertial navigation system integrates accelerations, changes in rotation, and the wheel velocity, which is measured at the rear axle of the vehicle to calculate the open-loop position of the vehicle. This yields a locally consistent estimate, which is updated at 200 Hz but drifts over time. Additionally, we observed that the orientation, namely the roll, pitch, and yaw angles, is measured with high precision and also without drift by the system. We also found that the pose estimate is not significantly improved by filtering the attitude. Hence, we reduce the localization problem to three dimensions instead of six.

Once again, we use Monte Carlo localization (MCL) [42] for implementing the filtering algorithm. MCL utilizes a finite set of weighted samples, the so-called particles, to represent the posterior. Each particle represent a possible pose of the robot and has an assigned weight $w_{[i]}$. The scheme given in Eq. 8.1 is implemented by executing two alternating steps:

1. In the prediction step, we update the position of each particle by sampling a motion from the motion model $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$.

2. In the correction step, we assign a new weight $w_{[i]}$ to each particle by evaluating $p(\mathbf{s}_t \mid \mathbf{x}_t)$ based on the current observation $\mathbf{s}_t$.

Finally, the set of particles needs to be re-sampled to focus the finite number of particles in the regions of high likelihood. This ensures that the particle set yields a good approximation of the pose distribution. Re-sampling may also drop good samples, which is known as particle impoverishment [205]. We therefore consider a criterion to decide whether to re-sample or not. To this end, we evaluate the number of effective particles as proposed in [47, 77] as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \widetilde{w}_{[i]}^2}, \tag{8.2}$$

where N denotes the number of particles and $\widetilde{w}_{[i]}$ corresponds to the normalized weight of the particle such that $\sum_{i=1}^{N} \widetilde{w}_{[i]} = 1$. The idea is to only re-sample, when N_{eff} drops below the threshold of $N/2$.

The prediction model $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is implemented by drawing a 3D motion vector from a Gaussian whose mean $\mathbf{u}_t$ is given by the relative 3D motion vector of the inertial system since time step $t-1$. In addition, we constrain the height coordinate z of each particle to remain in a boundary around the surface modeled by the map. This is motivated by the fact that positions above or below the surface are not admissible and we want to focus the limited number of particles in the high density regions of the probability density function. In all our experiments this boundary was set to 20 cm around the drivable surface.

The range measurement $\mathbf{s}_t$ is integrated to calculate a new weight for each particle according to the sensor model $p(\mathbf{s}_t \mid \mathbf{x}_t)$. In our sensor model, we treat each beam independently. The likelihood of a whole laser scan is hence given by

$$p(\mathbf{s} \mid \mathbf{x}) = \prod_{k=1}^{K} p(\mathbf{s}^k \mid \mathbf{x}), \tag{8.3}$$

where K is the number of beams in each measurement $\mathbf{s}$ from the laser sensor. In Eq. 8.3 and in the following, we drop the index t for convenience. Our sensor model $p(\mathbf{s}^k \mid \mathbf{x})$ is based on an approach that is known as likelihood fields or endpoint model [205] extended to 3D.

In the endpoint model, the probability $p(\mathbf{s}^k \mid \mathbf{x})$ depends on the distance d^k between the endpoint of the k-th laser beam and the closest obstacle in the map, as we have seen in Chapter 6. The model just considers the endpoints and does not take into account the beam characteristic of the laser. This leads to a sensor model that can be evaluated efficiently as no expensive ray-casting operation needs to be carried out in the 3D map. MLS maps model the environment by a collection of vertical surface patches. Each vertical segment represents an occupied volume. As there is no efficient way to determine the closest vertical segment for a query point, we approximate all vertical segments by sampling a set $\mathscr{S}$ of 3D points. Then the distance d^k of the beam endpoint $\mathbf{p}^k$ to the closest obstacle in the map is approximated as the distance $d(\mathbf{p}^k, \mathscr{S})$ between $\mathbf{p}^k$ and $\mathscr{S}$. The distance $d(\mathbf{p}^k, \mathscr{S})$ can be efficiently determined by means of a kD-tree [18]. With this approximation and by considering the fact that $p(s^k \mid \mathbf{x})$ is Gaussian, we compute likelihood as

$$p(\mathbf{s}^k \mid \mathbf{x}) \approx \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{d(\mathbf{p}^k, \mathscr{S})}{\sigma}\right)^2\right], \tag{8.4}$$

where $\sigma = 1\,\text{m}$ in our current implementation. Plugging Eq. 8.4 into Eq. 8.3, we obtain the entire sensor model.

Finally, we bootstrap our localization algorithm by initializing the particles based on the GPS measurements provided by the vehicle when driving outside. Therefore, the map of the environment also contains the surrounding outdoor area where the GPS signal is available. In the outdoor part of the map, the high-end navigation system provides a GPS estimate that is sufficiently accurate to initialize the particle filter for position tracking.

8.3 Path Planning

Given a search space, in our case the surface connectivity graph $\mathscr{C}$ described in Section 8.1.3, we want to find a feasible path between a starting and a goal location. By defining a cost function

for each motion command the robot can execute and an admissible heuristic which efficiently guides the search, we can use A* to search for the path. By exploiting the classification of the drivable surface the A* search guides the robot towards the goal.

Given a global path through the multi-level graph $\mathscr{C}$, we use a local planner to navigate through the current level of the structure, i.e., we follow the global path. The local planner operates on the 2D obstacle map that is created online by the system. The implementation of the local planner considered for the experiments is the one used in the DARPA Urban challenge [46]. For completeness, we briefly outline the main components of this planner below.

The task of the local planner is to find a safe, kinematically feasible, near-optimal in length, and smooth trajectory across the current 2D level of the environment. The continuous control set of the robot yields a complex optimization problem in continuous variables. For computational reasons, the algorithm breaks the optimization into two phases. First an A* search is performed, which is then followed by a numerical optimization taking the result of the A* search as input.

We assume that the vicinity of the robot can be reasonably described by a 2D plane. We therefore trigger an update of these structures whenever the robot changes the level in the environment. Hence, it is sufficient for the local planner to carry out an A* search on the four-dimensional state space $\langle x, y, \theta, d \rangle$, where $\langle x, y, \theta \rangle$ defines the 2D position and orientation of the vehicle and $d = \{0, 1\}$ corresponds to forward or backward motion. The efficiency of A* depends on the heuristic for guiding the search. To this end, we consider two different heuristics. The first one models the non-holonomic nature of the car but ignores the current obstacle map. This heuristic, which can be pre-computed offline, ensures that the search approaches the goal with the right heading θ. As second heuristic, we consider the result of a standard 2D Dijkstra search on the occupancy map. This heuristic is dual to the first component since it ignores the non-holonomic constraints of the vehicle but considers the currently sensed obstacles. Both heuristics are admissible heuristics for A*. Hence, we combine both by defining our heuristic as the maximum over both components. The output of the first phase using the A* search yields a safe and kinematically-feasible trajectory. We can, however, only use a highly-discretized set of control actions in the search for efficiency reasons. This small set, in turn, may lead to sub-optimal paths.

The second phase of local planning improves the quality of the trajectory computed in the first phase via numerical optimization in continuous coordinates. We apply a conjugate-gradient (CG) descent algorithm to the coordinates of the vertices of the path produced by A* to quickly obtain a locally optimal solution. Our optimization uses a carefully constructed potential function defined over 2D curves (see [46] for more details) to produce a locally optimal trajectory that retains safety and kinematic feasibility. In practice, the first phase (A*) produces a solution that lies in the neighborhood of the global optimum, which means that our second phase (CG) then produces a solution that is not only locally but globally optimal.

An example trajectory generated by the local planner for a parking maneuver is shown in Figure 8.4. The red curve shows the output of A*, whereas the blue curve shows the final smooth trajectory produced by the nonlinear optimization given the A* solution as initial state for the optimization.

In summary, our path planning framework employs a global planner that operates on the topological graph $\mathscr{G}$ to produce a global path through the multi-level structure. The global planner then iteratively calls a local planner [46] to find a safe, feasible, and smooth trajectory through the current 2D level of the environment. Additionally, the global planner monitors the progress of the local planner and updates the targets appropriately.

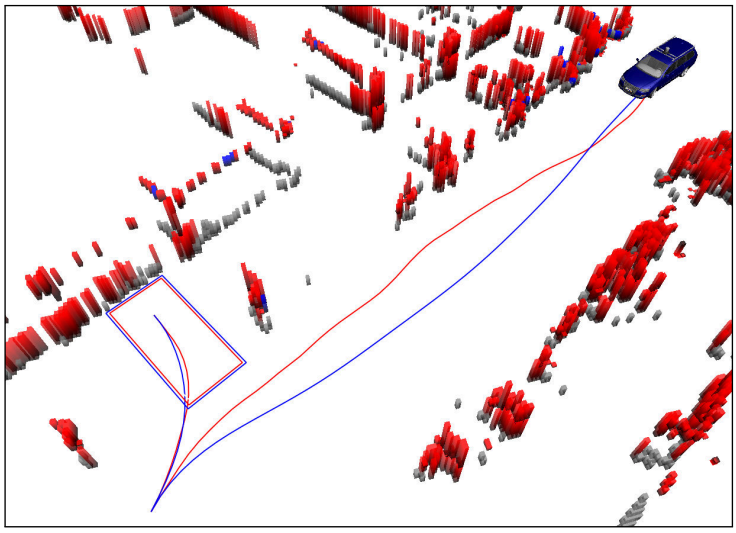


Figure 8.4: Example of a trajectory planned by the local planner. The red curve is the trajectory produced by phase one of our search (local-A*). The blue curve is the final smoothed trajectory produced by conjugate-gradient optimization.

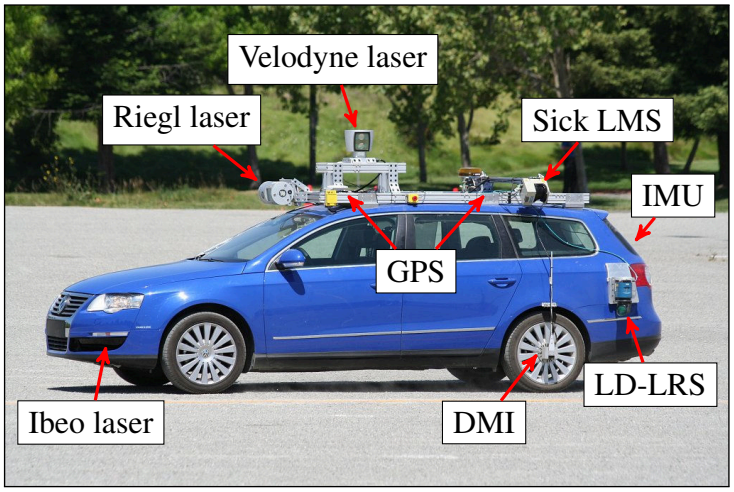


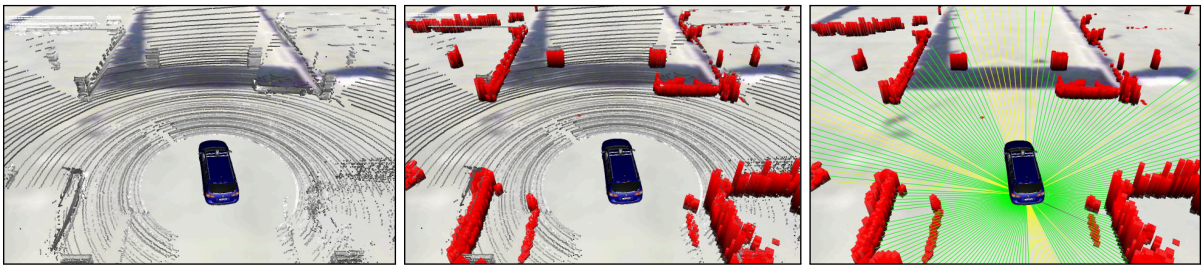
Figure 8.5: The car used for the experiment is equipped with five laser measurement systems and a multi-signal inertial navigation system that provides the odometry estimates.

8.4 Experiments

Our approach has been implemented and evaluated on a modified 2006 Volkswagen Passat Wagon (see Figure 8.5). The vehicle is equipped with multiple laser range finders (manufactured by IBEO, Riegl, Sick, and Velodyne), five BOSCH radars, two Intel quad core computer systems, and a custom drive-by-wire interface. Furthermore, a custom interface-board allows us to control the car.

For inertial navigation, an Applanix POS LV 420 system provides real-time integration of multiple dual-frequency GPS receivers, which includes a GPS Azimuth Heading measurement subsystem, a high-performance inertial measurement unit, wheel odometry via a distance measurement unit (DMI), and the Omnistar satellite-based Virtual Base Station service. The online position and orientation errors of this system were typically below 100 cm and 0.1 degrees, respectively, but only when GPS is available.

Although the car is equipped with multiple sensor systems, only data from the Velodyne



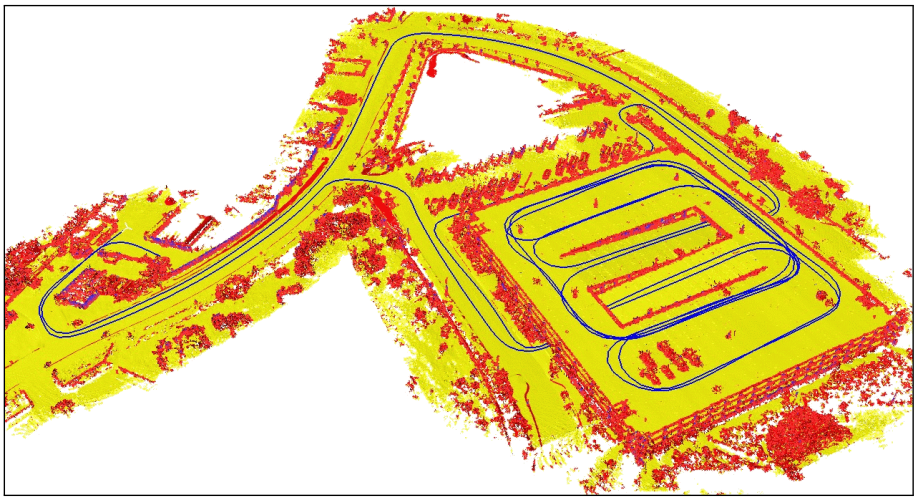
(a) (b) (c)

Figure 8.6: Necessary steps for the 2D map building. (a) First, the sensor measurements from the Velodyne laser are analyzed for obstacles. (b) The obstacle measurements are discretized in 15 cm grid cells (red cubes). (c) These cells are used to generate a virtual 2D scan for updating a two-dimensional map. Yellow beams indicate that no obstacle is measured in this direction and a fixed range is used for the update. The two-dimensional map is used for the low level parking planner.

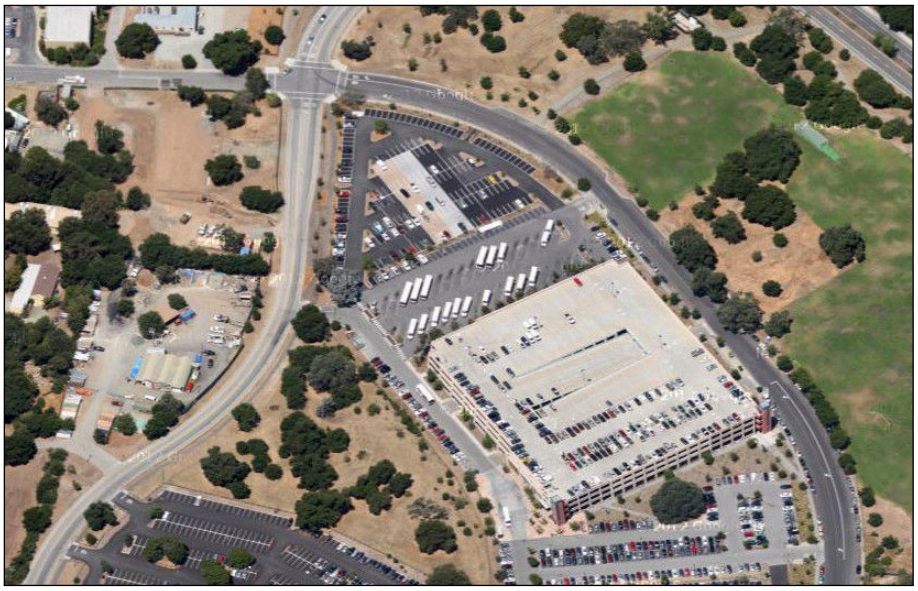
LIDAR and the two side mounted LD-LRS lasers were considered in our experiments. The Velodyne HDL-64E is the primary sensor for obstacle detection. Its spinning unit collects approximately one million data points per second. The system has a 26.8 degree vertical field of view and a 360 degree horizontal field of view. The spinning rate in our setup is 10 Hz. Thus, one full revolution includes approximately 100,000 points. The maximum range of the sensor is about 60 m. Due to the high mounting of the sensor and the shape of the car, an area close to the car is occluded. To compensate this blind spot, we use two additional LD-LRS laser scanners.

The software utilized in our experiments is based on the system used in the DARPA Urban Challenge. The architecture includes multiple modules for different tasks such as communication to the hardware, obstacle detection and 2D map generation. Montemerlo *et al.* [152] provide a full description of the system. Here, we focus on the modules relevant for the experiment. A low level controller executes commands to follow a trajectory, which is computed by the local planner described in Section 8.3. The velocity of the car is based on the curvature of the trajectory and the maximum speed was set to 10 km/h. Additionally an obstacle detection module analyzes the sensor data (see Figure 8.6a), discretizes the found obstacles in a fixed sized grid (see Figure 8.6b), and builds a 2D grid map. It generates a virtual 2D scan from the obstacle data (see Figure 8.6c) to update the map. The resulting map is then taken into account by the local planner to generate the trajectory. The localization described in Section 8.2 sends translational correction parameters to the other modules to determine the correct position of the vehicle. Finally, the global path planner computes a trajectory on the surface map to the goal point using A*. The global path is then divided into sub-goals. The program watches the progress of the car and updates the sub-goal to keep a fixed distance of approximately 20 m to the car. Hence, the global path planner leads the car along the computed trajectory and allows the local planner to avoid obstacles like parked cars that are not included in the map.

We carried out three different experiments to show the functionality of the system. First, we demonstrate that we can build a MLS-map from collected sensor data of the parking garage. Second, we illustrate that the described localization technique is essential for our experimental setup, and finally we show that based on our implementation, a car can autonomously drive and park in a multi-level parking structure.



(a) MLS map estimated by our approach



(b) aerial image of the parking garage

Figure 8.7: (a) The MLS map used for the experiment. The trajectory of the robot as it is estimated by the SLAM system is drawn in blue. (b) Aerial image of the parking structure (© Google).

8.4.1 Mapping

To obtain the data set, we steered the robot along a 7,050 m long trajectory through the parking garage and the surrounding environment. The trajectory contains multiple nested loops on different levels of the garage. The robot collected more than 15,000 3D point clouds by its Velodyne sensor. This data has been merged into 1,660 local MLS-maps used for scan-matching to estimate a model of the environment. The outcome of our SLAM algorithm along with an aerial image can be seen in Figure 8.7. The cell size of the generated map was set to 20 cm. The parking garage used for the experiments consists of four levels and covers an area of approximately 113 m by 100 m. Using a cell size of 20 cm, it requires around 120 MB of disk space for storing the map.

Despite the high accuracy of the inertial navigation system mounted in the vehicle, we found that scan-matching further improves the odometry estimate. In particular, the estimate of the navigation system exhibits a large drift in height, which is compensated by scan-matching.

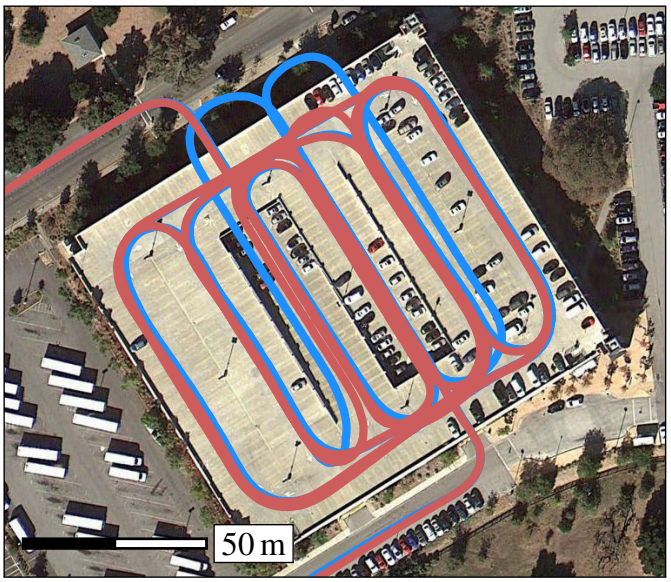


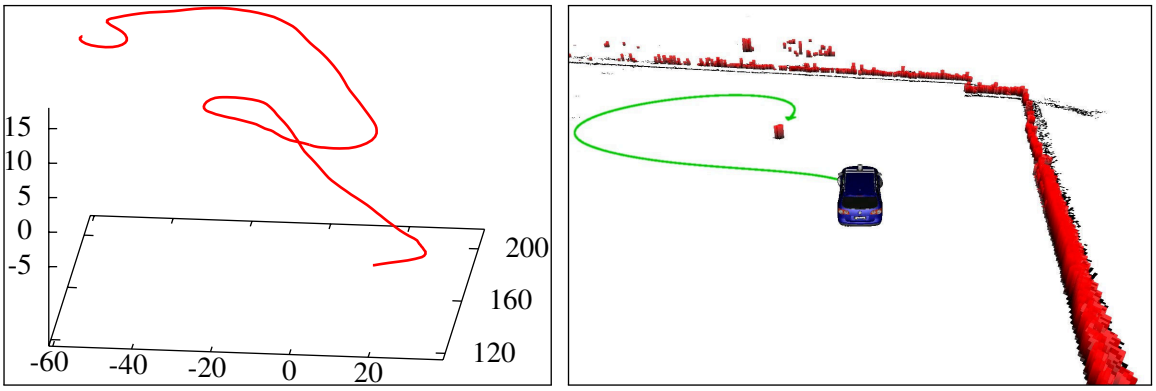
Figure 8.8: Bird's eye view of the parking garage. The blue trajectory shows the position as it is estimated by the inertial navigation system of the car, whereas the red trajectory is estimated using our localization approach (aerial image © Google).

8.4.2 Localization

The described localization algorithm was evaluated on several separate data sets that have been acquired at different points in time. Hence, the environment was subject to change, i.e., parked cars sensed during map building are no longer present or currently present obstacles were not observed while mapping the area. In all our experiments the particle filter was able to accurately localize the vehicle online. The algorithm performs pose correction with the update rate of the sensors using 1,000 particles, whereas the proposal distribution is updated at 200 Hz. Figure 8.8 depicts the outcome of the localization algorithm. Note that the trajectory as it is estimated by the inertial navigation system contains large errors while driving without GPS coverage. Actually, the estimated trajectory is outside the boundaries of the parking garage after the car reached the third level. This error remains until the car reaches the top level and again receives GPS fixes, which allow the navigation system to gradually reduce this error. In contrast to this, our localization algorithm based on the MLS-map and the range measurements is able to accurately localize the vehicle all the time. The accuracy of our localization is confirmed by the inertial navigation system whenever it receives valid GPS fixes as the trajectories overlay at that time.

8.4.3 Autonomous Driving

The following experiment is designed to show the abilities of our approach to autonomously drive in a multi-level parking garage. The task of the vehicle was to drive from the start position, which was close to the entrance, to a parking spot on the upper level of the parking garage. Figure 8.1 depicts an aerial image of the parking garage, in which the start and goal have been marked. The car autonomously traveled to the target location. The resulting 3D trajectory of the vehicle is depicted in Figure 8.9a. The trajectory has a total length of 375 m and it took 3:26 minutes to reach the goal with an average speed of 6.6 km/h (maximum speed 9.5 km/h). Figure 8.9b depicts the trajectory of the local planner to the goal. Videos documenting the experiments can be found on the Internet [58].



(a) trajectory of the robot (b) parking maneuver

Figure 8.9: Trajectory of the autonomous navigation inside the parking garage. (a) The complete trajectory from the start point in the first level to goal point in the fourth level. (b) The last part of the trajectory to the goal point behind an obstacle (lamp pole).

8.5 Related Work

The system described in this chapter addresses several aspects of autonomous navigation including localization, mapping, and path planning. In the following, we discuss the related work in each of these fields while focusing on our target application, namely autonomous navigation with an automobile.

Several authors have studied the problem of mobile robot localization in outdoor environments with range sensors or cameras in the past. For example, Adams *et al.* [1] extract predefined features from range scanners and apply a particle filter to localization. Davison and Kita [41] employ a Kalman filter for vision-based localization with point features on non-flat surfaces. Agrawal and Konolige [5] presented an approach to robot localization in outdoor terrains based on feature points that are tracked across frames in stereo images. Lingemann *et al.* [139] described a method for fast localization in in- and outdoor environments. Their system operates on raw data sets, which results in huge memory requirements. Operating on a compact 3D model, Hornung *et al.* [89] present a method that allows a humanoid robot to localize itself in complex environments including staircases and multiple levels. Kümmerle *et al.* [125] realize a localization algorithm for wheeled robots in urban environments. Both approaches employ a particle filter and they consider the sensor data from 2D range finders whose measurements are matched against a 3D model of the environment. Grzonka *et al.* [80] describe a full navigation system for a quadrator, in which the localization and mapping is based on 2D range data supported by an IMU.

Recently, Levinson and Thrun [134] presented an algorithm for mapping and localization in large urban environments. Their approach localizes the vehicle using 2D intensity images of the road surface which are obtained by the reflectivity measurements of a 3D range scanner mounted on the roof of the car. As stated by the authors, such a precise localization is a key enabling factor for navigating in narrow urban roads. In contrast to their method, our approach employs a 3D model of the environment and is therefore not restricted to roughly planar environments which only contain one driveable level. Baldwin and Newman [13] relax the dependency on a 3D range finder and suggest an approach solely operating on 2D range finders mounted with different inclination to accurately localize a vehicle given 3D priors.

Building maps of outdoor environments using range or vision data also gained interest in recent years. To reduce the memory requirements of outdoor terrain representations, several re-

searchers applied elevation maps [14, 87, 127, 171]. This representation only stores one height value per cell which represents the drivable surface but is not sufficient to represent vertical or overhanging objects. Therefore, Pfaff *et al.* [175] extended elevation maps to also deal with vertical and overhanging objects. To also address the issue of multiple levels in the environment, e.g., a bridge and the corresponding underpass, Triebel *et al.* [209] presented multi-level surface (MLS) maps. Another popular approach to modeling environments are grids [155] of volumetric 3D cells. Each cell encodes the probability that the cubic volume is occupied. To address the large memory requirements of a volumetric grid, data structures based on octrees [90] are a viable alternative as they combine cells having the same occupancy probability into large compounds. While an MLS map is not entirely probabilistic, as it does not encode free space, it yields a compact representation of the occupied volume. Furthermore, an MLS map provides sufficient information for a localization algorithm [125] and the underlying grid is advantageous when implementing a planning algorithm in a fast and efficient manner.

The local planner utilized in our approach builds on the existing work in discrete search in unknown environments (e.g., [59, 106, 156]) as well as kinematic forward search [103, 129, 177] and nonlinear optimization in the space of 2D curves [38]. Whereas strategies for specific dedicated parking maneuvers have been developed [169, 170] and nowadays are even available in off-the-shelf vehicles, these systems perform autonomous navigation only in a short range and are not able to plan complex navigation tasks through entire parking structures. The work probably most closely related to ours is the approach by Schanz *et al.* [186] who developed an autonomous parking system in a subterranean parking structure. Compared to our work described in this chapter, their system can only deal with given two-dimensional map representations and lacks the capability to detect obstacles in 3D. Additionally, their system cannot deal with multiple levels or drive on non-flat surfaces like ramps.

Based on the first Grand Challenges organized by DARPA, the robotics community gained interest in using a car as a robot. During the competition, several systems [38, 206, 214] showed the ability to autonomously drive over extensive periods of time through desert areas. Within the second challenge organized by DARPA, the Urban Challenge, the ability to navigate in dynamic urban environments with complex traffic scenarios while obeying traffic rules has been demonstrated successfully [152, 215]. These efforts resulted in fully autonomous demonstrations in urban traffic [161] beyond competition settings and have also led to commercial systems [71] which implement autonomous navigation with cars. These systems, however, heavily rely on the availability of GPS while navigating to estimate the position of the vehicle. As we have shown in the experiments, such a system is not able to localize the vehicle accurately in GPS-denied parts, such as the parking structure considered in our experiment.

8.6 Conclusions

In this chapter, we presented a novel approach for driving in complex GPS-denied multi-level structures with an autonomous car. We described the individual approaches for mapping, SLAM, localization, path-planning, and navigation. In particular, the techniques developed in this thesis are an important building block of the whole system. Without an accurate map of the parking structure, localization just based on GPS would fail and render the car unable to reliably reach the designated goal location.

Our approach has been implemented and evaluated with a real Volkswagen Passat Wagon in a large-scale parking garage. The experimental results demonstrate that our approach allows the robotic car to drive autonomously in such environments. The experiments furthermore illustrate that a localization algorithm is needed to operate indoors, also in the case in which the robot is

equipped with a highly accurate state-of-the-art combined inertial navigation system.

While the approach presented in this chapter operates on a vehicle driving on the road, we describe a system for a pedestrian assistant in the next chapter. The key components of both systems apply similar techniques, but they substantially differ in the specific implementation. The different sensor modalities and the envisioned applications impose challenges whose solutions demand distinct approaches, as we will see in the following chapter. For instance, we tackle the large-scale environment with special data structures for planning and localization.

Chapter 9

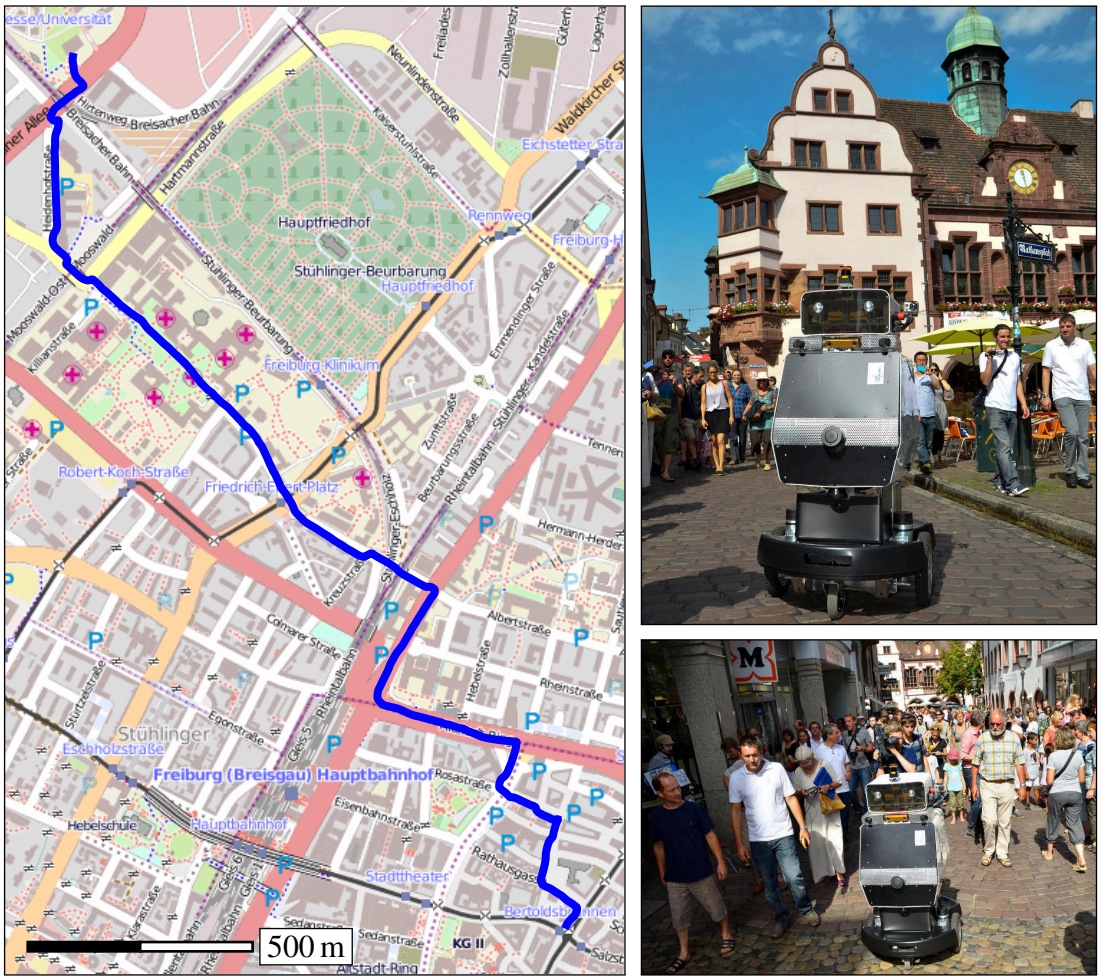
A Robotic Pedestrian Assistant

In this chapter, we present a navigation system for autonomous navigation with mobile robots in city environments that relies on the techniques presented in this thesis. Whereas we presented a navigation system for a robot driving on the road in the previous chapter, we will now describe our efforts for realizing a pedestrian-like robot. Urban areas pose numerous challenges to autonomous mobile robots as they are both large-scale and highly complex, and in addition to that dynamic. We describe different components including a SLAM system for dealing with huge maps of city centers, a planning approach for inferring feasible paths that take also into account the traversability and type of terrain, and an approach for accurate localization in dynamic environments. The navigation system has been implemented and tested in several large-scale field tests in which a robot managed to autonomously navigate from our University Campus over a 3.3 km long route to the city center of Freiburg.

• • • • • • • • • • •

In this part of the thesis, we up to here demonstrated how the techniques for mapping and state estimation which we developed in this work can serve as a building block of a navigation system for an autonomous car. In particular, the model of the environment, which was estimated by our SLAM approach, led to a self-contained system for planning and localization. In addition to this, we will present in this chapter how our techniques have been utilized and extended to enable a robot to navigate in densely populated city centers. We particularly focus on pedestrian-like navigation, i.e., driving on the sidewalk or in pedestrian areas instead of on the road. While traffic on the road follows certain rules, the flow of motion in pedestrian areas is less structured. Thus, navigating in densely populated pedestrian zones confronts the robot with a series of challenges, for example, the scenes may be highly complex and dynamic while at the same time GPS is unreliable. Only few systems, for example, the autonomous city explorer [15], have been designed for robot navigation in such populated urban environments.

In this chapter, we describe a navigation system that enables mobile robots to autonomously navigate through city-center scenes. The system utilizes the techniques presented in this thesis as core building blocks. In particular, the SLAM system for learning accurate maps employs HOG-Man presented in Chapter 3. Furthermore, we will extend the integration of prior information, which was discussed in Chapter 6, to robustly handle outliers in the prior measurements.



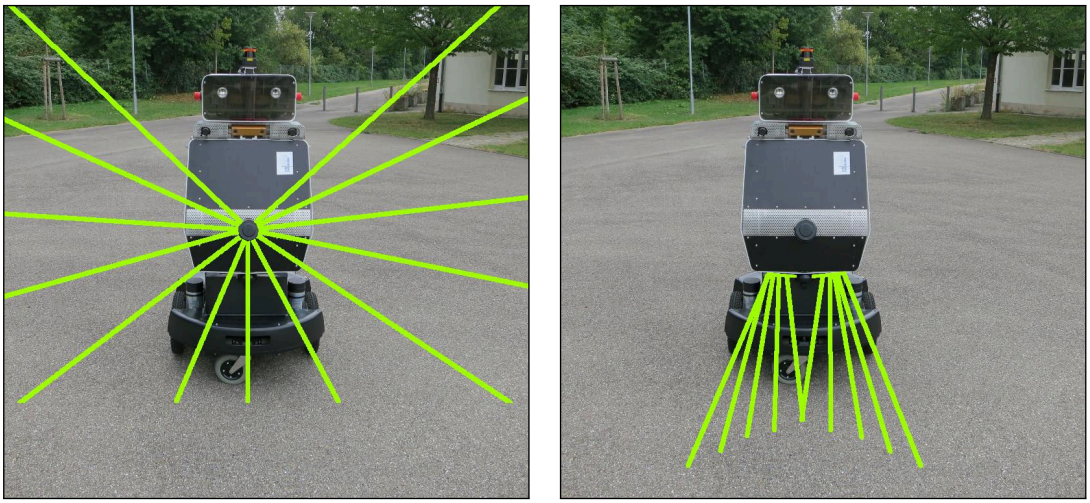
(a) the path taken by the robot (b) the robot navigating in downtown

Figure 9.1: Example trajectory traveled by our robot navigating in an urban environment including a pedestrian zone with a large number of people surrounding the robot. (a) Map data from OpenStreetMap (© OpenStreetMap contributors) along with an overlay of the trajectory taken by the robot drawn in blue. (b) The robot driving in the city center of Freiburg.

As the environment in which the robot has to navigate might be as large as a city, we furthermore suggest a memory-efficient data structure for such large-scale maps and a planning system which also exploits this data structure. The data structure is closely related to clusters in the hierarchy of HOG-Man, our approach for optimizing a pose-graph online. Other components of the navigation system include a variant of Monte Carlo localization and a terrain analysis that deals with vegetation, dynamic objects, and negative obstacles. In the following, we will describe how these components are integrated.

In addition to the description of the system, we will present results obtained during large-scale experiments. In particular, our efforts allowed the robot to travel from our university campus to the city center of Freiburg. This trial was publicly announced beforehand and attracted numerous journalists and curious people that followed the robot along its journey. Figure 9.1 depicts the trajectory of a length of around 3.3 km traveled by the robot along with exemplary pictures of the robot navigating in the busy city center of Freiburg. The navigation capabilities of the robot demonstrated during this event lead to positive press coverage, including reports in the main TV news broadcast.

Thus, the aim of this chapter is not only to describe the relevant components, but also to



(a) (b)

Figure 9.2: (a) One laser is mounted downwards to sense the surface in front of the robot to decide whether it is safe to navigate over a particular area. (b) A second horizontally mounted laser is combined with mirrors, which reflect a portion of its beams towards the ground. The data from those two lasers is used to find obstacles that are not visible in the horizontal range data.

further highlight the capabilities that can be achieved by building a system upon the approaches developed in this thesis. Furthermore, as an additional insight we try to motivate our design decisions, critical aspects, as well as limitations of the current setup.

The remainder of this chapter is organized as follows. After presenting the robot in the next section, we discuss the individual components of the system in Section 9.2, including the algorithms for SLAM, localization, obstacle detection, and path planning. Subsequently, we provide an evaluation of our system in Section 9.3 followed by a discussion about the limitations of the system in Section 9.4. In Section 9.5, we present the related work.

9.1 The Robot used for the Evaluation

The robot used to carry out the field experiments is a custom made platform developed within the EC-funded project EUROPA [55], which is an acronym for EUropean RObotic Pedestrian Assistant. The robot is equipped with a differential drive that allows it to move at a maximum velocity of 1 m/s. Using flexibly mounted caster wheels in the front and the back, the robot is able to climb steps up to a maximum height of approximately 3 cm. Whereas this is sufficient to negotiate a lowered pedestrian sidewalk, it has not been designed to go up and down normal curbs. Thus, the robot needs to avoid such larger steps. The footprint of the robot is 0.9 m $\times$ 0.75 m and the robot is around 1.6 m tall.

The main sensor input is provided by laser range finders. Two SICK LMS 151 are mounted horizontally in the front and in the back of the robot. The horizontal field of view of the laser mounted in the front is restricted to 180 °. The remaining beams are reflected by mirrors towards the ground surface in front of the robot. Additionally, another scanner is tilted 70 ° downwards to detect obstacles and to identify the terrain the robot drives upon. Figure 9.2 illustrates the setup of the non-horizontal laser beams. Additionally, a Hokuyo UTM-30LX is mounted on top of the head of the robot. Its measurements are used for mapping and localization, whereas the measurements of an XSens IMU are integrated to align the UTM horizontally with the ground surface by controlling a servo accordingly. The robot is furthermore equipped with a

Trimple GPS Pathfinder Pro to provide prior information about its position during mapping tasks. While the robot also has two stereo cameras on-board, their data is not used for the described navigation tasks.

9.2 System Overview

In order to autonomously navigate in an environment, our system requires to have a map of the area, which is estimated by the approaches described in this thesis. To depend on a map of the environment might seem like a huge drawback, but mapping can be done in a considerably small amount of time. For example, it took us around 3 hours to map a 7.4 km long trajectory by controlling the robot with a joystick. Furthermore, this only has to be done once as the main structures of an urban area do not tend to change quickly. Small modifications to the environment, like billboards or shelfs placed in front of shops, can be handled by our system in a robust manner. In the following, we describe how we obtain the map of the environment by means of a SLAM algorithm as well as the most important components of the autonomous navigation system, such as the algorithms for path planning, localization, and obstacle detection. The system enables our robot to operate in large-scale city centers. The entire navigation system described in this section runs on one standard quad core i7 laptop operating at 2.3 GHz.

9.2.1 Mapping

The key technique of our mapping algorithm is described in Chapter 2 and Chapter 3, namely the graph-based SLAM formulation for estimating the maximum-likelihood (ML) configuration. As we will extend this in the following, let us briefly recall the concept of this formulation. The vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)^\top$ describes the poses of the robot. Furthermore, the observation $\mathbf{z}_{ij}$ characterizes the motion of the robot between the time indices i and j with Gaussian noise given by the information matrix Ω_{ij}. Let $\mathbf{e}_{ij}(\mathbf{x})$ be an error function which computes the difference between the observation $\mathbf{z}_{ij}$ and the expected value given the current state of node i and node j. Additionally, let $\mathbf{e}_i(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ be an error function which relates the state of node i to its prior $\hat{\mathbf{x}}_i$ having the information matrix Ω_i.

As we have previously seen, we obtain the ML configuration of the trajectory taken by the robot including the prior information as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{ij \in \mathscr{G}} \mathbf{e}_{ij}(\mathbf{x})^\top \Omega_{ij} \mathbf{e}_{ij}(\mathbf{x}) + \sum_{i \in \mathscr{P}} \mathbf{e}_i(\mathbf{x}_i, \hat{\mathbf{x}}_i)^\top \Omega_i \mathbf{e}_i(\mathbf{x}_i, \hat{\mathbf{x}}_i), \tag{9.1}$$

where $\mathscr{G}$ and $\mathscr{P}$ are a set of constraints and priors, respectively.

On our robot, the high-end GPS sensor provides the set of priors $\mathscr{P}$. As GPS signals may be corrupted by multi-path effects and atmospheric distortions before reaching the receiver, we further extend the approach presented in Chapter 6 by an outlier rejection method to remove those measurements. Instead of directly solving Eq. 9.1, we consider a modified cost function for the prior measurements. To this end, we introduce $\tilde{F}(\mathbf{x})$, which is given as

$$\tilde{F}(\mathbf{x}) = \sum_{ij \in \mathscr{G}} \mathbf{e}_{ij}(\mathbf{x})^\top \Omega_{ij} \mathbf{e}_{ij}(\mathbf{x}) + \sum_{i \in \mathscr{P}} \rho\left(\mathbf{e}_i(\mathbf{x})^\top \Omega_i \mathbf{e}_i(\mathbf{x})\right), \tag{9.2}$$

where $\rho(\cdot)$ is a robust cost function. While other choices are possible, we consider the Pseudo Huber cost function [86] for the prior measurements, which is defined as

$$\rho(a) = 2\delta^2 \left(\sqrt{\frac{a}{\delta^2} + 1} - 1\right), \tag{9.3}$$

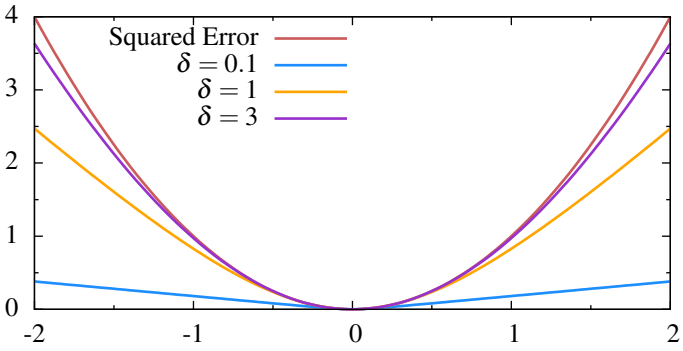
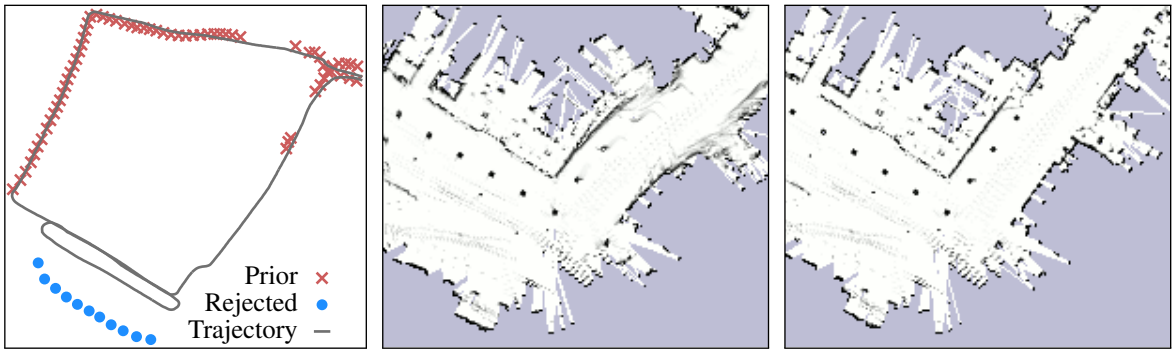


Figure 9.3: The Pseudo Huber cost function with different values for the parameter δ.



(a) rejected priors (b) map including all priors (c) map of our approach

Figure 9.4: Influence of outliers in the set of prior measurements. (a) Our method rejects prior measurements having a large error. (b) The map as it is estimated by taking into account all prior measurements. (c) Our method achieves a good estimate for the map by rejecting priors that are likely to be outliers.

where δ is a free parameter. The Pseudo Huber cost function approximates a, i.e., the squared error weighted by the information matrix for small values of a, and is approximately linear with slope δ for large a, see Figure 9.3. As we have seen in Section 2.3.1, each error function $\mathbf{e}_i$ contributes with an addend term H_i to the system matrix H and with $\mathbf{b}_i$ to the right hand side $\mathbf{b}$ of the sparse linear system. Consequently, we need to describe how the robustification affects those terms. Following the derivation given by Triggs *et al.* [210], we see that

$$H_i \approx J_i^\top \left(\rho' \Omega_i + 2\rho'' \left(\Omega_i \mathbf{e}_i \right) \left(\Omega_i \mathbf{e}_i \right)^\top \right) J \tag{9.4}$$

$$\mathbf{b}_i^\top = \rho' \mathbf{e}_i^\top \Omega_i J_i, \tag{9.5}$$

where ρ' and ρ'' are respectively the first and second derivative of $\rho(\cdot)$, J_i is the Jacobian of the error function, and $\mathbf{e}_i = \mathbf{e}_i(\mathbf{x})$ denotes the current error vector. Note that we consider the prior given by the GPS receiver instead of an aerial image as the robot is not equipped with a 3D range finder. Furthermore, a pure 2D range finder is more susceptible to errors in matching the observation to the aerial image as outlined in Chapter 6. Thus, we chose to consider the highly accurate GPS receiver which yields a sparse prior since the receiver is tuned to drop an ambiguous position estimate by neglecting satellites which are low above the horizon. This helps to remove incorrect estimates when the robot operates between tall buildings.

Applying the robustified error function weakens the influence of the outliers on the state. We, however, seek to completely remove the influence of measurements which have a high likelihood to be an outlier as indicated in Figure 9.4a. Thus, after the iterative optimization of Eq. 9.2 with Levenberg-Marquardt converged, we remove 2 % of the prior edges having

the largest residual. We repeat this process five times. Consequently, we keep approximately 90 % of the original prior information. Using this approach, some good GPS measurements might be rejected. We found in our practical experiments that the effect of outliers in the prior measurements may be severe (see Figure 9.4b and c), whereas losing some good measurements does not substantially worsen the map estimate. Including the prior information has several advantages. First, it improves the accuracy of the obtained maps (see Chapter 6). Second, if the robot extends its map, coordinates are easy to transform between different maps because the maps share a common global coordinate frame.

9.2.2 Map Data Structure

Obtaining a 2D map given the graph-based SLAM solution and the laser data is typically done in a straightforward manner, for example, by computing an occupancy grid [205]. Storing one monolithic occupancy grid for a large-scale environment, however, leads to a large memory footprint. For example, a 2 km by 2 km area at a resolution of 0.1 m and 4 bytes per cell requires around 1.5 GB of main memory. Instead of computing one large map, we consider the information stored in the hierarchy of pose-graphs to render maps locally and close to the position of the robot. A similar approach was recently described by Konolige *et al.* [112].

We generate the local map as follows. We apply Dijkstra's algorithm to compute the distance between the nodes in the graph. This allows us to only consider observations that have been obtained by the robot in the local neighborhood of its current location. We compute the set of nodes $\mathcal{V}_{map}$ to be used to build the local map as

$$\mathcal{V}_{map} = \{\mathbf{x}_i \in \mathbf{x} \mid \text{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base}) < r\}, \tag{9.6}$$

where $\mathbf{x}_{base}$ determines the reference frame of the map, $\text{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base})$ returns the distance between the two nodes according to Dijkstra's algorithm, and r is the maximum allowed distance for a node to be used in the mapping process. The reader may notice that the set $\mathcal{V}_{map}$ corresponds to one cluster in the hierarchy built by HOG-Man. Thus, we can directly exploit the clusters generated during optimization for rendering each local map. Furthermore, the arrangement of the local maps in space is given by optimizing the second layer in the hierarchy constructed by HOG-Man. As hard-disk space is rather cheap and its usage does not affect the performance of other processes, we store each local map on the disk after the first access to it by the system.

The localization and path-planning algorithms described in the following sections all operate on these local maps. The map is expressed in the local frame of $\mathbf{x}_{base}$ and we currently use a local map of 40 m $\times$ 40 m. Typically, each local map is locally consistent [91], even if the positions of the maps are not globally consistent. Hence, in contrast to a monolithic map we do not have to update the local maps if the optimization moves the origins of the maps in space.

9.2.3 Localization

To estimate the pose $\mathbf{x}$ of the robot given a map, we employ Monte Carlo localization [42] with a standard motion model. We consider a likelihood field [205] for evaluating the sensor model, which implicitly deals with obstacles not present in the map as it does not model the physical process underlying the range measurements. In contrast to most existing localization approaches, our system does not operate on a single grid map to estimate the pose of the robot. Given our graph-based structure we, accordingly, need to determine a vertex $\mathbf{x}_{base}$ whose map should be taken into account for evaluating the likelihood of a sensor reading. We determine the

base node $\mathbf{x}_{base}$ as the pose-graph vertex that minimizes the distance to $\mathbf{x}$ and furthermore guarantees that the current location of the robot was observed in the map. This visibility constraint is important to maximize the overlap between the map and the current observation. Without this constraint, the closest vertex might be outside a building while the robot is actually inside of it.

9.2.4 Traversability Analysis

For successful navigation, the robot needs to reliably avoid obstacles. Such obstacles can be divided into two categories, positive and negative obstacles. Whereas the first ones are objects sticking out of the ground plane by more than a threshold, the latter ones are sinkings with a depth above the threshold, such as a ditch or a street drain. For our robot the aforementioned threshold is just above 3 cm. In addition to describing our method for identifying such obstacles, we also present a technique which allows the robot to avoid other potentially hazardous surfaces, such as lawn, where the robot could easily get stuck due to its small caster wheel. Identifying lawn is not possible by just considering the range data as the vegetation appears to be smooth and drivable [218]. Here, we will present a technique exploiting the intensity data returned in addition to the range by a laser range finder.

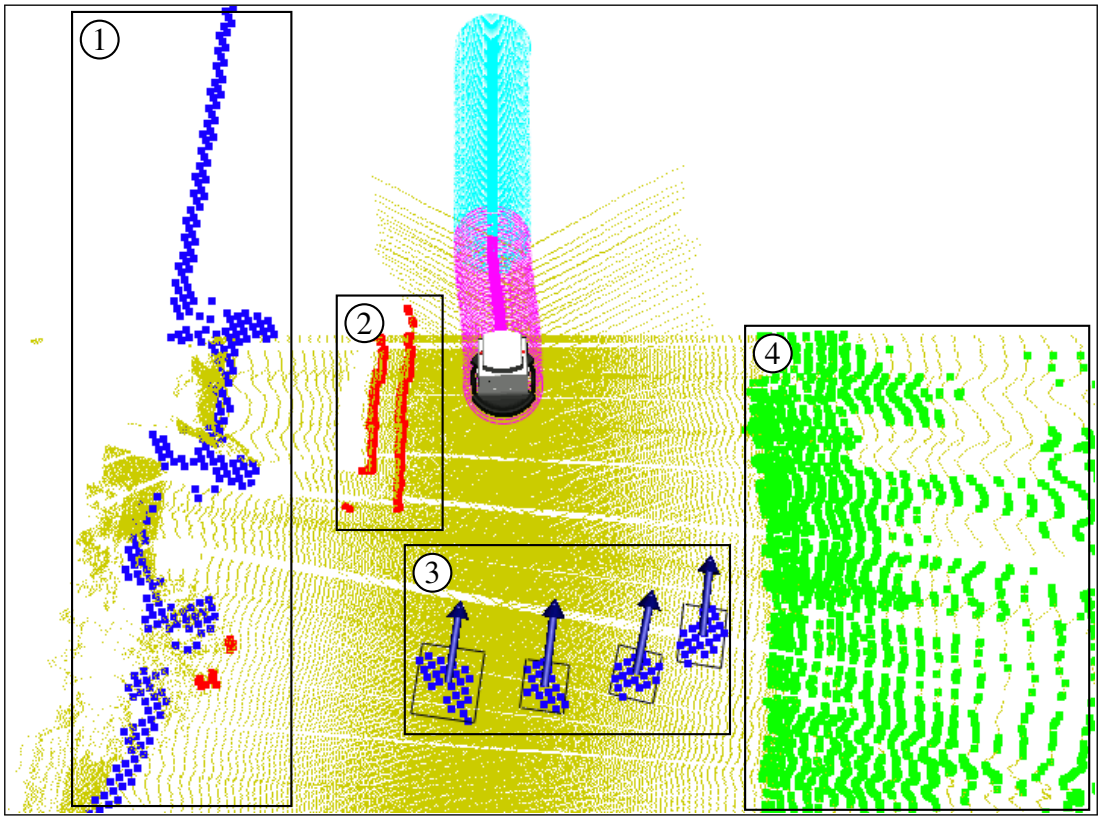
Obstacle Detection

As we are not able to specifically design the environment for our robot, not all obstacles are visible in the horizontally mounted range finders. Hence, we need to rely on the additional lasers mounted on the platform, see Figure 9.2. To this end, we analyze the scans obtained from the ground surface. Fitting a line to the range measurements allows us to detect line segments with a substantial variation in height or a slope above a threshold. Such segments are tagged as an obstacle. This is motivated by the fact that a flat ground plane in contrast would result in a straight line in the range data. Operating purely on single scans leads to an efficient detection. Furthermore, there is no need for accumulating range data into a point cloud, which might easily create fake obstacles due to the noise in the position estimates.

The detection of variations in the range data described above, however, causes false positives on manhole covers and gutters as their fine structure leads to variations in the range measurements, which in turn produce the wrong classification. To this end, an elevation map [87] is built by the robot, which allows us to estimate the size of the negative obstacle. Only negative obstacles with a size is above a threshold are forwarded to the planner.

As a last step, we test for each obstacle identified by our method whether there is also an obstacle present in the horizontal range data. Obstacles which have not been detected beforehand are forwarded to the planner framework to avoid them. This is required as the robot due to its sensor setup cannot easily re-observe the obstacles which are only visible in the non-horizontal beams. In contrast, obstacles visible in the horizontal range data can typically be re-observed. Thus, our filtering method prevents the creation of additional positive obstacles corresponding to dynamic objects if they are already observed in the horizontal measurements.

In addition to the detection of negative and positive obstacles, our robot tracks the motion of dynamic obstacles in its vicinity. To this end, we build a history of ten local grid maps. Each map accumulates the measurements of around 100 ms in our current implementation. We identify the dynamic aspects in the environment by determining the difference between the current map and the oldest map stored in the history. Obstacles that have previously not been occluded and that are only visible in the current map are potentially dynamic. These map elements are grouped into clusters and tracked over time by a nearest neighbor data association.



(a) detected obstacles and raw range data

(b) image of the area shown in (a)

Figure 9.5: (a) Visualization of the different kinds of detected obstacles. Blue points mark obstacles that are visible in the horizontal 2D laser scanners (areas 1 and 3). Red points mark 3D obstacles that are visible in the downwards facing laser beams but not in the 2D laser beams (mainly area 2). The black boxes with the arrows mark detected dynamic obstacles (area 3). Green points mark the detected vegetation/grass (area 4). The yellow dots visualize the accumulated point cloud from the laser measurements. The image additionally depicts the robot and its planned trajectory. (b) Image of the environment with a lawn on the right and a building with a two-step staircase on the left.

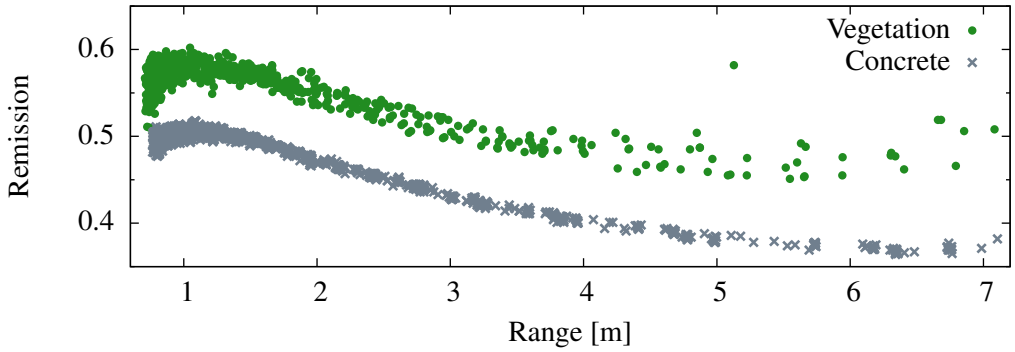


Figure 9.6: Range and remission data collected by the robot observing either a concrete surface or vegetation.

Tracking the objects yields a velocity estimate and a direction in which the object moves as can be seen in Figure 9.5a. This information allows the planner to react to pedestrians in an appropriate manner.

Vegetation Detection

Flat vegetation, such as lawn, cannot be reliably recognized in range measurements. Since lawn should be avoided by the robot, we apply an extension of an approach developed in joint work [218]. This approach considers the reflected intensity which is provided by a laser scanner along with the range. The classification exploits the fact that living plants exhibit a strong reflectivity of near infrared light, which is approximately the wavelength of the laser light emitted by a laser range finder.

In contrast to Wurm *et al.* [218], we detect vegetation with a fixed downward looking laser instead of a tilting laser. This results in an easier classification problem as the range of a beam hitting the presumably flat ground surface correlates with the incidence angle. Figure 9.6 illustrates the data obtained with our platform. As can be seen from the image, the two classes can be separated by a nonlinear function. We chose to fit a function to the lower boundary of the vegetation measurements. This partitioning function allows us to identify measurements which are likely to be vegetation with high efficiency. The resulting classification accuracy is slightly worse compared to the original approach but faster. An example, for the detection can be seen in Figure 9.5a. Furthermore, by accumulating the results in cells of a 2D grid and deciding the label for each cell by a majority vote of the observations falling into the cell, we are able to robustly detect flat vegetation in the environment as can be seen in Figure 9.7.

Non-Smooth Ground Surfaces

The terrain analysis is able to identify the drivable parts and objects that have to be avoided. The resolution of the laser range finder, however, is not sufficient to detect fine bumps in the ground surface. For example, cobble stone pavement is in general a drivable surface albeit its surface is locally rough. This roughness causes vibrations in the robot while driving upon the pavement. Therefore, we exploit the measurements of the IMU, which is able to sense the vibration. In situations where the vibrations exceed a threshold, we gradually reduce the maximum velocity for the platform. The reduction of the velocity is proportional to the sensed vibration. Since the laser measurements are not able to observe the fine bumps causing the vibrations, the robot has no means to identify whether the surface permits driving fast again without inducing vibrations.

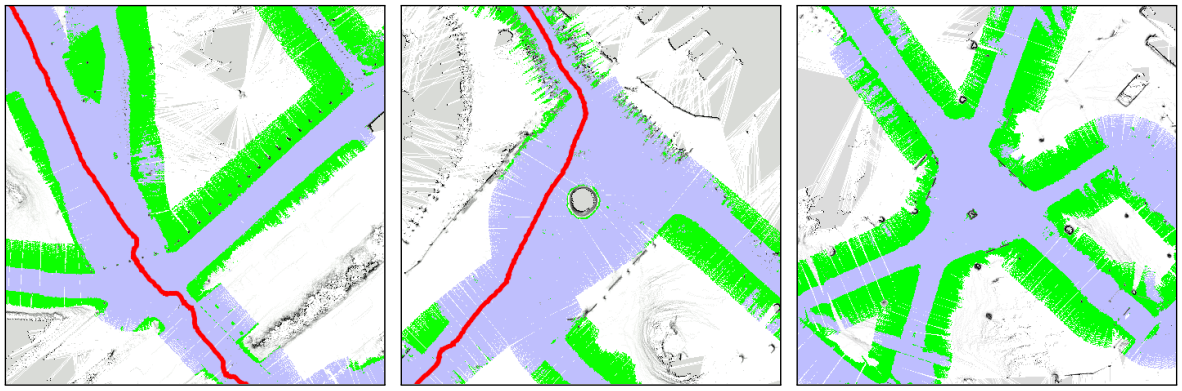


Figure 9.7: Examples for the vegetation detection. The trajectory drawn in red indicates an example path taken by the robot during one of the field tests.

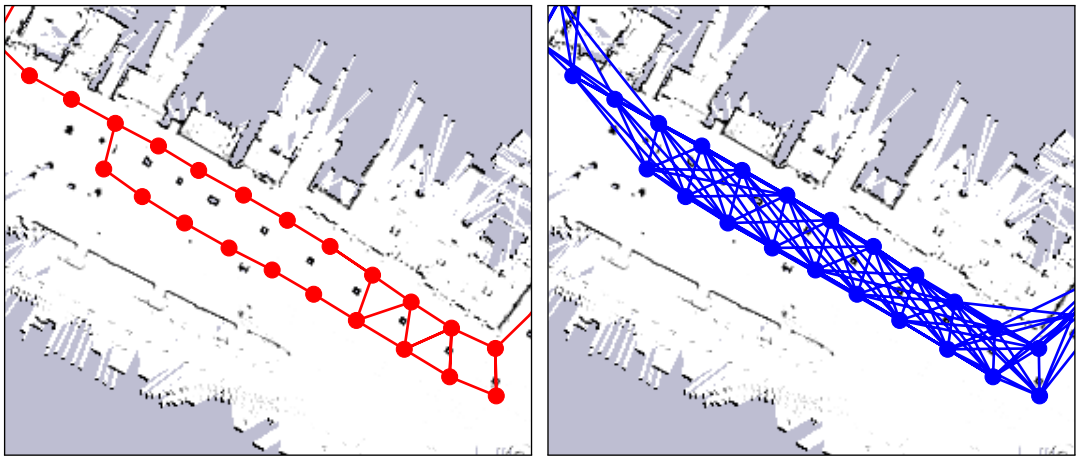
Hence, after a short delay we greedily increase the maximum velocity again up to the maximum given by the current vibrations.

9.2.5 Path Planner

Our planner considers different levels of abstraction to compute a feasible path for the robot towards a goal location. The architecture consists of three levels. On the highest level, only the topology of the environment is considered, i.e., the graph connecting local maps. The intermediate level employs Dijkstra's algorithm on the local maps to calculate way-points which serve as input for the low-level planner developed by Rufli *et al.* [181]. This low-level planner actually computes the velocity commands sent to the robot. Note that by using this hierarchy, we lose the optimality of the computed paths. Konolige *et al.* [112], who developed a planning framework similar to ours in parallel, report that the resulting paths are only approximately 10 % longer, whereas the time needed to obtain them can actually be several orders of magnitude shorter.

Given the pose estimates of our SLAM module, our planner constructs a topology $\mathscr{T}$ represented by a graph. This graph is constructed as follows: Each node $\mathbf{x}_i$ of the graph is labeled with its absolute coordinates in the world. Furthermore, each node comes with its local traversability map describing the drivable environment in the neighborhood of $\mathbf{x}_i$ which serves as the background information for the planner. Additionally, each cell in the map encodes the cost of driving from $\mathbf{x}_i$ to the cell. This can be pre-computed efficiently by a single execution of Dijkstra's algorithm starting from $\mathbf{x}_i$. We refer to this as the reachability information of the map.

Two nodes are connected by an edge if there is a path from one node to the other given the information stored in their local maps. The edge is labeled with the cost for traversing the path which is determined by planning on the local maps. If such a path cannot be found, we assign a cost of infinity to this edge. Otherwise, we assign to the edge the cost returned by the intermediate-level planner, which is typically proportional to the length of the path. Yet, in contrast to the straight-line distance, the cost better reflects the local characteristics of the environment. By this procedure, which is carried out once as a pre-processing step, the planner will consider the real costs for the robot to traverse the edge instead of only considering the Euclidean distance. Note that the set of edges contained in the topology graph $\mathscr{T}$ in general differs from the set of constraints $\mathscr{G}$ generated by the SLAM module. The topology graph exhibits a denser connectivity as can be seen in Figure 9.8.



(a) pose-graph (b) topology graph generated by the planner

Figure 9.8: (a) Partial view of the pose-graph with its constraints used for estimating the poses. (b) The same view of the topology graph generated by the planner shows that this graph typically features a denser connectivity.

While driving autonomously, the robot may encounter unforeseen obstacles, e.g., a passage might be blocked by a construction site or parked cars. Our planner handles such situations by identifying the edges in the topology which are not traversable in the current situation. Those edges are temporarily marked with infinite costs, which allows the hierarchical planner framework to determine another path to the goal location.

Planning a path from the current location of the robot towards a desired goal location works as follows. First, we need to identify the nodes or maps in $\mathscr{T}$ which belong to the current position of the robot and the goal. To this end, we refer to the reachability information of the maps. We select the maps with the shortest path from the center of the map to the robot and the goal, respectively. Given the robot node and the goal node, the high-level planner carries out an A* search on $\mathscr{T}$. Since the cost of traversing an edge corresponds to the real cost of the robot to traverse the edge, this search provides a fast approximation of an A* on a complete grid map of the environment but is orders of magnitude faster. The result is a list of way-points towards the goal. Following this list too closely may, however, lead to sub-optimal paths. Hence, we perform Dijkstra's algorithm in the local map starting from the current location of the robot and select as intermediate goal for the low-level planner the farthest way-point that is still reachable. Note that the local map containing the current position of the robot is augmented online with the static obstacles found by the obstacle detection.

9.3 Experiments

In this section, we describe a set of experiments in which we evaluated the system described in this chapter. The experiments show what can be achieved by a navigation system that builds upon the techniques developed in this thesis.

9.3.1 Mapping and Navigation

We obtained a map of the environment by steering the robot along a 7,405 m long trajectory. The map contains the area between the Faculty of Engineering of the University of Freiburg and the city center of Freiburg. Figure 9.9 depicts the pose-graph obtained by our SLAM approach

Figure 9.9: The pose-graph of the map used for the experiments on top of an aerial image. The trajectory as it is estimated by our approach aligns well with the aerial image (aerial image © Google).

as an overlay on top of an aerial image. The graph consists of 14,549 poses of the robot and 16,362 constraints.

Based on this map, we further evaluated the capabilities of the navigation approach. In particular, we carried out six navigation experiments in which the robot navigated from our campus to the city center and back. The robot traveled a distance of approximately 20 km. While navigating, we had to intervene only three times. One intervention was required due to the localization failure described in more detail below. Additionally, the robot once got stuck in front of a little bump and one further time was manually stopped by us because of an obstacle that we believed not being perceivable by the robot. An analysis of the collected data revealed that the obstacle in fact was detected by the robot and probably would have been avoided. Furthermore, the wireless emergency stop button was pressed unintentionally once causing a safety stop.

In addition to the field tests, which have been carried out during the development of the whole system, a final experiment was publicly announced. The goal of this event was to demonstrate the capabilities of a state-of-the-art navigation system to the general public and the press. A particular goal of this event was to reveal whether a robot utilizing our approach is able to robustly navigate through a densely populated environment. The event itself attracted journalists from both TV and newspapers and led to a nationwide and international coverage in top-media. Video material documenting the experiments can be found on the homepage of the project [55].

9.3.2 Localization

The localization in the map estimated by our SLAM technique is one of the key components of our system. In our setting the measurements obtained by the robot are heavily affected by pedestrians, cyclists, and cars on the street moving through the field of view of the robot. Furthermore, the environment itself may change. For example, cars may be parked at slightly

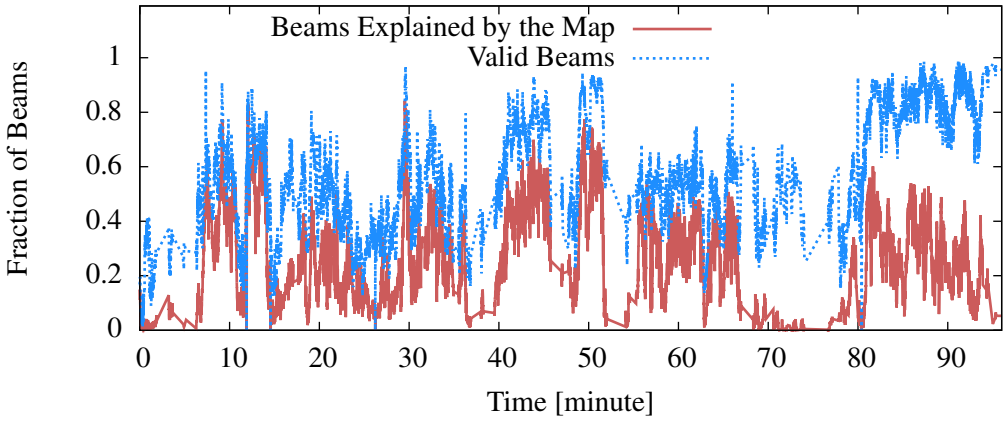


Figure 9.10: This plot shows the fraction of valid beams returned by the range scanner and the fraction of beams that can be explained by the map of the environment. The robot entered a crowded area in the city center after around 78 minutes. In this period, the localization algorithm can only consider approximately 50 % of the valid readings for localization.

different locations. Thus, we analyzed the impact of partial occlusion in the range data caused by objects which are not modeled in the map.

Given the position estimate of the robot and the map, we determined how many of the valid beams are explained by the map. Here, a valid beam is a measurement which is shorter than the maximum sensible range of the laser and we assume that a beam matches with the map if its distance to the closest element in the map is less than 0.2 m. Figure 9.10 illustrates the fraction of beams matching with the map and the fraction of valid beams. The number of valid beams correlates with the kind of space the robot is in. For instance, in an open region only a small amount of beams is reflected by obstacles, which means that only a small subset of the beams provide information for localizing the robot. Ideally, the difference between the valid beams and the beams matching with the map should be small. A substantial difference indicates that beams hit unexplained objects, such as pedestrians blocking the view of the robot. For example, after 78 minutes the robot navigated through a highly crowded area. This leads to a large fraction of measurements that cannot be explained by the map.

In the experiment depicted in Figure 9.10 a localization error occurred after around 76 minutes. As we can see in the figure, between minutes 70 and 76 the robot traveled in an area where only a small amount of features is available to the localization algorithm for determining the pose of the robot. The robot traveled around 200 m in this time period. The presence of a small amount of features and being surrounded by pedestrians for an extended time period, led to an error in the position estimate of around 2 m. This caused problems in negotiating a sidewalk after crossing a street and we had to re-localize the robot.

In other instances, sharing the same characteristics, for example, around minute 37 and around minute 52 the robot drove substantially shorter distances of 100 m and 50 m with a small amount of meaningful sensations. In both situations, the system was able to overcome the problem because it receives relevant information early enough again.

One of the large experiments was carried out during the night at which typically a substantially smaller number of pedestrians is around such that the measurements are less affected by occlusions. In turn, most of the valid beams match well to the map. In this experiment, the robot successfully reached its goal location without any problems traveling along a slightly different path of 3.5 km length. A visual inspection of the localization result revealed that the position of the robot was correctly estimated at all times.

9.4 Discussion

The experimental evaluation indicated several bases of decision-making for the arrangement of the sensors and the algorithmic principle of the perception processes. For example, we realized that crossing a road or interacting with cars poses a difficult perception problem. Such that enormously long-range sensors, such as radar, for sensing the vehicle approaching the robot, are required. Additionally, in certain traffic scenarios it is not decidable by just considering range data whether the robot actually has right of way for crossing a green light at a pedestrian light. Consider a police car which is on duty. In such a scenario, an elaborated vision system, which recognizes the blue light driving, or audio sensation for identifying the siren might be required to recognize a car driving with an exemption. We therefore solved this hard problem by letting the robot stop and ask for permission whenever it wants to cross a street. In our current system, these safety-relevant areas are marked manually in the map.

As mentioned above, the navigation system described in this chapter has been implemented on the robot characterized in Section 9.1. The design of the robot, obviously, affected the development of the software components, which were needed to achieve the envisioned task. For example, the obstacle detection had to cope with the sensor setup. Here, a 3D range sensor would allow us to apply different methodologies than the ones considered so far. In particular, such a device would enable the robot to re-observe obstacles that are currently not visible in the horizontal range data. On the other hand, the design of the robot also facilitated certain components. The almost circular shape of the robot, for instance, allows us to restrict the collision tests to two dimensions, see the circles in cyan and magenta in Figure 9.5, which can be checked in a fast and efficient manner for collisions with the perceived obstacles.

We furthermore realized that other aspects are pretty challenging. For example, curly leafs on the ground look similar to little rocks in the range data. Whereas the robot can easily drive over leafs, rocks can actually have a substantial effect on the platform itself. Additionally, pets or other animals like pigeons or ducks need to be modeled appropriately to effectively navigate in their vicinity. Here, the perception capabilities of the robot would need to be improved to robustly handle the different obstacle categories.

9.5 Related Work

The problem of autonomous navigation in populated areas has been studied intensively in the past. One of the pioneering systems were the robots RHINO [25] and Minerva [204] which operated as interactive mobile tour-guides in crowded museums. An extension of this tour-guide concept to interactive multi-robot systems was the RoboX system developed by Siegwart *et al.* [191] for the Expo'02 Swiss National Exhibition. Gross *et al.* [79] installed a robot as a shopping assistant that provided wayfinding assistance in home improvement stores. Although these systems were able to robustly navigate in heavily crowded environments, they were restricted to two-dimensional representations of the environment and assumed that the robots operated in a relatively confined planar area.

Relatively few robotic systems have been developed for autonomous navigation in city centers. The concept closest to the one described in this chapter probably is the one of the Munich City Explorer developed by Bauer *et al.* [15]. In contrast to our system, which operates autonomously and does not heavily rely on human intervention, the city exploration system depends on interaction with humans to get the direction where to move next. The city explorer only builds local maps and does not autonomously plan a path from its current position to the overall goal location. A further related approach has been developed in the context of the

URUS project [185], which considered urban navigation but focused more on networking and collaborative actions as well as the integration with surveillance cameras and portable devices.

Also, the problem of autonomous navigation with robotic cars has been studied intensively. For example, there has been the DARPA Grand Challenge during which autonomous vehicles showed the ability to navigate successfully over large distances through desert areas [38, 206, 214]. During the DARPA urban challenge, several car systems have been presented that are able to autonomously navigate through dynamic city street networks with complex car traffic scenarios and under consideration of road traffic navigation rules [152, 215]. Recently, commercial self-driving cars [71] have been developed and legalized to perform autonomous navigation with cars. In contrast to these methods, which focused on car navigation, the system described in this chapter has been developed to enable mobile robots to perform pedestrian-like autonomous navigation in urban environments with many types of dynamic objects like pedestrians, cyclists, or pets.

A long-term experiment about the robustness of an indoor navigation system was recently presented by Marder-Eppstein *et al.* [149]. Here, an accurate and efficient obstacle detection operating on the data obtained by tilting a laser range finder has been realized. In contrast to this system, our approach has a component for tracking moving obstacles to explicitly deal with dynamic objects in highly populated environments and also includes a terrain analysis component that is able to deal with a larger variety of terrain.

9.6 Conclusions

In this chapter, we presented a navigation system that enables a mobile robot to autonomously navigate through city centers. To accomplish this task, our navigation system uses an extended SLAM routine that deals with the outliers generated by the partially GPS-denied environments, a localization approach that utilizes a special data structure for large-scale maps, dedicated terrain analysis methods also for dealing with negative obstacles, and a trajectory planning system that incorporates dynamic objects.

The system has been implemented and demonstrated in a large-scale field test, during which the robot autonomously navigated over a path of more than three kilometers through the city center of Freiburg thereby negotiating with several potential hazards. These field tests demonstrate the capabilities that can be achieved by the approaches presented in this thesis in challenging settings. In particular, the publicly carried out demonstration attracted curious people and allowed them to get hand-on experiences with robotic technology. Furthermore, the event triggered positive news coverage across Europe, including the main German news programs.

Conclusions and Outlook

Chapter 10

Conclusions

A model of the environment is essential for a series of applications, we therefore focused on developing approaches to learn a map for navigation tasks with a mobile robot. Particularly, our approaches have to cope with a considerable amount of noise and uncertainty. In this thesis, we presented a collection of techniques that allow a mobile robot to perform state estimation in real-world settings. The state in these settings is high-dimensional as it in addition to the positions of the robot also includes the model of the environment.

Based on a map, it is possible to implement a variety of applications, such as cleaning, surveillance, or fetch and delivery. Hence, we explained the probabilistic formulation that underlies the SLAM problem. This led to a least squares problem, for which we introduced a general framework. Our approach allows us to solve this problem in a fast and accurate manner. Furthermore, it is applicable to other related problems, for instance, Bundle Adjustment, calibration, or model fitting. In our experiments, we showed that our general framework achieves a similar performance as domain specific implementations and that it outperforms many state-of-the-art approaches.

As SLAM is an online problem, we introduced a novel approach which enables a robot to efficiently compute the maximum likelihood estimate for the current state. The intermediate solutions in such an online setting are, for instance, required to plan the next action or to exploit this information for data association. Our algorithm is based on a hierarchy of factor graphs and can be interpreted as a divide-and-conquer approach. In addition to the more efficient computation, we also showed that the hierarchy leads to better convergence properties. Our approach is able to determine the correct solution where other techniques fail.

We furthermore proposed a metric which enables us to objectively compare the solutions of different SLAM algorithms. Instead of performing the comparison in a global reference frame, our metric only considers relative motions. This allows us to address shortcomings of metrics based on a global reference frame. We applied our metric to a collection of publicly available and widely used data sets. The evaluation shows that the graph-based formulation, which is the basis of our approach, outperforms other state-of-the-art methods in terms of accuracy.

While learning a map of an environment the robot greatly benefits from fusing the information of different sources, for example, the wheel odometry and the measurements of a laser range finder. Thus, we developed a technique which estimates the calibration parameters of the robot while performing SLAM. We extended our least squares estimator to include the corresponding parameters. In contrast to existing approaches, our method allows the robot to dynamically react to changes, such as the variation of the wheel diameters introduced by carrying a load.

Subsequently, we focused on improving the accuracy of the map. Most existing solutions to SLAM start from scratch. In contrast, our method allows us to include prior information

inferred from an aerial image. To this end, the robot matches sensor measurements to the aerial image. This matching process yields a set of priors about the location of the robot and we add this set as constraints to the estimation process. As we have shown, this leads to a better map estimate. We furthermore presented a technique that appropriately models the noise in the range measurements. In contrast to obtaining the map by integrating the individual measurements into a common representation by applying a mapping with known poses technique, our approach constructs a joint optimization problem that includes both the position of the robot and the range measurements. This allows us to determine the maximum likelihood estimate on a fine-grained level. In addition to improving the accuracy of the models, which is an achievement on its own, we demonstrated that a more accurate map also greatly improves the localization performance.

Inspired by the results of our state estimation techniques, we developed two navigation systems for mobile robots. Whereas the first one operates on a car, the second one implements a robotic pedestrian assistant. Both systems demonstrate real-world applications that build upon the techniques developed in this thesis. Since even highly accurate inertial navigation systems cannot estimate the position accurately in GPS-denied parts of the environment, we proposed to localize the robot within a 3D map. Furthermore, the map can be considered for planning an optimal path to a desired location. This allows the robotic car that uses our technology to park autonomously in a complex multi-level parking garage. Furthermore, for realizing a robotic pedestrian assistant, which should operate in densely populated city centers, we had to address several aspects. For example, we developed an efficient planning framework, which allows us to compute a path towards the goal by taking into account the data-structures of our hierarchical optimization approach. Additionally, we extended the inclusion of prior information to robustly deal with outliers. The system managed to autonomously navigate over extended time-periods and required only minor human assistance.

In summary, our approaches enable a mobile robot to answer the following questions:

- How can a mobile robot efficiently, accurately, and robustly estimate a model of the environment, even in online settings?
- How can a robot account for the effects of its mission on its calibration parameters, such as an additional load compressing the wheels of the robot?
- How can a robot exploit publicly available maps to improve its own estimate?
- How can a robot a model the noise in the range data and the remaining uncertainty in the position estimates?
- How can a robot navigate in complex and dynamic urban environments?

All techniques presented in this thesis have been implemented and tested using both real-world data collected with a mobile robot and simulated data. We furthermore demonstrated that the techniques allow a robot to robustly and accurately estimate the state. To support our claims, we performed an extensive collection of experiments, in which we compared the performance of our approaches with the state of the art. In addition to the theoretical contributions to advance the state of the art, our real-world experiments highlight the practicality of our techniques in challenging and realistic scenarios. We believe that the proposed approaches will allow us in the future to build systems that can assist humans in their work and everyday life.

Future Work

Despite the encouraging results presented in this thesis, there are several possibilities for further extensions, which could be addressed in future research. For example, throughout this thesis we assumed that the environment is static while the robot performs SLAM. Whereas our approaches are robust enough to cope with highly dynamic map elements, such as people or bicyclists as shown in Chapter 9, the main structures of the environment should be static. In a dynamic environment the recognition of already seen places becomes hard because the environment is subject to change. This may lead to errors in the map. Furthermore, in a lifelong learning scenario, the robot at some point in time has a good model of its environment and localizing with respect to the model might be sufficient for carrying out the task. The robot may, however, require to detect that parts of the model are outdated and re-mapping that area is potentially beneficial. So the robot is stuck between exploration and exploitation. It would be particularly interesting to investigate an integrated approach combining SLAM with path planning and exploration in a lifelong setting.

In such a lifelong setting, the wear of the robot influences the performance of the robot. The internal parameters of the robot are no longer stationary. Our parameter calibration partially addresses this issue. In a potential application, however, the robot should identify hardware failures, such as a broken sensor that reports erroneous measurements. In addition to that, our approach so far addresses only one commonly employed kinematic principle. Future research could deal with extending the online estimation of the calibration parameters to other drive systems, such as omni-directional systems or Ackermann steering.

Moreover, our hierarchical approach so far clustered the factor graph without considering the topology of the environment. Taking into account the topology allows us to segment the map into meaningful parts, such as buildings, streets and urban districts. A meaningful segmentation could also be exploited in multi-robot scenarios, where estimates of several robots have to be merged into one entire model. Furthermore, the hierarchical approach is inspired by the divide-and-conquer principle. This could be exploited to speed-up the computation by extensive parallelization of the whole estimation process.

Furthermore, the possible future research is not limited to robotics. Nowadays, navigation systems for cars, which guide us towards our desired location, are commonly available. A corresponding system is yet not available for all kinds of pedestrians. As we have seen in Chapter 9, navigation with a robot that moves similar to a pedestrian is complex since we need to identify obstacles, such as stairs and lawn. The data obtained by our navigation system could potentially be exploited to provide the necessary information for implementing a routing system for a wheelchair or for families with child strollers.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] M. Adams, S. Zhang, and L. Xie. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[2] P. Agarwal and E. Olson. Evaluating variable reordering strategies for SLAM. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[3] S. Agarwal and K. Mierle. Ceres solver: Tutorial & reference. https://code.google.com/p/ceres-solver, 2012.

[4] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010.

[5] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive GPS. In *International Conference on Pattern Recognition (ICPR)*, 2006.

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

[7] V. Andersen, H. Aanæs, and J. Bærentzen. Surfel based geometry resonstruction. *Theory and Practice of Computer Graphics 8*, 2010.

[8] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino. Simultaneous calibration of odometry and camera for a differential drive mobile robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[9] G. Antonelli and S. Chiaverini. Linear estimation of the odometric parameters for differential-drive mobile robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[10] G. Antonelli, S. Chiaverini, and G. Fusco. A calibration method for odometry of mobile robots based on the least-squares technique: theory and experimental validation. *IEEE Trans. on Robotics*, 21(5):994–1004, 2005.

[11] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *Robotics Automation Magazine*, 13(3):108 –117, 2006.

[12] B. Balaguer, S. Carpin, and S. Balakirsky. Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems. In *Workshop on Performance Evaluation and Benchmarking for Intelligent Robots*, 2007.

[13] I. Baldwin and P. Newman. Laser-only road-vehicle localization with dual 2D push-broom LIDARS and 3D priors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[14] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. R. L. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer Society Press*, 22(6):18–22, 1989.

[15] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss. The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1:127–140, 2009.

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

[17] O. Bengtsson and A.-J. Baerveldt. Robot localization based on scan-matching – setimating the covariance of for the icp algorithm. *Journal of Robotics & Autonomous Systems*, 44:29–40, 2003.

[18] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[19] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[20] P. Biber and W. Strasser. The normal distributions transform: A new approach to laser scan-matching. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[21] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Trans. on Robotics*, 12(6):869 –880, 1996.

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

[23] J. Brookshire and S. Teller. Automatic calibration of multiple coplanar sensors. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[24] J. Brookshire and S. Teller. Extrinsic calibration from per-sensor egomotion. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[25] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1998.

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

[27] M. Byrod and K. Astroem. Conjgate gradient bundle adjustment. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010.

[28] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[29] L. Carlone, R. Aragues, J. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

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

[31] A. Censi. On achievable accuracy for range-finder localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[32] A. Censi, A. Franchi, L. Marchionni, and G. Oriolo. Simultaneous calibration of odometry and sensor parameters for mobile robots. Technical Report CaltechAUTHORS:20120805-115123559, California Institute of Technology, 2012.

[33] A. Censi, L. Marchionni, and G. Oriolo. Simultaneous maximum-likelihood calibration of robot and sensor parameters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[34] S. Ceriani et al. RAWSEEDS ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4), 2009.

[35] C. Chen and H. Wang. Large-scale loop-closing by fusing range data and aerial image. *Int. Journal of Robotics and Automation*, 22(2):160–169, 2007.

[36] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1991.

[37] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):1–14, 2008.

[38] L. B. Cremean et al. Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*, 23(9):777–810, 2006.

[39] M. Cummins and P. Newman. Highly scalable appearance-only SLAM – FAB-MAP 2.0. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[40] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006. Part of the SIAM Book Series on the Fundamentals of Algorithms.

[41] A. Davison and N. Kita. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2001.

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

[43] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. Journal of Robotics Research*, 25(12):1181–1204, December 2006.

[44] M. Ding, K. Lyngbaek, and A. Zakhor. Automatic registration of aerial imagery with untextured 3D LiDAR models. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2008.

[45] C. U. Dogruer, K. A. Bugra, and M. Dolen. Global urban localization of outdoor mobile robots using satellite images. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[46] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous driving in unknown environments. In *Proc. of the Int. Symposium on Experimental Robotic (ISER)*, July 2008.

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

[48] A. Doucet, N. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2000.

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

[50] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13:99–110, 2006.

[51] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003.

[52] A. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2004.

[53] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[54] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Trans. on Robotics*, 21(4):588–596, August 2005.

[55] The European robotic pedestrian assistant. http://europa.informatik.uni-freiburg.de, 2009.

[56] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

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

[58] Experiment videos. http://www.informatik.uni-freiburg.de/∼kuemmerl/media.html. last visited on 12/18/2012.

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

[60] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[61] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, 2005.

[62] J. D. Foley, A. van Dam, K. Feiner, J. F. Hughes, and R. L. Phillips. *Introduction to Computer Graphics*. Addison-Wesley, 1993.

[63] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *Int. Journal of Robotics Research*, 22(12):985–1003, 2003.

[64] U. Frese. A proof for the approximate sparsity of SLAM information matrices. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

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

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

[67] S. Friedberg, A. Insel, and L. Spence. *Linear Algebra*. Pearson, 4th edition, 2002.

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

[69] C. Gao and J. R. Spletzer. On-line calibration of multiple lidars on a mobile vehicle platform. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[70] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and its Applications*, 34(0):3 – 28, 1980.

[71] Google self-driving car project. http://googleblog.blogspot.com, 2012.

[72] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008.

[73] G. Grisetti, R. Kümmerle, and K. Ni. Robust optimization of factor graphs by using condensed measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[74] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[75] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

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

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

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

[79] H.-M. Gross, H. Boehme, C. Schroeter, S. Mueller, A. Koenig, E. Einhorn, C. Martin, M. Merten, and A. Bley. TOOMAS: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[80] S. Grzonka, G. Grisetti, and W. Burgard. A fully autonomous indoor quadrotor. *IEEE Trans. on Robotics*, 8(1):90–100, 2012.

[81] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

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

[83] D. Haehnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[84] D. Haehnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in SLAM. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2003.

[85] D. Hähnel, S. Thrun, and W. Burgard. An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003.

[86] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[87] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1989.

[88] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 2011.

[89] A. Hornung, K. M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[90] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[91] A. Howard. Multi-robot mapping using manifold representations. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

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

[93] S. Huang, H. Wang, U. Frese, and G. Dissanayake. On the number of local minima to the point feature based slam problem. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[94] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2010.

[95] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In *Proceedings of the Internetional Conference on Computer Vision - Workshop on Dynamical Vision*, 2007.

[96] S. J. Julier. The scaled unscented transformation. In *Proc. of the American Control Conference*, 2002.

[97] S. J. Julier, R. De Nardi, and J. D. B. Nelson. Multi-rate estimation of coloured noise models in graph-based estimation algorithms. In *International Conference on Information Fusion (FUSION)*, 2012.

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

[99] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Journal of Robotics & Autonomous Systems*, 57:1198–1210, December 2009.

[100] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Int. Journal of Robotics Research*, 31:217–236, February 2012.

[101] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, December 2008.

[102] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. of the ACM/IEEE Conf. on Supercomputing*, 1998.

[103] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic road maps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, pages 566–580, 1996.

[104] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *Int. Journal of Robotics Research*, 30(1):56–79, 2011.

[105] A. Kleiner and C. Dornhege. Mapping for the support of first responders in critical domains. *Journal of Intelligent and Robotic Systems*, 64(1):7–31, 2010.

[106] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[107] K. Konolige. Sparse sparse bundle adjustment. In *Proc. of the British Machine Vision Conference (BMVC)*, 2010.

[108] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Trans. on Robotics and Automation*, 24(5):1066–1077, 2008.

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

[110] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[111] K. Konolige. Large-scale map-making. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[112] K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[113] T. Korah and C. Rasmussen. Probabilistic contour extraction with model-switching for vehicle localization. In *IEEE Intelligent Vehicles Symposium*, pages 710–715, 2004.

[114] R. Kümmerle, G. Grisetti, and W. Burgard. Simultaneous calibration, localization, and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[115] R. Kümmerle, G. Grisetti, and W. Burgard. Simultaneous parameter calibration, localization, and mapping. *Advanced Robotics*, 26(17):2021–2041, 2012.

[116] R. Kümmerle, G. Grisetti, C. Stachniss, and W. Burgard. Simultaneous parameter calibration, localization, and mapping for robust service robotics. In *Proc. of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2011.

[117] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

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

[119] R. Kümmerle, P. Pfaff, R. Triebel, and W. Burgard. Active Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Fachgespräche Autonome Mobile Systeme (AMS)*, 2007.

[120] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A navigation system for robots operating in crowded urban environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013. Accepted for Publication.

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

[122] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.

[123] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[124] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, 2007.

[125] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics*, 25:346–359, 2008.

[126] J. Kurlbaum and U. Frese. A benchmark data set for data association. Technical report, University Bremen, Germany, 2008. http://www.informatik.uni-bremen.de/agebv/en/DlrSpatialCognitionDataSet.

[127] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *Int. Journal of Robotics Research*, 21(10-11):917–942, 2002.

[128] Y. Latif, C. C. Lerma, and J. Neira. Robust loop closing over time. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[129] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University, 1998.

[130] J. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer Verlag, 2003.

[131] K. W. Lee, S. Wijesoma, and J. I. Guzmán. A constrained SLAM approach to robust and accurate localisation of autonomous ground vehicles. *Journal of Robotics & Autonomous Systems*, 55(7):527–540, 2007.

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

[133] K. Y. K. Leung, C. M. Clark, and J. P. Huissoon. Localization in urban environments by matching ground level video images with an aerial image. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[134] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[135] H. Li, R. W. Sumner, and M. Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum*, 27(5), 2008.

[136] H. Li and R. Hartley. Five-point motion estimation made easy. In *Proc. of the Int. Conf. on Pattern Recognition (ICPR)*, 2006.

[137] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proc. of Robotics: Science and Systems (RSS)*, 2008.

[138] J. Lim, J.-M. Frahm, and M. Pollefeys. Online environment mapping. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2011.

[139] K. Lingemann, H. Surmann, A. Nüchter, and J. Hertzberg. High-speed laser localization for mobile robots. *Journal of Robotics & Autonomous Systems*, 51(4):275–296, 2005.

[140] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[141] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzman. Accurate visual odometry from a rear parking camera. In *Intelligent Vehicles Symposium (IV)*, 2011.

[142] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal on Computer Vision*, 60(2):91–110, November 2004.

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

[144] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 1998.

[145] W. Maddern, A. Harrison, and P. Newman. Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D lidars. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[146] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems (2nd ed.), 2004.

[147] M. Magnusson, T. Duckett, and A. J. Lilienthal. 3D scan registration for autonomous mining vehicles. *Journal of Field Robotics*, 24(10):803–827, 2007.

[148] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based SLAM using visual loop closures. *IEEE Trans. on Robotics and Automation*, 24:1002–1014, 2008.

[149] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[150] A. Martinelli and R. Siegwart. Estimating the odometry error of a mobile robot during navigation. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2003.

[151] K. Matsuo and J. Miura. Outdoor visual localization with a hand-drawn line drawing map using FastSLAM with PSO-based mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[152] M. Montemerlo et al. Junior: The Stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[153] M. Montemerlo and S. Thrun. Large-scale robotic 3-d mapping of urban structures. In *Proc. of the Int. Symposium on Experimental Robotic (ISER)*, 2004.

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

[155] H. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, Robotics Institute, 1996.

[156] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[157] 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.

[158] K. Ni and F. Dellaert. Multi-level submap based SLAM using nested dissection. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[159] K. Ni and F. Dellaert. HyperSfM. In *Proc. of the IEEE Int. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012.

[160] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[161] T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J. R. Bohmer. Stadtpilot: First fully autonomous test drives in urban traffic. In *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2011.

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

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

[164] 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.

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

[166] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[167] E. Olson and M. Kaess. Evaluating the performance of map optimization algorithms. In *RSS Workshop on Good Experimental Methodology in Robotics*, 2009.

[168] G. Parekh, M. Skubic, O. Sjahputera, and J. M. Keller. Scene matching between a map and a hand drawn sketch using spatial relations. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[169] I. Paromtchik and C. Laugier. Autonomous parallel parking of a nonholonomic vehicle. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 1996.

[170] I. Paromtchik and C. Laugier. Automatic parallel parking and returning to traffic maneuvers. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1997.

[171] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot. 3-D modelling and robot localization from visual and range data in natural scenes. In *1st International Conference on Computer Vision Systems (ICVS)*, number 1542 in LNCS, 1999.

[172] M. P. Parsley and S. J. Julier. Towards the exploitation of prior information in SLAM. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[173] M. P. Parsley and S. J. Julier. Exploiting prior information in GraphSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[174] P. Pfaff, R. Kümmerle, D. Joho, C. Stachniss, R. Triebel, and W. Burgard. Navigation in combined outdoor and indoor environments using multi-level surface maps. In *Workshop on Safe Navigation in Open and Dynamic Environments*, 2007.

[175] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *Int. Journal of Robotics Research*, 26(2):217–230, 2007.

[176] O. Pink and C. Stiller. Automated map generation from aerial images for precise vehicle localization. In *Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2010.

[177] E. Plaku, L. Kavraki, and M. Vardi. Discrete search leading continuous exploration for kinodynamic motion planning. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[178] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes, 2nd Edition*. Cambridge Univ. Press, 1992.

[179] D. M. Rosen, M. Kaess, and J. J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[180] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[181] M. Rufli, D. Ferguson, and R. Siegwart. Smooth path planning in constrained environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[182] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[183] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Range sensor based model construction by sparse surface adjustment. In *Proc. of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2011.

[184] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate 3D surface models by sparse surface adjustment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[185] A. Sanfeliu. URUS project: Communication systems. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009. Workshop on Network Robots Systems.

[186] A. Schanz, A. Spieker, and K.-D. Kuhnert. Autonomous parking in subterranean garages-a look at the position estimation. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2003.

[187] T. Schoen and F. Lindsten. Manipulating the multivariate gaussian density. Technical report, Linkoeping University, 2011.

[188] A. V. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[189] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[190] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[191] R. Siegwart et al. RoboX at Expo.02: A large-scale installation of personal robots. *Journal of Robotics & Autonomous Systems*, 42(3-4), 2003.

[192] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *Int. Journal of Robotics Research*, 28(5):595–599, May 2009.

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

[194] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 1988.

[195] J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 2005.

[196] B. Sofman, E. L. Ratliff, J. A. Bagnell, N. Vandapel, and T. Stentz. Improving robot navigation through self-supervised online learning. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

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

[198] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[199] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2011.

[200] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[201] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[202] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[203] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[204] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

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

[206] S. Thrun et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, June 2006.

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

[208] G. D. Tipaldi, M. Braun, and K. O. Arras. FLIRT: Interest regions for 2D range data with applications to robot navigation. In *Proc. of the Int. Symposium on Experimental Robotic (ISER)*, 2010.

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

[210] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

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

[212] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[213] J. Underwood, A. Hill, and S. Scheding. Calibration of range sensor pose on mobile platforms. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

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

[215] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

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

[217] K. M. Wurm, H. Kretzschmar, R. Kümmerle, C. Stachniss, and W. Burgard. Identifying vegetation from laser data in structured outdoor environments. *Robotics and Autonomous Systems*, 2012. In Press.

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

[219] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.