Robot Perception for Indoor Navigation

Felix Endres

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Robot Perception for Indoor Navigation

Felix Endres

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

Dekan	Prof. Dr. Georg Lausen
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Thomas Brox
	Albert-Ludwigs-Universität Freiburg
Tag der Disputation	4. August 2015

Abstract

In recent years, commercially available mobile robots, that operate in indoor environments, have found their ways into private homes, office environments, and industrial settings. They fulfill important duties such as transportation, telepresence, and cleaning. While some tasks, such as vacuum cleaning, can be achieved with rudimentary perception, for more complex tasks sophisticated perception capabilities are fundamental for autonomous robots. Even for the mentioned vacuum cleaning, improved perception allows for substantial increases in efficiency. In this thesis, we investigate novel approaches for the modelling of indoor environments for robot navigation. Being an important foundation for higher level skills, a particular focus lies on simultaneous localization and mapping (SLAM), which allows a robot to construct a model of its environment during operation. In the context of SLAM, we develop an approach for RGB-D cameras, that captures dense 3D maps for robot navigation. For this SLAM system, we propose novel methods to increase the accuracy of the trajectory estimation and the robustness against misassociations during individual motion estimates. Further, we address a major limitation on the hardware side of RGB-D cameras, namely the limited field of view. We investigate SLAM with multiple RGB-D cameras and develop an approach for automated extrinsic calibration of RGB-D cameras via SLAM. We further propose an extension of RGB-D sensors with mirrors to bisect the field of view into two roughly opposite views. While this does not increase the overall information perceived, we show that the divided field of view is beneficial in the context of SLAM. Additionally, we exploit the structural properties of this catadioptric extension to constrain the mentioned calibration method, such that planar motion of the robot is sufficient for online calibration of the two views. To autonomously access all areas in a private home or office, a further key skill for robot navigation is the operation of doors. In this context, we extend the state of the art by novel methods for learning a model of the kinematics and dynamics of a door. We demonstrate that the knowledge about the dynamics of a door allows the robot to accurately predict the motion of the door from inertia. We use this ability to employ a door opening strategy with low requirements on the dexterous workspace of the manipulator. To show the benefits of the approaches proposed in this thesis, we thoroughly evaluate them in experiments with real robots and real sensor data. Overall, the proposed approaches lower the cost of the sensing equipment and the required complexity of the manipulator. These factors are particular important for commercial robots targeted at households and small businesses.

Zusammenfassung

Roboter wurden in der Vergangenheit hauptsächlich im Bereich der Automatisierung der industriellen Fertigung eingesetzt, in Deutschland z.B. in der Automobilindustrie. Solche Roboter sind jedoch nur sehr eingeschränkt autonom – im Normalfall können sie lediglich sehr begrenzt auf ihre Umgebung reagieren und folgen einer strikten Programmroutine. Daher ist es notwendig beim Einsatz solcher Roboter die Arbeitsumgebung stark an diese anzupassen. In der industriellen Fertigung werden daher speziell strukturierte Bereiche eingerichtet, in denen Roboter am Fließband oder der Fertigungsstraße eindeutig vordefinierte Bedingungen vorfinden. So wird zum Beispiel die Position eines Werkstücks auf dem Förderband so kontrolliert, dass der Erfolg des vorprogrammierten Bewegungsablaufs garantiert werden kann.

Autonome Roboter hingegen setzen ihre Sensoren ein, um ihre Umgebung wahrzunehmen und sich so ein Modell ihrer Umwelt zu verschaffen. Anhand des Modells können autonome Roboter dann Entscheidungen über ihre Aktionen treffen. Die Forschung an solchen autonomen, intelligenten Systemen hat in den letzten Jahrzehnten große Fortschritte gemacht. Besonders die Weiterentwicklung probabilistischer Verfahren hat den Erfolg im Umgang mit Sensormessungen und dem zugehörigen Messrauschen vorangetrieben. Dadurch wurden spannende neue Anwendungen wie z.B. Häfen mit automatisierter Schiffsbe- und entladung ermöglicht. In den letzten Jahren sind auch erste autonome Roboter für den nicht-industriellen Bereich auf den Markt gekommen, z.B. in Form von autonomen Staubsaugern und -wischern, Rasenmähern und Telepräsenzrobotern. Insgesamt ist davon auszugehen, dass Roboter in Zukunft immer mehr Arbeiten übernehmen können.

Das Ziel dieser Arbeit ist es, die Grundlagen der Autonomie von mobilen Robotern weiter zu entwickeln. Unser Fokus liegt dabei auf Robotern, die innerhalb von Gebäuden eingesetzt werden, wie z.B. Haushaltsroboter. Eine wichtiges Fundament für einen mobilen Roboter ist die Möglichkeit in der Umgebung zu navigieren. Dazu muss sich der Roboter einerseits lokalisieren können, andererseits sollte er seine Umgebung kartieren können. Wurde der Roboter nicht im Vorfeld mit einer Karte seines Einsatzorts ausgestattet – was gerade im Konsumentenbereich kaum möglich ist – muss er gleichzeitig eine Karte erstellen, während er sich anhand dieser lokalisiert. Im Englischen wird dieses Problem *simultaneous localization and mapping*, kurz SLAM, genannt. Die Forschung des letzten Jahrzehnts hat viele Ansätze zur Lösung des SLAM-Problems hervorgebracht.

Für diese Ansätze wurden meist Laserscanner oder Kameras verwendet, um die Umgebung wahrzunehmen. Laserscanner tasten die Umgebung mit einem rotierenden Strahl ab, so dass z.B. ein horizontaler Schnitt des Gebäudes aus Sicht des Roboters vermessen wird. Aus einer Serie solcher Messungen kann mittels SLAM-Verfahren eine konsistente zweidimensionale Karte erstellt werden. Alternative können SLAM-Verfahren auch andere Sensoren wie Kameras verwenden. Bei einzelnen Kameras ist es dabei schwierig genaue metrische Informationen zu gewinnen, weswegen für SLAM häufig Stereokameras verwendet werden. Mit diesen können metrische Informationen für die Pixel durch Triangulation berechnet werden. Dazu ist es notwendig die jeweiligen Pixel einander zuzuordnen, auf die ein Punkt in den beiden Kameras projiziert wird. Dies ist gerade im Innenbereich oft problematisch, da diese Zuordnung über die Textur des Bildes geschieht. Auf einfarbigen Wänden, Decken oder auch Böden ist eine eindeutige Zuordnung daher kaum möglich. Neuartige RGB-D Kameras, die in der Unterhaltungselektronikbranche entwickelt wurden, umgehen dieses Problem, indem die Triangulation nicht zwischen zwei Bildern geschieht, sondern zwischen einem aktiv projizierten Infrarotmuster und dessen Abbildung in einer Infrarotkamera. Zusätzlich wird mit einer regulären Kamera ein Farbbild aufgenommen. In den aktuellen Modellen funktionieren die Distanzmessungen für den Bereich von ca. 0,6 bis 5 m. Im Nahbereich erreichen die Distanzmessungen dabei eine Genauigkeit im Bereich weniger Millimeter, allerdings nimmt der Messfehler bedingt durch die Funktionsweise quadratisch mit der Distanz zu.

Mit diesen Eigenschaften sind RGB-D Kameras besonders zum Erstellen von dreidimensionalen Umgebungsmodellen in Innenräumen geeignet. Wir erarbeiten daher in den Kapiteln 3 und 4 ein SLAM-System, das sich die Eigenschaften von RGB-D Kameras zu nutze macht, um ein möglichst genaues Umgebungsmodell für die Navigation mobiler Roboter zu erstellen. Dazu übertragen wir in Kapitel 3.1 zunächst SLAM-Techniken, die für Stereokameras oder Laserscanner eingesetzt werden, und evaluieren die resultierenden Eigenschaften des Systems. In Kapitel 3.2 diskutieren wir dafür geeignete Fehlermaße und stellen Testdatensätze vor, die verschiedenste Szenarien abdecken. Wir stellen fest, dass das System unter bestimmten Bedingungen bereits eine hohe Genauigkeit erreicht. In schwierigen Szenarien, z.B. in großen Räumen oder bei starker Bewegungsunschärfe, sinkt die Leistung jedoch ab. In Kapitel 3 analysieren wir die Probleme die unter schwierigeren Bedingungen auftreten und stellen mehrere Ansätze zur Verbesserung des RGB-D SLAM-Systems vor. Kapitel 3.3 beschreibt mehrere Techniken, mit denen wir die Suche und den Vergleich von visuellen Merkmalen verbessern, um sowohl die Laufzeit zu reduzieren, als auch die Schätzung der Sensorbewegung zu verbessern. In Kapitel 3.4 benutzen wir den Posengraph, eine Graphstruktur bestehend aus den Sensorposen als Knoten und den relativen Bewegungsschätzungen zwischen diesen als Kanten, um die Kandidaten für zukünftige Bewegungsschätzungen zu leiten. Dadurch verbessert sich die Genauigkeit, wenn der Roboter während dem Kartieren einen Ort zum wiederholten

Mal besucht. Konkret nutzen wir die Graph-Nachbarschaft, um Kandidaten für die Suche nach Merkmalskorrespondenzen zu bestimmen, da sich bereits erfolgreich berechnete Bewegungsschätzungen in dieser Nachbarschaft widerspiegeln. In Kapitel 3.6 stellen wir einen Ansatz zur Bewertung der Plausibilität von Bewegungsschätzungen mittels eines Sensormodells vor, der sich das Prinzip der Sichtlinie zunutze macht. Bestehende Sensormodelle, wie sie z.B. in der Monte-Carlo Lokalisierung verwendet werden, erlauben eine relative Gewichtung verschiedener Hypothesen. Wir erarbeiten hingegen ein absolutes Qualitätsmaß.

Diese Entwicklungen werten wir umfassend aus. Dies erfolgt anhand des beschriebenen Evaluierungsdatensatzes, der es uns erlaubt, die Genauigkeit der Trajektorienrekonstruktion zu berechnen. Appendix A.1 listet die Ergebnisse unseres Ansatzes für über 40 Sequenzen auf, die verschiedenste Szenarien abdecken, wie z.B. SLAM mit einer handgeführten oder auf einem Roboter montierten Kamera. In Kapitel 3.7 vergleichen wir die Ergebnisse mit anderen Ansätzen und zeigen, dass unser Ansatz aktuell zu den besten SLAM-Systemen gehört.

Unsere Experimente belegen, dass RGB-D Kameras sich hervorragend als Sensoren für die Kartierung eignen. Eine der größten verbleibenden Limitierungen ist aber das kleine Sichtfeld. Im Gegensatz zu 3D-Laserscannern, die horizontal meist zwischen 180° und 360° abdecken, ist das Sichtfeld horizontal nur etwa 57° und vertikal etwa 43° groß. Dies und die eingeschränkte Tiefenwahrnehmung (zwischen 0,6 bis 5 m) führen leicht zu Situationen, in denen nur wenige Daten verfügbar sind, die für die Bewegungsschätzung verwendet werden können. Gerade bei autonomen Robotern mit statisch montiertem Sensor kann dies zu Problemen führen, z.B. wenn sich der Roboter unmittelbar vor einer Tür oder Wand befindet. Deshalb erweitern wir unser SLAM System in Kapitel 4, so dass mehrere Sensoren verwendet werden können. Wir evaluieren die Vorteile eines zweiten Sensors in Kapitel 4.1. Dabei zeigt sich, dass ein zweiter Sensor kritische Informationen liefern kann, die die Kartierung stark verbessern können. Mehrere Sensoren bringen aber auch höhere Anschaffungskosten, ein höheres Gewicht und höheren Strombedarf mit sich. Wir entwickeln daher in Kapitel 4.2 eine Erweiterung für RGB-D Kameras, die das Blickfeld mit Spiegeln aufteilt. Wir zeigen, dass diese Lösung in einer SLAM Anwendung ähnliche Vorteile für die Genauigkeit bringt wie ein zweiter Sensor.

Um die Daten von mehreren RGB-D Sensoren zu vereinen, müssen deren relative Positionen kalibriert sein. In Kapitel 4.3 stellen wir einen Ansatz vor, diese Kalibrierung direkt während der Kartierung zu bestimmen. Dazu formulieren wir Fehlerterme, die sowohl Informationen aus der geschätzten jeweiligen Eigenbewegung der Sensoren nutzen, als auch gemeinsame, zu unterschiedlichen Zeiten entstandene Observationen der beiden Sensoren. Dieser Ansatz lässt sich sowohl für reguläre RGB-D Kameras nutzen, als auch um die Teil-Sichtfelder des katadioptrischen RGB-D Sensors zueinander zu kalibrieren. Für letzteren Sensor entwickeln wir in Kapitel 4.4 spezifischere Varianten

der Fehlerterme, welche die bekannte Struktur des Sensors ausnutzt, um die Freiheitsgrade des Optimierungsproblems zu vermindern. Wir zeigen experimentell, dass hierdurch das Konvergenzverhalten der Kalibrierung verbessert wird. Insbesondere ermöglicht der Ansatz die Kalibrierung der Sichtfelder allein durch Bewegung in der Ebene, also z.B. während der ersten Fahrt eines Haushaltsroboters.

Ist ein Roboter in der Lage Innenräume zu kartieren und sich zu orientieren kann er autonome Reinigungs- oder Logistikaufgaben erfüllen. Die aktuelle Generation kommerzieller Roboter ist allerdings nicht in der Lage Türen zu öffnen. Dies bedeutet in den meisten Gebäuden wiederum dass die Umgebung an den Roboter angepasst werden muss. Türen kontrolliert zu öffnen ist für Roboter eine hoch komplexe Aufgabe, für die die bisherigen Ansätze zumeist hohe Anforderungen an die Freiheitsgrade und Reichweite des Manipulators stellen. In Kapitel 5 erarbeiten wir Verfahren, mit denen ein Roboter die kinematischen und dynamischen Eigenschaften einer Tür anhand seiner Beobachtungen modellieren kann. Kapitel 5.2 behandelt dabei die Lernverfahren, die auf 3D Daten basieren. Um dem Roboter zu ermöglichen seine Umgebung autonom zu modellieren, entwickeln wir in Kapitel 5.3 ein Verfahren mit dem der Roboter sein Modell interaktiv initialisieren kann. Dafür nutzen wir taktile Messungen während der ersten Manipulation der Tür, um eine ungefähre Schätzung der Türdynamik vorzunehmen. Unsere Experimente mit einem Roboter zeigen, dass die Informationen in diesem Modell benutzt werden können, um die Aufgabenstellung stark zu vereinfachen.

Zusammengenommen ergeben die beschriebenen Verfahren eine neue Grundlage für die Navigation in Gebäuden, auf welcher Ansätze zur Lösung komplexerer Aufgaben entwickelt werden können. Im Vergleich zu bisherigen Ansätzen senken unsere Ergebnisse dabei die Anforderungen an die Sensoren und Manipulatoren der Roboter wesentlich. Damit leisten wir der weiteren Verbreitung von autonomen Robotern in Haushalten und Unternehmen Vorschub.

Acknowledgment

In the following I would like to express my gratitude to the people who made this thesis possible. First of all, I would like to thank my advisor Prof. Dr. Wolfram Burgard, head of the Autonomous Intelligent Systems lab, for giving me the opportunity to work in his group and his continuous support throughout the years. I also want to thank Prof. Dr. Jeff Trinkle, Dr. Jürgen Sturm and Dr. Rainer Kümmerle for their helpful and motivating supervision of various parts of my work. I also thank Prof. Dr. Thomas Brox for acting as a reviewer for this thesis.

I am deeply grateful to the smart and cordial bunch of people in the AIS lab. I have greatly progressed in many areas thanks to the close collaborations and discussions in a way which would have been impossible to achieve on my own. Their companionship has made my time in the lab an outstanding part of my life. There are too many to name all of them, but I would like to mention a few who have made a special impact. First of all, I want to reach out to Jürgen Hess, who brought me to the AIS lab during my Master's. He sparked the work on RGB-D SLAM and as my office mate was the first person to bounce my ideas off and to share the joy and sorrows of a PhD student with. Further I want to mention Christoph Sprunk, who I collaborated with in several endeavors and who always proved incredibly helpful and supportive.

Last but not least I would like to thank my family and my partner Lena Wenz for their support and love throughout the years.

Contents

Chapter 1

Introduction

While robots have revolutionized industrial mass-production several decades ago, these robots are hardly more than versatile machines running a complex, but fixed program. In general, manufacturing robots do not exhibit autonomous intelligence. Except for basic control flow, they are mostly unaware of their environment and only limited sensory input is used during operation. Therefore, great care is taken that the environment of the robot is as predictable as possible, e.g., by using fences to lock humans out of the robot's workspace. This almost complete lack of autonomy is one of the major obstacles which has to be overcome to allow robots to become mobile in a priori unknown environments and possibly to interact with humans.

However, tremendous steps towards the goal of autonomous intelligent systems have been achieved in the current millennium. In recent years, autonomous robots that help out with household tasks have arrived in millions of homes, such as floor cleaning and lawn mowing robots. Many companies have started up to make a business out of robots that fulfill complex tasks, involving operation in unknown environments and interaction with humans. Important groundwork for this revolution has been carried out in the robotics research community. Particularly the rise of probabilistic methods has led to huge advancements in the ability of robots to use their sensors in order to acquire a model of the real world. Combined with the progress in sensor technology and processing power, the improvement of robotic skills proceeds at a fast pace.

This thesis describes our research undertaken to help mobile robots conquer indoor environments. Our contributions rank among the many efforts of the robotics community to refine the fundamental skills of such robots. When a robot is deployed in an environment such as a home or an office space, it is usually not feasible to equip the robot with an accurate model of that environment a priori. Therefore, the robot will first need to create a model of the world. For a mobile robot, this model in general needs to comprise a map that allows the robot to localize and plan a collision-free path according to its assignment. Building such a map from on-board sensors is a challenging problem, as the robot needs to be localized to create a map from sensor measurements, but needs a map to localize itself. These tasks therefore need to be done concurrently and the problem is hence re-

ferred to as simultaneous localization and mapping (SLAM). The SLAM problem has been subject to intensive research and feasible solutions have been developed for many scenarios. For robots operating in the plane, the use of onboard laser range scanners, that measure the closest obstacles in this plane lead to very accurate results. Cameras have been a viable alternative, particularly for applications where the robot moves in three dimensions and where laser scanners are too expensive or too heavy, e.g., micro aerial vehicles such as quadrocopters. However, extracting distance information – as required for precise localization and for building geometric maps – from monocular camera images is a hard problem. For this reason, often stereo camera systems have been deployed, which exploit the known baseline between the camera pair to use triangulation to obtain the distance to visual features that could be identified in both images. A new type of 3D camera, called RGB-D camera, has recently been developed in the entertainment industry. As the term "RGB-D" indicates, these cameras provide dense distance (D) measurements for almost all color (RGB) pixels. Due to mass production, RGB-D cameras combine the advantages of cameras, such as low price, low weight, high frame rate, with the ability to obtain accurate geometric information. To exploit the potential of this novel sensor type in robotics, we created a SLAM system for RGB-D cameras, which allows a robot to create a three-dimensional map of its environment in real time during operation. The system is presented in Chapter 3. We first built it by transferring best practices from laser and vision based SLAM approaches to the new sensor system and evaluated the respective performance. We then present several novel methods to improve the accuracy and robustness of RGB-D SLAM specifically, with a focus on challenging scenarios.

An important measure to robustness that stands out from the proposed algorithmic contributions is the use of multiple viewpoints, which needs to be implemented in software as well as in hardware. In Chapter 4 we extend our SLAM system to handle multiple RGB-D cameras and demonstrate the impact of the second sensor on the accuracy in experiments. As an alternative for robots with tight financial or payload constraints, we propose a novel catadioptric setup, an RGB-D camera combined with mirrors. The mirrors split the field of view of a single RGB-D camera such that, e.g., a robot obtains a front and a rear view. We show that this configuration leads to comparable benefits in a SLAM application as a second RGB-D camera. The fusion of multiple RGB-D sensors inherently requires their extrinsic calibration. We propose a novel method for online self-calibration of the extrinsic parameters via SLAM. In case of the catadioptric RGB-D sensor, we further derive a method to exploit the known structure of the device to allow calibration from planar motion only, so that a wheeled robot can fully self-calibrate during operation.

Besides the creation of a map, the navigation of robots in indoor environments typically requires the opening of doors. To fulfill a transportation task across rooms or to vacuum-clean all rooms, a robot needs to be able to open doors to be independent of human help. In Chapter 5 we investigate the modelling of doors from sensor data. In the past,

research has focused on learning the kinematic properties of the door. We propose to additionally learn a model of the dynamics of the door, as this allows to handle the door in a more dexterous way. We devise an approach to learn the kinematics and dynamics from depth perception which enables the robot to accurately predict the behavior of the door and to open it with a point contact instead of a fixed grasp. This reduces the kinematic complexity required, meaning that smaller, cheaper manipulators can be used.

1.1 Key Contributions

To summarize our key contributions to the field of indoor perception for autonomous mobile robots, we propose

- a SLAM system for RGB-D cameras which employs novel techniques to provide accurate results even in challenging scenarios,
- an approach for automated extrinsic calibration for multiple RGB-D cameras that incorporates the calibration into the SLAM backend,
- a catadioptric RGB-D sensor based on a single RGB-D camera, which yields a high performance benefit for the accuracy of SLAM as compared to a regular RGB-D camera,
- an approach for quick and accurate estimation of the kinematics of a door from perception with a depth sensor,
- an approach for learning the dynamics of a door from observing demonstrations or by autonomous interaction.

To validate the approaches proposed in this thesis we thoroughly evaluated them on data from real sensors and, if applicable, applied them on real robots.

Our publications regarding RGB-D SLAM were met with vivid interest. We participated with the initial version of our RGB-D SLAM software in the ROS 3D contest [30] in 2011, and won the first place in the "most useful" category. The software has since been used by many groups as a building block. Our evaluation of the RGB-D SLAM system, published in 2012 [33], has been cited over 180 times. Our paper which describes the evaluation benchmark [117] has been cited over 160 times since its publication in 2012. Our journal article, which describes most of the contributions in Chapter 3 was recently published in the *Transactions on Robotics*, a journal with one of the highest impact factors in robotics and has led the journal's ranking of the most popular papers.

1.2 Open Source Software

We released major parts of the software developed for this thesis as open source. The RGB-D SLAM system is available as a github repository[1]. The extension to multiple viewpoints and a CAD model to create the catadioptric sensor proposed in Section 4.2 is available on a separate project page[2]. The estimation of the kinematic structure of a door is available in the Google code repository of the Autonomous Intelligent Systems laboratory[3]. Further, smaller contributions in the form of bug reports, patches or new functionality have been made to the evaluation tools of the RGB-D benchmark [4], OpenCV[5] and ROS[6].

1.3 Publications

Major parts of this thesis have been published to the scientific community during the course of work, either in the form of journal articles or as papers submitted to and presented at conferences. This section lists these publications.

- 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 International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, May 2012.

- F. Endres, J. Hess, N. Engelhard, J. Sturm, and W. Burgard. 6D visual SLAM for RGB-D sensors. *at - Automatisierungstechnik*, 60:270–278, May 2012.

- 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)*, Vilamoura, Portugal, 2012.

- F. Endres, J. Trinkle, and W. Burgard. Learning the dynamics of doors for robotic manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

- F. Endres, J. Trinkle, and W. Burgard. Interactive perception for learning the dynamics of articulated objects. In *Proceedings of the ICRA 2013 Mobile Manipulation Workshop on Interactive Perception*, Karlsruhe, Germany, May 2013.

[1] https://github.com/felixendres/rgbdslam_v2
[2] http://ais.informatik.uni-freiburg.de/projects/datasets/catadioptric-rgbd/
[3] http://alufr-ros-pkg.googlecode.com/svn/trunk/dynamic_door_manipulation
[4] https://svncvpr.in.tum.de/cvpr-ros-pkg/trunk/rgbd_benchmark/rgbd_benchmark_tools
[5] http://opencv.itseez.com
[6] http://ros.org

- F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3D mapping with an RGB-D camera. *IEEE Trans. on Robotics*, 30(1):177–187, Feb 2014.

- F. Endres, C. Sprunk, R. Kuemmerle, and W. Burgard. A catadioptric extension for RGB-D cameras. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

This thesis does not report on the following publications that have been published during my time as a research assistant under the supervision of Prof. Wolfram Burgard.

- S. Ito, F. Endres, M. Kuderer, G. D. Tipaldi, C. Stachniss, and W. Burgard. W-RGB-D: Floor-plan-based indoor global localization using a depth camera and WiFi. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

- F. Endres, J. Hess, and W. Burgard. Graph-based action models for human motion classification. In *ROBOTIK*, May 2012.

- C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin. A nonparametric learning approach to range sensing from omnidirectional vision. *Robotics and Autonomous Systems*, 58(6):762 – 772, 2010.

- F. Endres, C. Plagemann, C. Stachniss, and W. Burgard. Scene analysis using latent Dirichlet allocation. In *Proc. of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

- F. Endres, J. Hess, N. Franklin, C. Plagemann, C. Stachniss, and W. Burgard. Estimating range information from monocular vision. In *Workshop Regression in Robotics - Approaches and Applications at Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

1.4 Collaborations

Parts of the work for this thesis was done in collaboration with others. To clarify the attribution of the work described in this thesis, collaborations beyond the supervision of my Ph.D. advisor Wolfram Burgard and the fruitful discussions with many colleagues is detailed below.

- The basis of the RGB-D SLAM system as described in Section 3.1 is collaborate work with Jürgen Hess and Nicholas Engelhard, working under the supervision of Jürgen Sturm.

- The RGB-D benchmark described in Section 3.2 is joint work with Jürgen Sturm, where the author was mainly involved in the analysis of the error metrics (Section 3.2.2), testing of the dataset and evaluation tools, and served as dynamic object in a few sequences.
- The extrinsic calibration approach described in Section 4.3 was developed under supervision of Rainer Kümmerle. The experiments in Sections 4.3 and 4.4 where performed together with Christoph Sprunk.
- The work on modeling of door dynamics for robotic manipulation, described in Chapter 5, was developed under the supervision of Jeff Trinkle.

1.5 Notation

We use the following notation throughout this thesis.

Symbol	Meaning
$x, v, I, F \ldots$	scalar value
$\boldsymbol{x}, \boldsymbol{n} \ldots$	column vector
$\boldsymbol{x}^T, \boldsymbol{n}^T \ldots$	row vector
$\boldsymbol{X}, \boldsymbol{M}, \ldots$	matrix
$\boldsymbol{x}^T, \boldsymbol{M}^T \ldots$	transposed vector, matrix
$\boldsymbol{\Lambda}, \boldsymbol{\Sigma}$	information matrix, covariance matrix
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{C}, \mathcal{G}, \mathcal{T} \ldots$	set
$\mathcal{D}$	training data set
$\overline{\boldsymbol{x}}$	mean of a set $\{\ldots, \boldsymbol{x}_i, \ldots\}$
$\hat{\boldsymbol{x}}$	a quantity derived from, or otherwise related to $\boldsymbol{x}$
$\tilde{\boldsymbol{x}}$	initial guess
$\mathbf{x}^*$	optimal value of $\boldsymbol{x}$
$\boldsymbol{x}'$	a value in the neighborhood of $\boldsymbol{x}$
$\Delta \boldsymbol{x}$	a small step away from $\boldsymbol{x}$
$\Delta_{\boldsymbol{xy}}$	the difference between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e., $\boldsymbol{x} - \boldsymbol{y}$
$\oplus, \ominus$	operators for motion composition and difference
$:=$	definition

Chapter 2

Background

In this chapter, we discuss probabilistic estimation techniques, which are well established in robotics and related fields, and which we apply in this thesis. The adoption of probabilistic models is crucial for inferring information from noisy sensor data. To statistically model real-world data, i.e., data that has not been carefully screened and preprocessed by experts, we need to take care how we choose the model. Particularly we need to consider the explicit and implicit assumptions that are inherent to a model or can be otherwise incorporated into it. On the one hand there are models with convenient statistical and computational properties but strong assumptions about the data. Such models, e.g., parametric models as presented in Sections 2.2 and 2.3 are a good choice as long as the data meets the underlying assumptions. In this case, the parametric model itself provides prior information which speeds up the estimation process. On the other hand, if such a model cannot be conceived beforehand, or the data deviates from the known parametric models, we need to take measures that relax the assumptions, or drop them completely. For example, Section 2.3.3 introduces a non-parametric regression approach, which is based only on few assumptions about the data and mostly relies on the data itself to make estimates. However, the drawback of techniques with strong reliance on the data is the need for more training observations before reliable predictions can be performed.

In Section 2.4 we look into approaches that allow us to use parametric models even if the data deviates from the specific model, or only a part of the data fulfills the assumptions.

2.1 Probabilistic Estimation

Many tasks in robotics involve problems of a statistical nature. Robots perceive the world through sensors and act on it through actuators. These processes are inherently subject to noise; dealing with uncertainty is therefore fundamental for many algorithms in robotics. Probabilistic methods allow us to model the statistical nature of the robot's measurements as well as the uncertainty about the state of the world (including the robot itself).

As a typical example of probabilistic estimation consider the inference of the probability distribution $p(\boldsymbol{x}|\boldsymbol{z})$. Let the random variable $\boldsymbol{x}$ represent the position of an obstacle relative to the robot, and the variable $\boldsymbol{z}$ denote a distance measurement, e.g., from a range sensor. In the following we will refer to the former as the *state* and to the later as *observation*. While we would like to infer the state from the observation, we often cannot directly assess the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{z})$. However, *Bayes rule* allows us to rewrite the estimation problem as

$$p(\boldsymbol{x}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}, \boldsymbol{x})}{p(\boldsymbol{z})} = \frac{p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{z})}. \tag{2.1}$$

This allows us to express $p(\boldsymbol{x}|\boldsymbol{z})$, the probability of the state given the observation, in terms of the *data likelihood* $p(\boldsymbol{z}|\boldsymbol{x})$, the *prior* for the state $p(\boldsymbol{x})$, and the prior for the observation $p(\boldsymbol{z})$.

2.1.1 Maximum a Posteriori

In many applications it is not necessary to estimate the full distribution $p(\boldsymbol{x}|\boldsymbol{z})$, e.g., when we are only interested in the state $\boldsymbol{x}^*$ with the highest probability density for the observation $\boldsymbol{z}$, i.e.,

$$\boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{z}) \tag{2.2}$$

$$= \operatorname*{argmax}_{\boldsymbol{x}} \frac{p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{z})}, \tag{2.3}$$

where we apply Bayes rule (Equation 2.1) to rewrite the right hand side. Often, the probability of the observation $p(\boldsymbol{z})$ can only be computed as the marginal distribution of the numerator, i.e.,

$$\int p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{2.4}$$

which is infeasible to compute for a complex state space. In this case, computing only the state with the highest probability has a crucial advantage, because it allows to simplify Equation 2.3 to

$$\boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x}} p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x}), \tag{2.5}$$

where we dropped the denominator $p(\boldsymbol{z})$ as it is independent of $\boldsymbol{x}$. This is called the *maximum a posteriori* solution of $p(\boldsymbol{x}|\boldsymbol{z})$. The name stems from the fact that commonly $p(\boldsymbol{x})$ represents our belief about the state $\boldsymbol{x}$ before we obtain the observation $\boldsymbol{z}$ and is

therefore called *prior*, whereas the right hand side of Equation 2.5 denotes the maximum of $p(\boldsymbol{x}|\boldsymbol{z})$ after observing $\boldsymbol{z}$.

While applicable to many problems, note that the reduction to the maximum may turn out problematic, e.g., in situations where the maximum does not represent the distribution sufficiently well [83], which is often a problem in high dimensional spaces.

2.1.2 Maximum Likelihood

If we do not have any prior belief for the state $\boldsymbol{x}$, we may apply the principle of indifference and assume that every state of $\boldsymbol{x}$ is equally likely and hence $p(\boldsymbol{x})$ is constant. Thus, $p(\boldsymbol{x})$ is a constant independent of $\boldsymbol{x}$ and can therefore be dropped. This leaves us with

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{z}|\boldsymbol{x}), \tag{2.6}$$

which is called the *maximum likelihood* solution, because it only depends on the likelihood of the data $p(\boldsymbol{z}|\boldsymbol{x})$.

Assuming $p(\boldsymbol{x})$ to be constant leads to an improper prior [9] for probability densities on an infinite domain, where the uniform distribution is not properly defined. However, since the term is dropped this is not a problem in the presented case.

As for maximum a posteriori, this solution is widely applicable, yet comes with further limitations. A common problem arising from the non-informative prior is overfitting the parameters to the specific observations. This is often the case when the amount of observations does not sufficiently constrain the degrees of freedom of the state.

2.2 Least Squares

The method of least squares is ubiquitous in modern statistical analysis and is tightly connected with the presented probabilistic approach. In this thesis, it has been directly applied to solve several estimation problems and is the core component of graph-based SLAM solvers.

2.2.1 Maximizing Probabilities by Error Minimization

Under certain conditions, maximizing the data likelihood or the "a posteriori" distribution is equivalent to minimizing a sum of squared error terms. Consider the basic problem of estimating the state of the world $\boldsymbol{x}$ from a number of sensor measurements $\boldsymbol{z}$, as introduced above.

Consider a model $\boldsymbol{z}_t = \boldsymbol{h}_t(\boldsymbol{x})$ that relates the variable $\boldsymbol{x} \in \mathbb{R}^n$ to a dependent random variable $\boldsymbol{z}_t \in \mathbb{R}^m$. In the state estimation problem, $\boldsymbol{h}_t(\boldsymbol{x})$ represents the physical princi-

ples underlying the measurement of $\boldsymbol{z}_t$ given the state $\boldsymbol{x}$. For time series data, $\boldsymbol{h}_t(\boldsymbol{x})$ may compute the observation at time t, given a time-dependent model, parameterized with $\boldsymbol{x}$. However, the index t is not necessarily related to time. It identifies a measurement and thus may be used to index a specific pixel of a camera image, or a beam from a laser range finder.

Assume a model $\boldsymbol{h}_t(\boldsymbol{x})$ that models sensor observations $\boldsymbol{z}_t$ with added Gaussian noise. For a dataset of N measurements $\mathcal{D} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$, we then compute the probability density of the data as

$$p(\mathcal{D}|\boldsymbol{x}) = \prod_{t=1}^{N} \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{h}_t(\boldsymbol{x}), \boldsymbol{\Lambda}_t^{-1}). \tag{2.7}$$

The maximum likelihood solution $\boldsymbol{x}^*$ is therefore computed as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_{t=1}^{N} \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{h}_t(\boldsymbol{x}), \boldsymbol{\Lambda}_t^{-1}) \tag{2.8}$$

$$= \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_{t=1}^{N} \nu_t \ \exp(-\tfrac{1}{2}(\boldsymbol{z} - \boldsymbol{h}_t(\boldsymbol{x}))^T \boldsymbol{\Lambda}_t \, (\boldsymbol{z} - \boldsymbol{h}_t(\boldsymbol{x}))), \tag{2.9}$$

where ν_t is the normalizing constant of the tth Gaussian distribution, which is independent of $\boldsymbol{x}$ and can hence be omitted in the following. Further, maximizing the logarithm of a function leads to the same result as maximization of the function itself [9]. This allows us to simplify Equation 2.9 to

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \ -\frac{1}{2} \sum_{t=1}^{N} (\boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x}))^T \boldsymbol{\Lambda}_t \, (\boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x})) \tag{2.10}$$

$$= \underset{\boldsymbol{x}}{\operatorname{argmin}} \quad \sum_{t=1}^{N} (\boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x}))^T \boldsymbol{\Lambda}_t \, (\boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x})). \tag{2.11}$$

We therefore see, that maximizing the likelihood corresponds to minimizing the sum of the Mahalanobis distance between the actual and the predicted measurements. Defining the difference between prediction and observation as

$$\mathbf{e}_t := \mathbf{e}_t(\boldsymbol{x}) := \boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x}) \tag{2.12}$$

we obtain the more succinct notation

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{t=1}^{N} \mathbf{e}_t^T \boldsymbol{\Lambda}_t \, \mathbf{e}_t \tag{2.13}$$

Notice that the precision matrices $\boldsymbol{\Lambda}_t$ are symmetric and positive semi-definite. To show that maximum likelihood estimation – under the assumption of uncorrelated Gaussian noise – can be phrased as a least squares problem we redefine the error as

$$\hat{\mathbf{e}}_t = \boldsymbol{L}_t^T \mathbf{e}_t = \boldsymbol{L}_t^T (\boldsymbol{z}_t - \boldsymbol{h}_t(\boldsymbol{x})), \tag{2.14}$$

where $\boldsymbol{L}_t$ is obtained using the Cholesky decomposition $\boldsymbol{\Lambda}_t = \boldsymbol{L}_t \boldsymbol{L}_t^T$. This allows to rewrite Equation 2.11 as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{t=1}^{N} \hat{\mathbf{e}}_t^T \hat{\mathbf{e}}_t \tag{2.15}$$

$$= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{t=1}^{N} \sum_{s=1}^{m} \hat{e}_{ts}^2, \tag{2.16}$$

where $\hat{e}_{ts}$ is the sth component of the vector $\hat{\mathbf{e}}_t$.

2.2.2 Linear Least Squares

If the model $\boldsymbol{h}_t(\boldsymbol{x})$ is linear with respect to the parameter vector $\boldsymbol{x}$, the global minimum can be found in closed form. Consider the problem of estimating the state $\boldsymbol{x} \in \mathbb{R}^n$ from N observations $\boldsymbol{z}_t \in \mathbb{R}^m$ with the linear sensor model

$$h_t(\boldsymbol{x}) = \boldsymbol{M}_t \boldsymbol{x} \tag{2.17}$$

with Gaussian noise, such that

$$p(\boldsymbol{z}_t | \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{h}_t(\boldsymbol{x}), \boldsymbol{\Lambda}_t^{-1}). \tag{2.18}$$

This allows us to define the error function $F(\mathcal{D}, \boldsymbol{x})$ as

$$F(\mathcal{D}, \boldsymbol{x}) = \sum_{t=1}^{N} (\boldsymbol{z}_t - \boldsymbol{M}_t \boldsymbol{x})^T \boldsymbol{\Lambda}_t (\boldsymbol{z}_t - \boldsymbol{M}_t \boldsymbol{x}) \tag{2.19}$$

$$= \sum_{t=1}^{N} \boldsymbol{z}_t^T \boldsymbol{\Lambda}_t \boldsymbol{z}_t - 2 \boldsymbol{z}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{M}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t \boldsymbol{x} \tag{2.20}$$

and the derivative is given by

$$\frac{\delta F(\mathcal{D}, \boldsymbol{x})}{\delta \boldsymbol{x}} = \sum_{t=1}^{N} -2 \boldsymbol{z}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t + 2 \boldsymbol{x}^T \boldsymbol{M}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t. \tag{2.21}$$

Setting the derivative to zero yields

$$0 = \sum_{t=1}^{N} -2\boldsymbol{z}_t^T \boldsymbol{M}_t \boldsymbol{\Lambda}_t + 2\boldsymbol{x}^T \boldsymbol{M}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t \tag{2.22}$$

$$\underbrace{\sum_{t=1}^{N} \boldsymbol{z}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t}_{1\times n} = \boldsymbol{x}^T \underbrace{\sum_{t=1}^{N} \boldsymbol{M}_t^T \boldsymbol{\Lambda}_t \boldsymbol{M}_t}_{n\times n}. \tag{2.23}$$

Thus the vector $\boldsymbol{x}^*$ that minimizes the squared residual can be directly computed by solving the above system of linear equations.

2.2.3 Non-Linear Least Squares

In robotic tasks such as SLAM the models $\boldsymbol{h}_t(\boldsymbol{x})$ are often non-linear. E.g., in SLAM a model will contain non-linear terms due to angular components of the pose and observations. To approach these problems, we therefore need to drop the linearity assumption.

In SLAM, a robot navigates through an environment and relates the different poses of its trajectory by sensor measurements. These measurements, e.g., odometry readings, laser range scans or landmark observations, can be used to estimate the relative transformation between two poses. For simplicity, we shall assume in the following discussion that the state $\boldsymbol{x} = [\ldots, \boldsymbol{x}_i^T, \ldots]^T$ contains the poses of the robot at times i, and the measurements to yield constraints that can be expressed by a single model $\boldsymbol{h}_{ij}(\boldsymbol{x})$ describing the rigid motion between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the robot poses at times i and j. Assuming Gaussian noise with covariance $\boldsymbol{\Lambda}_{ij}^{-1}$ on the measurements $\boldsymbol{z}_{ij}$, each constraint is expressed mathematically by a quadratic error term

$$\mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij} \mathbf{e}_{ij}, \tag{2.24}$$

where the definition of $\mathbf{e}_{ij}$ is analogous to Equation 2.12

$$\mathbf{e}_{ij} := \mathbf{e}_{ij}(\boldsymbol{x}) := \boldsymbol{z}_{ij} - \boldsymbol{h}_{ij}(\boldsymbol{x}). \tag{2.25}$$

Similar to Equation 2.13, the error of a state $\boldsymbol{x}$ given the set of measurements $\mathcal{G}$ then follows as

$$F(\mathcal{D}, \boldsymbol{x}) = \sum_{ij \in \mathcal{G}} \mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij}\, \mathbf{e}_{ij}. \tag{2.26}$$

and we seek to find $\boldsymbol{x}^*$, the solution with the minimum error. Given a non-linear model

$\boldsymbol{h}_{ij}(\boldsymbol{x})$, such as

$$\boldsymbol{h}_{ij}(\boldsymbol{x}) = \boldsymbol{x}_j \ominus \boldsymbol{x}_i, \tag{2.27}$$

where $\ominus$ denotes the difference operator for rigid motions in the plane or in space, we can not compute the global minimum in closed form. Therefore, to find the solution, iterative methods such as Gauss-Newton or Levenberg-Marquardt are used. Both methods linearize the model at a state $\boldsymbol{x}$ to find the local gradient and compute a new state with lower error. The state $\tilde{\boldsymbol{x}}$ used for the first iteration is called the *initial guess*. Depending on the specific shape of the error function, the choice of the initial guess is often crucial for convergence to the global minimum.

For a non-linear model, we can compute the Jacobian $\boldsymbol{J}_{ij}$ at the estimate as $\tilde{\boldsymbol{x}}$

$$\boldsymbol{J}_{ij} = \left.\frac{\delta \boldsymbol{h}_{ij}(\boldsymbol{x})}{\delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}} \tag{2.28}$$

to obtain the local gradient. The Jacobian $\boldsymbol{J}_{ij}$ is used to project the change in the state space linearly to a change in the sensor space, i.e., the space of $\boldsymbol{z}_{ij}$. Assuming that the linearization sufficiently approximates the function locally, i.e.,

$$\boldsymbol{h}_{ij}(\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}) \approx \boldsymbol{h}_{ij}(\tilde{\boldsymbol{x}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{x} \tag{2.29}$$

we can compute the error at a nearby state $\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}$ by

$$\begin{aligned}
\mathbf{e}_{ij}(\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}) &= \boldsymbol{h}_{ij}(\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}) - \boldsymbol{z}_{ij} && (2.30)\\
&\approx \boldsymbol{h}_{ij}(\tilde{\boldsymbol{x}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{x} - \boldsymbol{z}_{ij} && (2.31)\\
&= \mathbf{e}_{ij} + \boldsymbol{J}_{ij}\Delta\boldsymbol{x} && (2.32)
\end{aligned}$$

We can use this approximation to rewrite the error function in Equation 2.26 to

$$\begin{aligned}
F(\mathcal{D}, \tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}) &= \sum_{ij\in\mathcal{G}} \mathbf{e}_{ij}(\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x})^T \boldsymbol{\Lambda}_{ij}\, \mathbf{e}_{ij}(\tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}) && (2.33)\\
&\approx \sum_{ij\in\mathcal{G}} (\mathbf{e}_{ij}(\tilde{\boldsymbol{x}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{x})^T \boldsymbol{\Lambda}_{ij}\, (\mathbf{e}_{ij}(\tilde{\boldsymbol{x}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{x}) && (2.34)\\
&= \sum_{ij\in\mathcal{G}} \mathbf{e}_{ij}^T\boldsymbol{\Lambda}\mathbf{e}_{ij} + 2\mathbf{e}_{ij}^T\boldsymbol{\Lambda}_{ij}\boldsymbol{J}_{ij}\Delta\boldsymbol{x} + \Delta\boldsymbol{x}^T\boldsymbol{J}_{ij}^T\boldsymbol{\Lambda}_{ij}\,\boldsymbol{J}_{ij}\Delta\boldsymbol{x}. && (2.35)
\end{aligned}$$

This formulation allows us to solve for $\Delta\boldsymbol{x}^*$, the update step that minimizes the error under the linearity assumption. Analogous to the global solution of the linear case, we

take the first derivative with respect to $\Delta \boldsymbol{x}$

$$\frac{\delta F(\mathcal{D}, \tilde{\boldsymbol{x}} + \Delta \boldsymbol{x})}{\delta \Delta \boldsymbol{x}} = 2 \sum_{ij \in \mathcal{G}} \mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij} + 2 \sum_{ij \in \mathcal{G}} \Delta \boldsymbol{x}^T \boldsymbol{J}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij}. \tag{2.36}$$

Setting the derivative to zero yields the following linear system of equations

$$0 = 2 \sum_{ij \in \mathcal{G}} \mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij} + 2 \sum_{ij \in \mathcal{G}} \Delta \boldsymbol{x}^T \boldsymbol{J}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij} \tag{2.37}$$

$$- \sum_{ij \in \mathcal{G}} \mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij} = \sum_{ij \in \mathcal{G}} \Delta \boldsymbol{x}^T \boldsymbol{J}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij} \tag{2.38}$$

$$\underbrace{- \sum_{ij \in \mathcal{G}} \mathbf{e}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij}}_{\boldsymbol{b}^T} = \Delta \boldsymbol{x}^T \underbrace{\sum_{ij \in \mathcal{G}} \boldsymbol{J}_{ij}^T \boldsymbol{\Lambda}_{ij} \boldsymbol{J}_{ij}}_{\boldsymbol{A}^T}, \tag{2.39}$$

In contrast to the solution described by Equation 2.23, the solution to this system of equations will, in general, not be the optimal solution. Due to the linearization, $\boldsymbol{x} + \Delta \boldsymbol{x}$ will not be at the correct minimum. The Gauss-Newton algorithm iterates the above update procedure until the improvement in error is below a threshold. The Levenberg-Marquardt algorithm interpolates the Gauss-Newton algorithm with gradient descent to guarantee convergence.

2.3 Regression

Regression problems form an important subset of probabilistic estimation. A common application of regression methods is curve fitting, where we seek to model a function $z_t = f(t)$ given a set of observations $\mathcal{D} = \{z_1 \ldots, z_N\}$. While the role of t is similar to the index t used in the previous section in the sensor model, t here denotes an *input value*, such as a location in a continuous space and its value is directly used in the computation.

There are many approaches to this problem, the choice of which depends on the available prior knowledge about the properties of the function to estimate. In Sections 2.3.1 and 2.3.2 we discuss *parametric regression*, where we assume that the observations z_t follow a model of a given functional form $f(t, \boldsymbol{x})$, of which we want to estimate the parameterization $\boldsymbol{x}$. In Section 2.3.3 we discuss Gaussian processes, a very versatile non-parametric regression method with only few prior assumptions about the functional form of the model.

2.3.1 Linear Regression

For a particular class of regression problems, the method of least squares can be applied analogously to the state estimation methods described in the previous sections. If the process underlying the observations can be modeled by a linear superposition of *basis functions* $\phi_k(t)$ and we assume the observations to be subject to uncorrelated Gaussian noise, the maximum likelihood solution for the coefficient vector $\boldsymbol{x}$ of the model $f(t, \boldsymbol{x})$ is given by the least squares solution. Note that the basis functions are, in general, chosen to be non-linear with respect to the independent variable t.

When a model with few degrees of freedom is used, i.e., a low number of functions are superposed, the choice of the used basis functions strongly restricts the possible shapes of the fitted curve. This restriction to a certain shape can be interpreted as a particular form of a prior and should be motivated by the properties of the problem.

In this thesis, we apply regression in Chapter 5 to determine the deceleration of a door from friction, by observing its trajectory in form of a dataset $\mathcal{D}$ comprised of timestamped opening angles. Assuming constant friction throughout the trajectory, the underlying process is known from classical mechanics to take the form of a second order polynomial with respect to time. Using polynomial basis functions to estimate the function, the parametric model takes the general form

$$f(t, \boldsymbol{x}) = \sum\nolimits_{k=0}^{d} \phi_k(t) x_k = \boldsymbol{\phi}_t^T \boldsymbol{x} \tag{2.40}$$

where d denotes the degree of the polynomial and the basis functions are defined as

$$\phi_k(t) = t^k. \tag{2.41}$$

To avoid a cluttered notation, we defined a vector of the basis functions

$$\boldsymbol{\phi}_t^T = [t^d \ \dots \ t^0]. \tag{2.42}$$

Assuming uncorrelated Gaussian noise with zero mean and precision λ_t on each measurement, the data likelihood under the above parametric model is given by

$$p(z_t|\boldsymbol{x}) = \mathcal{N}(z_t; f(t, \boldsymbol{x}), \lambda_t^{-1}). \tag{2.43}$$

This is a univariate form of Equation 2.18 and therefore the maximum likelihood solution is – analogous to Equation 2.23 – found by solving a system of linear equations of the form

$$\underbrace{\sum\nolimits_t z_t \lambda_t \boldsymbol{\phi}_t^T}_{1\times d} = \boldsymbol{x}^T \underbrace{\left(\sum\nolimits_t \lambda_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T\right)}_{d\times d}. \tag{2.44}$$

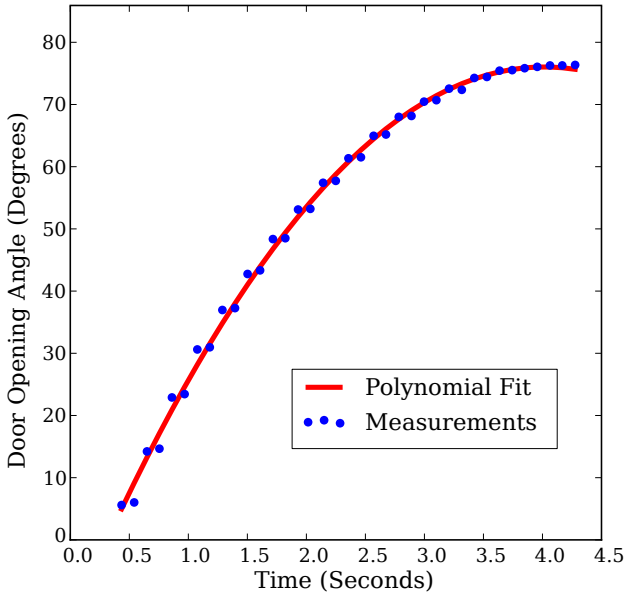


Figure 2.1: Estimation of the trajectory of a released door by using a parametric model for linear regression of a second order polynomial. The chosen parametric model is highly robust to the apparent systematic "stepping" error in the sensor data.

Under the assumption of homoscedasticity, i.e., the noise is independent of t, the precision λ_t is constant and cancels out, yielding

$$\underbrace{\sum\nolimits_t z_t \boldsymbol{\phi}_t^T}_{1\times d} = \boldsymbol{x}^T \underbrace{\left(\sum\nolimits_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T\right)}_{d\times d}. \tag{2.45}$$

Figure 2.1 shows a measured door trajectory and the corresponding estimated second-order polynomial. Note, that the linear regression approach is not limited to problems for which the functional form is known. Given a sufficient degree, fitting a linear combination of polynomials, or other basis functions, can be used to approximate arbitrary functions. However, care has to be taken to not fit a complex model to few observations, as the flexibility of the model will lead to a curve that fits the noise on the observations - a problem referred to as *overfitting*.

A further limitation of parametric regression is the extrapolation behavior, i.e., the shape of the fitted function beyond the limits of the training dataset. As mentioned before, the shapes which the curve can assume depends on the basis functions. The least squares solution will superposition the individual functions $\phi_k(t)$, such that the error on the training dataset is minimized and the shape of the curve approximates the dataset. Going beyond the dataset in the input space, there are no further constraints on the shape of the curve. The shape of the curve is therefore only collaterally defined by the training data and choice of basis functions, and therefore quickly loses any relation to the observed

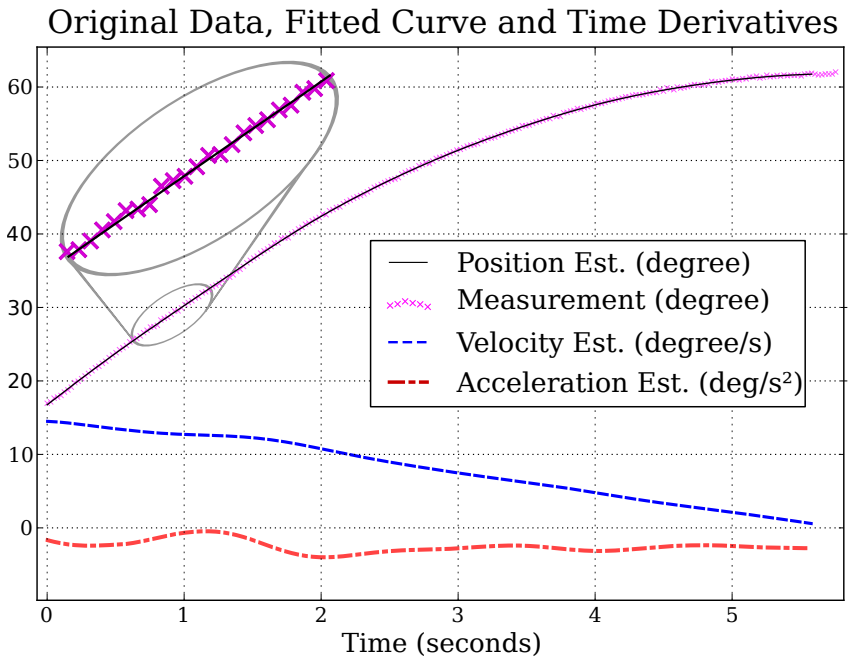


Figure 2.2: Locally weighted regression (thin black curve) of measured angles (magenta crosses) of a door opening demonstration. The dashed blue and red curves correspond to the first and second derivative of the black curve, and thus constitute the estimations of angular velocity and acceleration. Even though we fitted local polynomials of second order for this plot, this method is able to capture the variation in deceleration, as apparent by the red curve.

data. Consider the door arrested by friction shown in Figure 2.1. After stopping, the door stays at its final opening angle, i.e. the curve should become flat. Yet, the fitted second order polynomial describes a parabola. Therefore, when extrapolating beyond the arresting point, it necessarily indicates a re-acceleration in the direction of its original position.

2.3.2 Locally Weighted Regression

For the task of fitting a curve of known parametric form but unknown coefficients, the regression approach presented in the previous section is highly efficient and robust to noise. However, in our motivation of the polynomial basis functions, for fitting the trajectory of a released door, we assumed a constant deceleration from friction. In our experiments we found that this assumption does not hold in practice. In our application, we are mainly interested in learning the deceleration to make predictions about the trajectory and therefore require a regression approach that lets us capture this variation.

A straight-forward solution would be to apply the approach from the previous section using a model with a much higher number of basis functions. However, we would

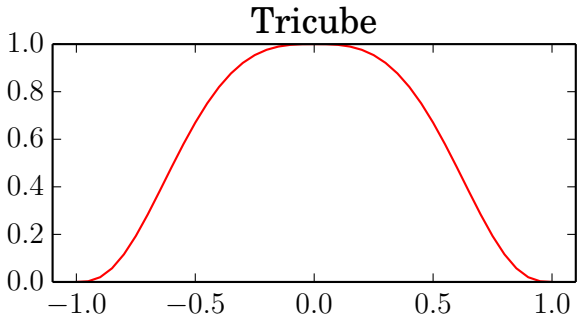


Figure 2.3: The tricube weighting function with length scale 1.

need to determine a degree of freedom which provides a good trade-off between accurate representation of the deceleration and overfitting to noise. Further, while fitting of a second-order polynomial was directly motivated by the underlying mechanics, it is unclear, whether the change in deceleration is approximated well by a polynomial, i.e., why the polynomial form would be a good prior for the variation in the deceleration.

A solution with a better justification is the use of locally weighted regression [19], where we reduce the assumption of constant deceleration to a small local neighborhood. Therefore, instead of solving Equation 2.45 for the whole trajectory, we compute a set of local solutions in the range of interest. To compute such a local solution around $t = t'$, we introduce the weighting function $w_{t'}(t)$ to the error function

$$F_{t'}(\boldsymbol{x}) = \sum\nolimits_t w_{t'}(t)(z_t - \boldsymbol{\phi}_t^T \boldsymbol{x})^2. \tag{2.46}$$

such that only a small (time-)neighborhood influences the solution. In contrast to the precision λ_t in Equation 2.44, the value of $w_{t'}(t)$ for a specific t depends on t'. The weights are computed using a weighting function centered at t'. A common choice is the tricube kernel [19] shown in Figure 2.3. It is defined by

$$\Delta_t = \frac{|t' - t|}{l} \tag{2.47}$$

$$w_{t'}(t) = (1 - \Delta_t^3)^3, \qquad \text{for} |\Delta_t| < 1 \tag{2.48}$$

$$w_{t'}(t) = 0, \qquad \text{for} |\Delta_t| > 1. \tag{2.49}$$

Here, l is a length scale, that defines in which distance to t' the function vanishes. It thus sets the trade-off between locality and insensitivity to noise. In contrast to the use of a higher degree of freedom for the parametric model, the choice of this parameter has a tight connection to the measurement process. It should be set depending on the density, local variation and accuracy of the observations. For irregular data sampling it is also

possible to use a fixed number of neighboring observations and adapt the length scale accordingly.

To determine the local least squares solution at t', we solve

$$\underbrace{\sum\nolimits_t z_t w_{t'}(t) \boldsymbol{\phi}_t^T}_{1\times d} = \boldsymbol{x}^T \underbrace{\left(\sum\nolimits_t w_{t'}(t) \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T\right)}_{d\times d}. \tag{2.50}$$

Figure 2.2 shows a door trajectory and the approximate estimations for velocity and deceleration. It is clearly visible that the deceleration varies substantially. Computing local solutions to generate the profile of the deceleration comes at a cost: There is no analytical solution to the deceleration profile, which means we need to apply numerical methods, e.g., for integration.

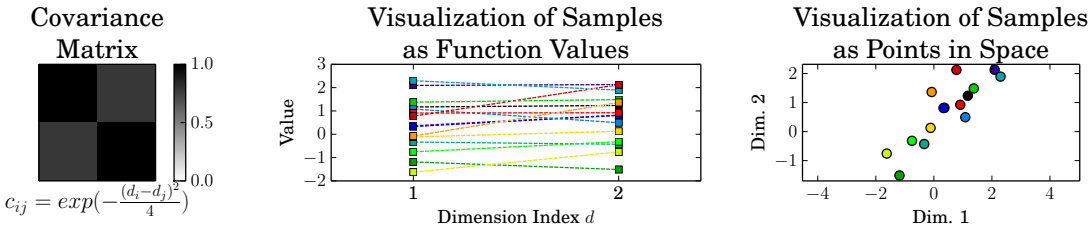
2.3.3 Non-Parametric Regression

In general, if prior knowledge about the process underlying the data is known, it is beneficial to exploit it. In the previous sections, we exploited the fact that the trajectory of an object follows the equations of motion, which relate the observed positions to velocities and accelerations. In cases where no prior information about the specific shape of the process is available, it is often more appropriate not to restrict the shape by the selection of a predefined set of basis functions.

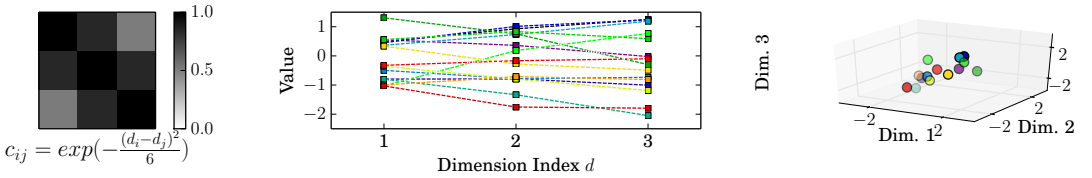
In this section we discuss Gaussian processes (GPs) [102], a non-parametric approach to regression. In contrast to parametric regression, where we only use the learned coefficients and the basis functions to compute the prediction, the prediction of a GP at an input location t is computed from the training data itself.

A Gaussian process represents a Gaussian distribution over functions. Consider first a regular multivariate Gaussian distribution, which is a distribution over a space of finite dimensions. It is defined by a mean $\boldsymbol{\mu}$ and a covariance matrix Σ. The entries of Σ define the relationship for each pair of dimensions by the covariance of their values. With the mean and covariance matrix, we can draw samples, i.e., vectors of the given dimensionality, which follow the distribution. An important property of Gaussian distributions is, that we can compute the conditional distribution in closed form, i.e., given values for a subset of the dimensions, we can compute the distribution of the remaining dimensions. Further, the marginal distribution of a Gaussian distribution is readily available, by restricting $\boldsymbol{\mu}$ and Σ to the elements not related to the marginalized dimensions. This last property allows us to work in spaces of arbitrary (including infinite) dimensionality, by evaluating only a subset of the dimensions.

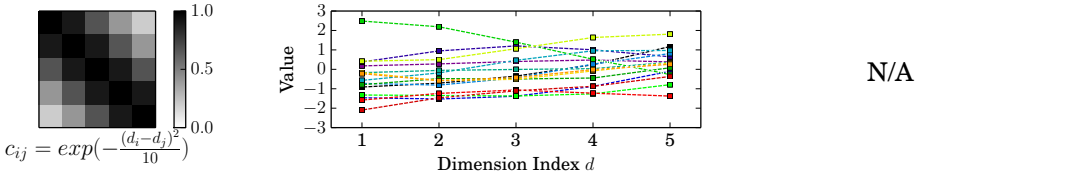
A function $z = f(t)$ with $t \in \mathbb{R}$ can be interpreted as a vector of infinite dimensionality $\boldsymbol{z} = [\ldots, f(t), \ldots]^T$. Each dimension of this vector corresponds to a value of the continuous independent variable t. A GP represents an infinite dimensional Gaussian dis-



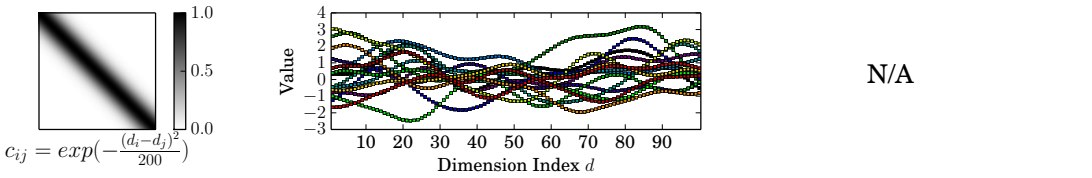
(a) Two dimensional samples from a Gaussian process with zero mean.



(b) Three dimensional samples from a Gaussian process with zero mean.



(c) Five dimensional samples from a Gaussian process with zero mean.



(d) 100 dimensional samples from a Gaussian process with zero mean.

Figure 2.4: Illustration of the function vs. vector space view of the samples drawn from Gaussian processes. The center and right columns show samples drawn from a zero-mean Gaussian with the covariance matrix shown in the left column. The covariance matrix is constructed using a function of the input locations and can be used to create matrices of arbitrary size. Here, we use the squared exponential covariance function, which places a higher probability to smoother functions because it assigns high covariances to nearby dimensions (i.e., input locations).

tribution and thus corresponds to a distribution over functions. As for a regular Gaussian distribution, a GP can be sampled. Instead of vectors of a finite dimension, a sample of a GP represents a function. In practice, of course, we cannot draw a sample of infinite dimensionality, just as we cannot evaluate a given function at every input location. However, as for functions, we can compute an arbitrarily dense subset of values. Figure 2.4 gives an intuition for the interpretation of functions as vectors and illustrates the concept of sampling vectors of increasing (but finite) dimensionality from a Gaussian process.

Analogous to a Gaussian distribution a GP is specified by the mean function $\mu(t)$ and the covariance function $k(t, t')$. The mean function can take any form and is usually set to the training sample mean (which then needs to be subtracted from the samples). The covariance function defines the entries of the (infinitely large) covariance matrix. We can therefore write

$$f(t) \sim \mathcal{GP}(\mu(t), k(t, t')) \tag{2.51}$$

The covariance function defines how the values z_t and $z_{t'}$ covariate, based on the input locations t and t'. In this work we use the commonly used squared exponential covariance function (cf. [102], Eqn. 2.31). For a univariate input location t it is defined by

$$k_{SE}(t, t') = \sigma_f^2 exp(-\frac{(t - t')^2}{2l^2}) \tag{2.52}$$

where σ_f is a hyperparameter affecting the amplitude of the sampled functions and l is the length scale, a hyperparameter that affects the smoothness of the samples. The effect of different length scales is illustrated in Figure 2.5.

To use Gaussian processes for regression, we want to predict the values $\boldsymbol{z}^*$ at the input locations $\boldsymbol{t}^*$, given a vector of N observations $\boldsymbol{z}$ at the respective input locations $\boldsymbol{t}$. The joint distribution of observations and predictions is given by

$$\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{z}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{t}) \\ \boldsymbol{\mu}(\boldsymbol{t}^*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) & \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}^*) \\ \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}) & \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}^*) \end{bmatrix}\right), \tag{2.53}$$

where $\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{b})$ is a matrix whose elements k_{ij} are given by the covariance values $k(a_i, b_j)$. For observations with uncorrelated zero-mean Gaussian noise with standard deviation σ_n, we need to add the variance to the diagonal of $\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t})$, such that the joint distribution becomes

$$\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{z}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{t}) \\ \boldsymbol{\mu}(\boldsymbol{t}^*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma_n^2 I & \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}^*) \\ \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}) & \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}^*) \end{bmatrix}\right). \tag{2.54}$$

The predicted values can be obtained as the expected values of the conditional distribution

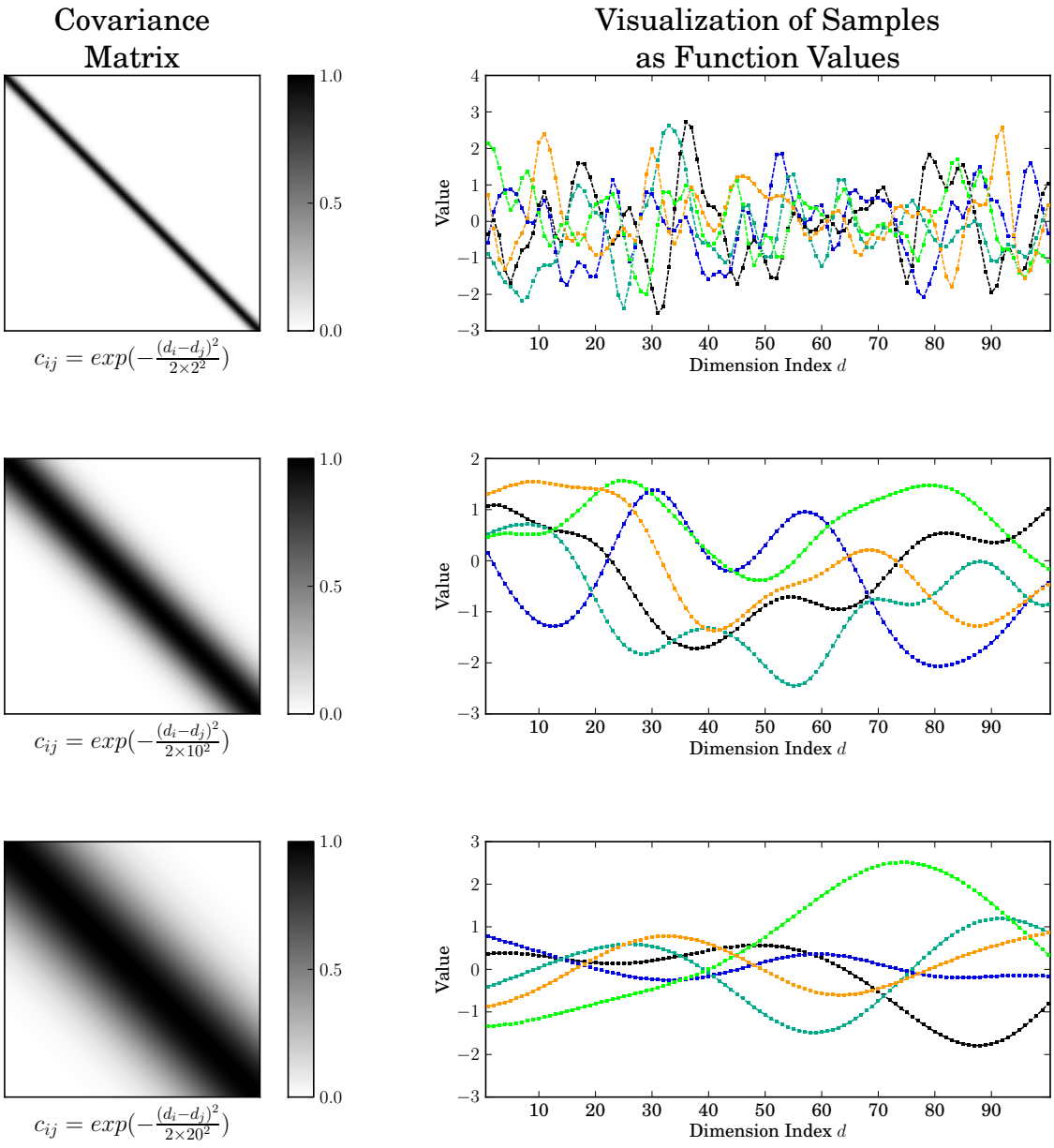


Figure 2.5: Illustration of functions sampled from Gaussian processes with different length scale parameter. Top: $l = 2$, middle: $l = 10$, bottom: $l = 20$.

$p(\boldsymbol{z}^*|\boldsymbol{z}, \boldsymbol{t}, \boldsymbol{t}^*)$, which is computed as

$$\mathbb{E}(\boldsymbol{z}^*) = \boldsymbol{\mu}(\boldsymbol{t}^*) + \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}) \left(\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma_n^2 I\right)^{-1} \boldsymbol{z}. \tag{2.55}$$

The matrix inversion in this equation is the main computational burden of using Gaussian processes and is in $O(N^3)$. Fortunately, it only needs to be updated on new training data. The term $(\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma_n^2 I)^{-1}\boldsymbol{z}$ can thus be precomputed after the learning phase, such that the online runtime of predictions only grows linearly with the training data size.

It is interesting to note the relationship of Gaussian processes to the simpler non-parametric approach of kernel regression [9, p. 301ff.]. In kernel regression, the prediction step that corresponds to Equation 2.55 is given by

$$\mathbb{E}(\boldsymbol{z}^*) = \boldsymbol{\mu}(\boldsymbol{t}^*) + \boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t}) \ \mathrm{diag}(\boldsymbol{\eta})^{-1} \ \boldsymbol{z}. \tag{2.56}$$

Here,

$$\eta_i = \sum\nolimits_{j=1}^{N} \boldsymbol{K}_{ij}(\boldsymbol{t}^*, \boldsymbol{t}). \tag{2.57}$$

are the normalization factors. For each prediction value $\boldsymbol{z}_i^*$, η_i^{-1} normalizes the weights of one row of $\boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t})$. Comparing Equation 2.55 and Equation 2.56, we now clearly see that the difference between GPs and kernel regression lies in this rescaling of the weights. In kernel regression, the weights $\boldsymbol{K}_{ij}(\boldsymbol{t}^*, \boldsymbol{t})$ that define the influence of the individual training values $\boldsymbol{z}_i$ on the prediction of $\boldsymbol{z}^*$ are divided row-wise by a scalar, such that the overall sum per predicted dimension is one - a regular normalization.

In contrast, for the GP these weights are multiplied with the inverse covariance matrix, which encodes the covariance among the training data. This is particularly important if the training data is not uniformly distributed in the neighborhood of the location $\boldsymbol{t}^*$.

To make this distinction clear, consider a case where ten training observations of value one have been observed at the same location. An eleventh observation with the value of zero is a short distance away. If we use kernel regression to predict the value at a point equidistant between these two clusters, the influence of the first cluster will be about ten times the influence of the second cluster, and the predicted value will be close to one. In a GP, however, the prediction will be much closer to 0.5. This is more in line with the observation process, where we would expect the first ten samples to observe the same underlying value and therefore be highly correlated. This correlation is taken into account with a GP. The prediction with the GP does tend slightly towards the first cluster, however, this is because the repeated observation reduces the uncertainty introduced by the observation noise σ_n.

Further differences are manifest in the interpolation and extrapolation behavior. Most kernels only assume positive values. For such a kernel, kernel regression can not predict

a value that is higher (or lower) than the highest (or lowest) training value because of the normalization. As a consequence of this, extrapolation of a monotonically increasing (or decreasing) dataset will at best approach the outermost value asymptotically. For GPs, the rescaled weights $\boldsymbol{K}(\boldsymbol{t}^*, \boldsymbol{t})(\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma_n^2 I)^{-1}$ will in general not sum to one, as the second term is independent of the first term. For areas without nearby training data, the weights of the training data on the prediction will tend to zero, reducing the prediction to the mean term $\boldsymbol{\mu}(\boldsymbol{t}^*)$. On the other hand, the predicted values can easily exceed the range of the training values, allowing the function to follow a trend – the range of such behavior depends on the length scale.

2.4 Robust Estimation Methods

The estimation techniques presented in the previous sections all follow the assumption that all the observed data points follow one type of distribution. In most cases the central limit theorem can be used to motivate the assumption of Gaussian noise on the observations. However, in the real world, there often are several sources of error underlying the distribution of the observed data and only a subset of the data samples contains usable information about the observed quantities, as e.g., in the case of salt-and-pepper noise of camera sensors. The subset of samples that follows the model up to noise is referred to as *inliers*. The remainder of the samples typically are very unlikely when assuming Gaussian distributed noise. They are called *outliers*. If present, even few outliers can have severe effects on the estimation result. Particularly least squares estimators are very susceptible to outliers, as one far-off outlier quickly dominates the sum of errors. Therefore this problem needs to be considered in many estimation problems and has spawned a research field called robust statistics [52] that provides many methods to deal with outliers. This section briefly covers the concepts applied throughout this thesis.

2.4.1 Robust Kernels

Huber [61] introduced the notion of *M-estimators*, short for Maximum likelihood-type estimators, a class of *robust kernel* functions $\rho(e)$ which replace the squared error terms e^2 of a least squares estimator with a function that reduces the impact of high error values. This mitigates the described situation of outliers dominating the sum of errors. According to Huber [61], the goal of using a robust kernel is that small deviation from the assumptions should impair the performance of the estimator only slightly, while larger deviations should not cause catastrophic failure. Concisely, to apply a robust kernel to a sum-of-squares error function, the function

$$F(\boldsymbol{x}) = \sum\nolimits_i e(x_i)^2 \tag{2.58}$$

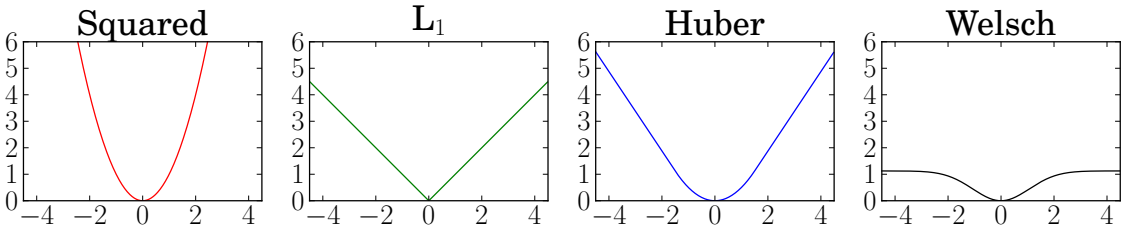


Figure 2.6: Examples for robust kernel functions in comparison to the squared error.

is rewritten to

$$\hat{F}(\boldsymbol{x}) = \sum\nolimits_i \rho(e(x_i)). \tag{2.59}$$

Numerous choices for the kernel function ρ have been proposed in the literature, e.g., the L_1 norm (the absolute value) and the so-called Huber function. Instead of quadratic growth with respect to the residual, these functions grow only linearly for large errors.

The Huber function is quadratic in the vicinity of zero but switches to linear growth at a threshold k. It is defined as

$$\rho(e) = \begin{cases} e^2 & \text{if } |e| \leq k \\ 2k|e| - k^2 & \text{otherwise,} \end{cases} \tag{2.60}$$

where e is the residual error. Such kernels with gentler growth than quadratic make the estimate less sensitive to data that does not follow the Gaussian assumption. Yet the result is still affected by these errors. In cases where outliers carry no information, we would ideally ignore them completely. There are kernels with constant or even decaying value for outliers above a certain threshold. While this is useful for input with drastic violations of the assumed error distribution, it raises the problem of local minima, i.e., the solution is not unique anymore. Figure 2.6 shows these different types of kernel functions in comparison to a regular squared error. Of the various alternatives, the Huber kernel has been reported to be rarely inferior to other choices [131], which has also proven true in preliminary experiments with the RGB-D SLAM approach presented in Chapter 3.

Unlike regular minimization of the squared error, most robust kernels do not have a closed form solution, requiring an iterative optimization, also called iterative reweighted last squares. Since non-linear least squares optimization approaches as discussed in Section 2.2.3 are inherently iterative, the adoption of robust kernels comes at almost no cost in this case. However, the lack of a closed form solution has an important drawback for kernels with a constant maximum for large errors or even decaying error. As for linearization, the value and derivative of the kernel function depends on the estimate. Consider the estimation of the mean of a set of scalars. The kernel function depends on e, in this example the deviation from the mean, which can only be computed given an estimate

of the mean – either from the previous iteration or an initial guess. However, a good initial guess is not trivial to achieve. When using regular least squares to initialize, the outliers will distort the estimate. For kernels with constant maximum, actual inliers may be treated as outliers according to the initial guess and will lose their positive influence on the optimization. For decaying kernels they may even drive the optimization away from the correct solution.

The next section discusses a robust estimation technique which defeats this problem by repeatedly computing an estimate from a small random subset of the data, retaining the best solution as determined by counting inliers.

2.4.2 Random Sample Consensus

Random sample consensus (RANSAC) is an iterative method to make the estimation of model parameters robust to gross errors in the input data. RANSAC was originally proposed by Fischler and Bolles [43] in a similar context as found in Chapter 3 and 4, namely the location determination problem – the problem to estimate the location of the camera based on visual landmarks with known position. When dealing with visual landmarks, small errors may occur, e.g., from imperfect calibration. More problematic, however, is the association of visual landmarks. A wrong association, when incorporated naively into a least squares estimation, generally introduces a drastic error into the final estimate. As mentioned in the previous section, robust kernels may be used to lessen the effect of outliers. However, for problems involving data association errors, the number of outliers and the magnitude of their error are often too high to be dealt with by robust kernels alone. While there are kernels that ignore outliers (by yielding a constant value above a threshold), they require a good initial guess that can be used to distinguish inliers from outliers.

RANSAC solves the problem by using a small, random subset of the data to generate a hypothesis, i.e., an estimate of the model parameters. Even if the dataset contains many outliers, there is a chance that this hypothesis is computed only from inliers. To determine the quality of the solution, the hypothesis is evaluated based on the number of samples of the original set that are in agreement with the hypothesis, typically by computing and thresholding of the residual error with respect to the model estimate. The solution can then be (possibly iteratively) improved by incorporating the inliers of the current hypothesis to compute an updated solution. The whole process is repeated with new initial samples for a given number of iterations. In general, the chance that the model estimation from outliers will be supported by more samples than a model computed from the inliers is marginal, particularly in high dimensions. The hypothesis with the highest support is therefore retained as the final solution. Pseudo-code for RANSAC is given by Algorithm 1. The number of iterations required to find a solution with a certain probability can be computed if the ratio of inliers and outliers is known. Even though this

Algorithm 1 Pseudocode for the RANSAC algorithm.

Input: Dataset $\mathcal{D}$, Threshold t, Iterations i
Output: Model Parameters $\mathcal{P}$

```
n := 0        // Size of current set of inliers
for 1 to i do
    S := RandomSubset(D)
    P̃ := EstimateModelParameters(S)
    E := ComputeErrors(D, P̃)
    I := ComputeInlierSet(D, E, t)
    if Count(I) > n then
        P := EstimateModelParameters(I)
        n := Count(I)
    end if
end for
return P
```

is rarely the case, this method is often used with approximate values.

There are many variants and extensions for the basic algorithm. The improvements can be grouped into approaches for higher accuracy, faster execution and higher robustness. Choi et al. [17] provide a comprehensive overview.

Chapter 3

3D SLAM with an RGB-D Camera

Contents

In this chapter we investigate simultaneous localization and mapping (SLAM) with RGB-D cameras. We develop a system that generates 3D maps using an RGB-D camera using state-of-the-art algorithms and introduce several approaches to increase the accuracy and the robustness. These include a method for exploiting the available pose graph to guide the search for matching features, techniques for improving the performance by online adaptive feature detection, statistical outlier detection in the graph optimization and verification of registration results for structured point sets. Furthermore, we present a benchmark for evaluating RGB-D SLAM systems. We use the benchmark to demonstrate the benefits of the above approaches, to thoroughly evaluate the performance of the final system and to compare it against other approaches. The results of the experiments demonstrate that our system can robustly deal with challenging scenarios such as fast camera motions and feature-poor environments while being fast enough for online operation. We made the software and the benchmark publicly available. Both have already been widely adopted by the robotics community.

. . . . .

The problem of simultaneous localization and mapping (SLAM) is one of the most actively studied problems in the robotics community in the last decade. The availability of a map of the robot's workspace is an important requirement for the autonomous execution of several tasks including localization, motion planning, and collision avoidance. Especially for mobile robots working in complex, dynamic environments, e.g., fulfilling transportation tasks on factory floors or in a hospital, it is important that they can quickly generate (and maintain) a map of their workspace using their on-board sensors. This, however, is considered a challenging chicken-and-egg problem because building the map requires the knowledge of the pose of the robot and finding the pose in turn requires the map of the environment. Therefore, both the poses of the robot and the map need to be estimated at the same time.

Given the recent advances in mobile robotics, there is an increasing demand for efficient solutions that provide three-dimensional metric maps. Manipulation robots, for example, require a detailed model of their workspace for collision-free motion planning and aerial vehicles need detailed maps for localization and navigation. Previously 3D mapping approaches relied on laser scanners or stereo cameras. While 3D laser scanners provide very accurate measurements, they are expensive and heavy, which prohibits

their use for consumer products and aerial vehicles. Stereo cameras require texture to compute the distance measurements from triangulation. However, indoor environments often contain sections without texture, e.g., monochrome walls, ceilings and floors. The commercial launch of mass-produced RGB-D cameras provides an attractive, powerful alternative to laser range scanners and stereo cameras, making 3D perception available at a fraction of the cost. Similar to stereo cameras, the mode of operation of these RGB-D cameras is based on triangulation. However, in contrast to the former, they actively project a pattern in the infrared spectrum and use its perception in an infrared camera for triangulation. Therefore, such an RGB-D camera is independent of the texture in the scene.

In the course of this thesis, we developed one of the first RGB-D SLAM systems that takes advantage of the dense color and depth images provided by RGB-D cameras. We propose several techniques that aim at further increasing the robustness and accuracy of the trajectory estimation. In particular,

- we propose the use of statistical hypothesis testing based on an environment measurement model (EMM) to validate the transformations estimated by feature correspondences or the iterative closest point (ICP) algorithm,
- we identify several key measures that increase the performance of state-of-the-art keypoint detection and matching methods to obtain reliable results in real-time,
- we propose an approach that selects candidate frames to match new frames against, that exploits and extends known loop closures,
- and we introduce a statistical outlier detection system to the graph optimization backend, which further increases the robustness of our system.

Further, we created a benchmark for RGB-D SLAM systems that comprises a huge variety of real-world scenarios, ground truth pose data and two error metrics. We use this benchmark to show that our RGB-D SLAM system allows us to accurately track the robot pose over long trajectories and under challenging circumstances. We provide comprehensive results for the 44 sequences of the benchmark with public ground truth in Appendix A. To allow other researchers to use our software, reproduce the results, and improve on them, we released the presented system and the benchmark as open-source. The code and detailed installation instructions are available online [34, 37, 116].

3.1 Sparse RGB-D SLAM

In general, a graph-based SLAM system can be broken up into three modules [6, 27]: Frontend, backend and map representation. The frontend processes the sensor data to extract geometric relationships, e.g., between the robot and landmarks at different points

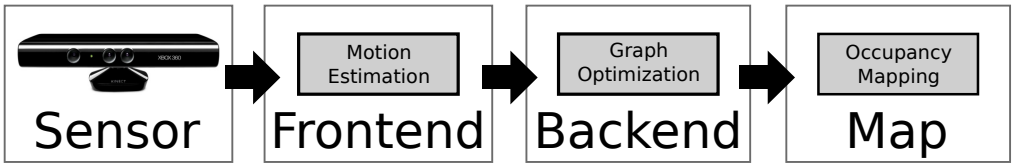


Figure 3.1: Schematic overview of our approach. The frontend processes the sensor data and extracts geometric constraints between the individual RGB-D frames. The backend constructs a pose graph from the constraints and estimates the maximum likelihood solution for the sensor trajectory using non-linear least squares techniques. From the optimized trajectory we finally generate a voxel occupancy map.

in time. The frontend is specific to the sensor type used. Except for sensors that measure the motion itself as, e.g., wheel encoders or IMUs, the robot's motion needs to be computed from a sequence of observations. Depending on the sensor type there are different methods that can be applied to compute the motion in between two observations. In the case of an RGB-D camera the input is an RGB image I_{RGB} and a depth image I_D. We determine the landmarks by extracting a high-dimensional descriptor vector $\boldsymbol{d}$ from I_{RGB} and store them together with their location $\boldsymbol{y} \in \mathbb{R}^3$ relative to the observation pose $\boldsymbol{x} \in \mathbb{R}^6$.

To deal with the inherent uncertainty introduced, e.g., by sensor noise, the backend of the SLAM system constructs a graph that represents the geometric relations and their uncertainties. By optimizing this graph structure we can obtain a maximum likelihood solution for the represented robot trajectory. With the known trajectory we project the sensor data into a common coordinate frame. However, in most applications a task-specific map representation is required, as using sensor data directly would be highly inefficient. We therefore create a three-dimensional probabilistic occupancy map from the RGB-D data, which can be efficiently used for navigation and manipulation tasks. We participated with this initial version of our RGB-D SLAM system in the ROS 3D contest [30] in 2011, and won the first place in the "most useful" category. Shortly after, we published the first system description [41] and an experimental evaluation [33]. A schematic representation of the presented system is shown in Figure 3.1. The following sections describe the illustrated components of the system.

3.1.1 Sensor

Recently introduced RGB-D cameras such as the Microsoft Kinect or the Asus Xtion Pro Live offer a valuable alternative to laser range scanners, the traditional 3D sensors in robotics, as they provide dense, high frequency depth information at a low price, size and weight. The mentioned sensors provide 640×480 pixel RGB color images depth measurements at a frequency of 30 Hz. According to the specifications the latency is about 90 ms and the range of the depth measurements is 0.5 m to 5.0 m. We found the

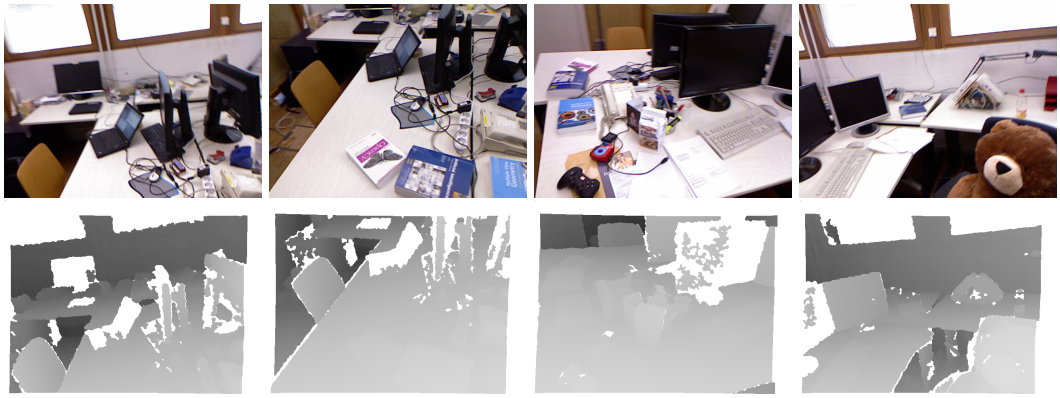


Figure 3.2: Color and depth images as obtained by a Microsoft Kinect. Comparable data is obtained from the Asus Xtion Pro Live. The gray levels represent the distance of the depth image. White corresponds to the absence of a measurement.

Figure 3.3: Color and depth as obtained from the project Tango mobile phone from Google.

depth range to be dependent on the lighting conditions and the reflectivity of the surface. Figure 3.2 shows examples of the obtained depth and color images. To measure the depth, the sensor projects structured light in the infrared spectrum, which is perceived by an infrared camera with a known baseline offset. Due to the active structured light, these sensors are sensitive to illumination. They are therefore generally not applicable in direct sunlight. In contrast, the preceding time-of-flight cameras, e.g., those from the SwissRanger series, are less sensitive to sunlight. However, they have lower resolutions, higher noise, are more difficult to calibrate, and much more expensive.

For the RGB-D cameras, the disparity between the projected pattern and the pattern observed by the infrared camera is used to compute the depth information by triangulation using the known baseline. Because of the pixel discretization, the depth is measured in discretized steps. Unfortunately the step size grows quadratically with the distance to the sensor [71]. For the Microsoft Kinect and Asus Xtion sensors, the depth image then needs to be registered to the color image, to obtain maximum accuracy. However, for many applications, using the per-pixel correspondence between depth and color is sufficient.

For the RGB-D cameras in the recent project Tango prototypes [48], manufactured for Google, the infrared and color sensing is combined on the same chip. They provide data at different resolution and lower frequency but otherwise similar to the above cameras, as illustrated in Figure 3.3. Because the lens and sensor circuit is shared, the user does not need to calibrate the correspondence of depth and color pixels.

In contrast to stereo vision, the depth measurements of RGB-D cameras are independent of the texture of the environment and therefore allow to extract dense RGB-D information. The infrared pattern does not interfere with the extraction of color or intensity based visual features. On the other hand, for more than one RGB-D camera the infrared projections will interfere, if they overlap [86].

In this thesis, we focus on RGB-D cameras based on the described principle of a projected infrared pattern. Nevertheless, while some of the described methods are dependent on certain properties of the sensor (i.e., frame rate, resolution, density, color), many of the described methods also apply to other sensing methods that provide visual and depth information.

3.1.2 SLAM Frontend: Motion Estimation

3.1.2.1 Visual Features

The *frontend* of our SLAM system uses the sensor input in form of landmark positions $\boldsymbol{Y} = \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ to compute geometric relations $\boldsymbol{z}_{ij}$ which allow us to estimate the motion of the robot between the states $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Visual features ease the data association for the landmarks. Their appearance is quantified by a feature descriptor and measures for the similarity of these descriptors can be defined. We then match pairs of the keypoint descriptors $(\boldsymbol{d}_i, \boldsymbol{d}_j)$ by computing their distance in the descriptor space. For the popular feature descriptors SIFT [82] and SURF [7] the commonly used distance metric is the Euclidean distance, i.e., $|\boldsymbol{d}_i - \boldsymbol{d}_j|$. For ORB, being a binary descriptor, the Hamming distance is used. By itself, however, the distance is not a criterion for association as the distance between corresponding descriptors can vary greatly. Due to the high dimensionality of the feature space it is generally not feasible to learn a mapping for a rejection threshold. As proposed by Lowe [82], we resort to the ratio between the distance to the nearest neighbor ($\boldsymbol{d}_{\text{n1}}$) and the distance to the second nearest neighbor ($\boldsymbol{d}_{\text{n2}}$) in feature space. For SIFT and SURF this is

$$r = \frac{|\boldsymbol{d}_i - \boldsymbol{d}_{\text{n1}}|}{|\boldsymbol{d}_i - \boldsymbol{d}_{\text{n2}}|}. \tag{3.1}$$

Under the assumption that a keypoint only fits to exactly one other keypoint in another image, the distance to the second nearest neighbor should be much larger. Thus, a threshold on the ratio between the distances to the nearest and second nearest neighbor can be

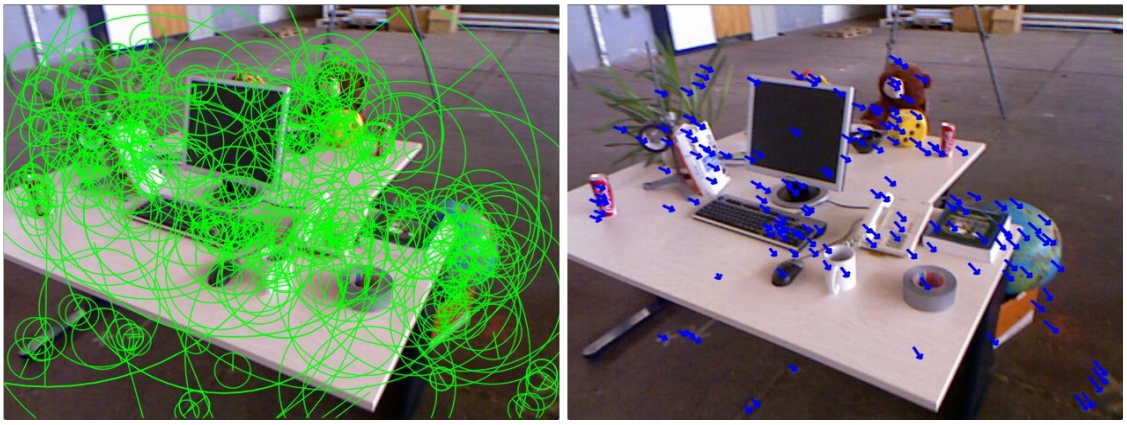


Figure 3.4: Detected keypoints using SIFT (GPU) and sparse optical flow from a previous frame.

used effectively to control the ratio between false negatives and false positives. To make the nearest neighbor search fast, we use the library for fast approximate nearest neighbor search, FLANN [90], as implemented in the OpenCV library [10]. The choice of feature detector and descriptor largely influences the accuracy and runtime performance of our system. We employ the implementations of the OpenCV library in our system, which lets us choose from a wide range of keypoint detectors and feature extractors. We found the respective combinations of detectors and descriptor for SIFT, SURF, and ORB [106] to provide different trade-offs between accuracy and runtime. Where SIFT is the most accurate, ORB is the fastest to compute, and SURF occupies the middle ground in both categories. For SIFT, we further incorporated a GPU-based implementation [127] to reduce the high runtime penalty.

Another influential choice is the number of features extracted per frame. To limit the runtime for extracting and matching the features, we limit the number of keypoints to 600 per frame. A higher limit does not improve the accuracy noticeably. Figure 3.4 shows an example of selected keypoints, and their motion computed from matching to a previous frame.

In Section 3.3 we discuss techniques to improve the performance of the standard OpenCV features.

3.1.2.2 Registration

We use a least-squares estimation method [121] to compute the motion estimate from the established 3D point correspondences. To be robust against false positive matches, we employ RANSAC [43] when estimating the transformation between two frames, which proves to be very effective against individual mismatches. See Section 2.4.2 for a brief description of the RANSAC algorithm. We quickly initialize a transformation estimate from three feature correspondences. The transformation is verified by computing the

inliers using a threshold θ based on the Mahalanobis distance between the corresponding features. For increased robustness in case of largely missing depth values, we also include features without a depth reading into the verification. It has been highly beneficial to recursively re-estimate the transformation. In every recursion step, we reduce the threshold θ for the inlier determination, as proposed by Chum et al. [18]. Combined with a threshold for the minimum number of matching features for a valid estimate, this approach works well in many scenarios.

With increasing size of the mapped area, indoor environments introduce additional challenges. Man-made environments usually contain repetitive structures, e.g., the same type of chair, window or repetitive wallpapers. Given enough similar features through such identical instances the corresponding feature matches between two images result in the estimation of a bogus transformation. The threshold on the minimum number of matches reduces the number of erroneous estimates from random similarities and repetition of objects with few features. However, our experiments show that setting the threshold high enough to exclude estimates from systematic misassociations comes with a performance penalty in scenarios without the mentioned ambiguities. The alternative robustness measures proposed in Section 3.6 and 3.5 are therefore highly beneficial in challenging scenarios.

To take the strongly anisotropic uncertainty of the measurements into account the transformation estimates can be improved by minimizing the squared Mahalanobis distance instead of the squared Euclidean distance between the correspondences. This has been independently investigated by Henry et al. [56] and referred to as *two-frame sparse bundle adjustment*. We implemented this approach by applying g^{2}o (see Section 3.1.3) after the motion estimation. We optimize a small graph consisting only of the two sensor poses and the previously determined inliers. However, in our experiments, this additional optimization step shows only a slight improvement of the overall trajectory estimates. We also investigated including the visual features directly as landmarks in the global graph optimization as it has been applied by other researchers, e.g., Maier et al. [85]. Contrary to our expectations we again observed only minor improvements. As the number of landmarks is much higher than the number of poses, the optimization runtime increases substantially. This method is therefore not feasible for online operation. Recent work in computer vision shows that full bundle adjustment requires clusters of computers [2] or powerful graphics hardware [45] to build larger maps of outdoor scenes within a day.

3.1.2.3 Visual Odometry and Loop Closure Search

Applying an egomotion estimation procedure between consecutive frames provides visual odometry information. However, the individual estimations are noisy, particularly in situations with few features or when most features are far away, or even out of range. Combining several motion estimates by additionally estimating the transformation to

frames other than the direct predecessor substantially increases accuracy and reduces the drift. Successful transformation estimates to much earlier frames, i.e., *loop closures*, may drastically reduce the accumulating error.

To find large loop closures we randomly sample l frames from a set of designated keyframes. The set of keyframes is initialized with the first frame. Any new frame that cannot be matched to the most recent keyframe is added as a keyframe to the set. In this way, the number of frames used for sampling is greatly reduced, while the field of view covered by the keyframes contains most of the perceived area.

3.1.3 SLAM Backend: Graph Optimization

The pairwise transformation estimates between sensor poses, that are computed by the SLAM frontend, form the edges of a pose graph. Due to estimation errors, the edges form no globally consistent trajectory. To compute a globally consistent trajectory we optimize the pose graph using the g^{2}o framework by Kümmerle et al. [77]. The g^{2}o framework performs a minimization of a non-linear error function that can be represented as a graph. Assuming measurements (e.g., the motion estimates) with uncorrelated Gaussian noise, leads to an iterative optimization as detailed in Section 2.2.3. In brief, we minimize a sum-of-squares error function of the form

$$\boldsymbol{F}(\boldsymbol{X}) = \sum_{ij \in \mathcal{C}} e_{ij}^2 \tag{3.2}$$

with

$$e_{ij}^2 = \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})^\top \boldsymbol{\Omega}_{ij}\, \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij}) \tag{3.3}$$

to find the optimal trajectory $\boldsymbol{X}^* = \operatorname{argmin}_{\boldsymbol{X}} \boldsymbol{F}(\boldsymbol{X})$. Here, $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \ \dots \ , \boldsymbol{x}_n^\top)^\top$ comprises the sensor poses we want to estimate. Furthermore, the terms $\boldsymbol{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ represent respectively the mean and the information matrix of a constraint relating the poses $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, i.e., the pairwise transformation computed by the frontend. Finally, $\mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})$ is a vector error function that measures how well the poses $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ satisfy the constraint $\boldsymbol{z}_{ij}$. It is $\mathbf{0}$ when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ perfectly match the constraint, i.e., the difference of the poses exactly matches the estimated transformation.

The optimization is especially beneficial in case of large loop closures, i.e., when revisiting known parts of the map, since the loop closing edges in the graph diminish the accumulated error. Unfortunately, large errors in the motion estimation step can impede the accuracy of large parts of the graph. This is primarily a problem in areas of systematic misassociation of features, e.g., due to repeated occurrences of objects, but may also happen due to systematic sensor noise introduced by the large steps in the discrete depth measurements for distant points. For challenging data where several bogus transformations are found, the trajectory estimate obtained after graph optimization

may be highly distorted. To mitigate this problem, we apply *robust kernels* to the error function. Section 2.4.2 discusses robust kernels in detail. A robust kernel replaces the squared terms of the error function with other M-estimator functions which take the same (non-squared) error as input, but increase slower than the squared error. This effectively changes the Gaussian error assumption to distributions with heavier tails. A variety of such M-estimator functions has been proposed in the literature [131]. In preliminary experiments, we found that the robust kernels *Huber*, *Cauchy* and the recently proposed *dynamic covariance scaling* [1] improve the robustness of the optimization results. The Huber kernel worked best in our setting. Further approaches to increase the robustness of the graph optimization are proposed in Sections 3.5 and 3.6.

Since the main computational cost of optimization lies in solving a system of linear equations, we investigated the effect of the used solver. g^2o provides three solvers, two of which are based on Cholesky decomposition (CHOLMOD, CSparse) and one implements preconditioned conjugate gradient (PCG). CHOLMOD and CSparse are less dependent of the initial guess, than PCG, both in terms of accuracy and computation time. In particular the runtime of PCG drastically decreases given a good initialization. In online operation this is usually given by the previous optimization step, except when large loop closures cause major changes in the shape of the graph. Due to this property, PCG is suggested for online operation. However, for offline optimization the results from CHOLMOD and CSparse are more reliable.

To deal with large-scale problems, we do not optimize the full graph in every time step. Depending on the application, a specific subgraph is selected for optimization.For visual odometry applications, we optimize in each time step only the nodes corresponding to frames for which a motion estimate was successfully computed. Because of the small size of this subgraph, the optimization time is negligible for most settings and can typically be done in real-time. We also use this approach in our offline experiments to achieve a better initial guess for the final optimization. The drawback of this strategy is that the information of loop closures will in general not be propagated through the graph, as the major part of the loop will be held fixed and be ignored respectively. If the map needs to be updated online, but the local optimization does not suffice, we can achieve a better approximation by including the nodes between the oldest frame that was matched since the previous optimization (i.e., to which the largest loop has been closed) and the current frame. This strategy yields almost exact results in most cases and therefore provides a compromise between fast optimization times in the absence of large loop closures and the quality of the map. However, the optimization time is not bounded and may spike for large loop closures.

Figure 3.5: Occupancy voxel representation of the sequence "fr1 desk" with 1 cm^3 voxel size. Occupied voxels are colored for easier viewing.

3.1.4 Map Representation

The system, as described above, computes a globally consistent trajectory. Using this trajectory we can project the original point measurements into a common coordinate frame, thereby creating a point cloud representation of the world. Such models, however, are highly redundant and require vast computational and memory resources, therefore the point clouds are often subsampled, e.g., using a voxel grid.

To overcome the limitations of point cloud representations, we use 3D occupancy grid maps to represent the environment. In our implementation, we use the octree-based volumetric mapping framework OctoMap [58]. The voxels are managed in an efficient tree structure that leads to a compact memory representation and inherently allows for accessing the stored occupancy values at multiple resolutions. The use of probabilistic occupancy estimation furthermore provides a means of coping with noisy measurements and errors in pose estimation. A crucial advantage in contrast to a point-based representation, is the explicit representation of free space and unmapped areas, which is essential for collision avoidance and exploration tasks.

The memory efficient 2.5D representation in a depth image can not be used for storing a complete map. Using an explicit 3D representation, each frame added to a point cloud map requires approximately 3.6 Megabytes in memory. An unfiltered map constructed from an office-sized scene as used in our experiments would require between two and five Gigabytes. In contrast, the corresponding OctoMaps with a resolution of 2 cm ranges from

Figure 3.6: Occupancy voxel map of the PR2 robot. Voxel resolution is 5 mm. Occupied voxels are colored for easier viewing.

only 4.2 to 25 Megabytes. A further reduction to an average of few hundred kilobytes can be achieved if the maps are stored binary (i.e., only “free” vs. “occupied”).

On the downside, the creation of an OctoMap requires more computational resources since every depth measurement is raycasted into the map. The time required fro processing each frame is highly dependent on the voxel size, as the number of traversed voxels per ray increases with the resolution. Raycasting takes about one second per 100,000 points at a voxel size of 5 cm and a maximum range of 4 m on a single core. At a 5 mm resolution, as in Figure 3.6, raycasting a single RGB-D frame took about 25 seconds on an Intel Core i7 CPU with 3.40 GHz. For generating a voxel map online we therefore need to lower the resolution and raycast only a subset of the cloud. In our experiments, using a resolution of 10 cm and a subsampling factor of 16 allowed for 30 Hz updates of the map and resulted in maps suitable for online navigation. However, a voxel map cannot be updated efficiently in case of major corrections of the past trajectory as obtained by large loop closures. Depending on the application it may therefore be reasonable to delay voxelization, recreate the map in case of a disruptive loop closure, or to create local submaps.

3.2 A Benchmark for RGB-D SLAM Approaches

The difficulty of a SLAM problem is highly dependent on the available sensor data. With an RGB-D sensor such as the Microsoft Kinect, under good conditions a very accurate map can be produced using adaptations of existing algorithms. Many approaches have been proposed since the availability of Kinect-style RGB-D cameras that have proven to be capable of accurately mapping an office-sized room. Unfortunately only few approaches were evaluated on datasets of which the difficulty can be judged. In this respect a public dataset and common error metrics to benchmark novel approaches can greatly stimulate the development of novel algorithms. Published results allow the comparison of different approaches and enable scientists to identify and focus on the most promising methods. These datasets and metrics need to be carefully designed, such that the measurements obtained from the benchmark reflect the quality desired in practical applications. There are several benchmark datasets available in the computer vision and robotics community that can be used to demonstrate the performance of SLAM approaches. As low-cost RGB-D sensors based on structured light became available only recently, no such benchmarking dataset has been available. We therefore created such a benchmark, consisting of datasets with ground truth information for the sensor trajectory and several tools to analyze the performance. Furthermore we used the presented RGB-D SLAM system to create a baseline for future comparisons. For all sequences with public ground truth, appendix A.1 details the accuracy that our approach achieves with the techniques proposed in the following sections. We also use this benchmark dataset throughout this chapter, to evaluate the performance impact of the individual contributions. Other researchers have meanwhile adopted the benchmark for the experiments in their publication and reported results. We are therefore able to present a comparison to those approaches in Section 3.7.

In the following we describe the dataset, the proposed error metric and details about the experimental settings used for the evaluations conducted in this thesis.

3.2.1 RGB-D Benchmark Datasets

In the benchmark we provide RGB-D datasets aimed at different application scenarios. The datasets consist of RGB-D sequences captured with two Microsoft Kinects and one Asus Xtion Pro Live sensor. Synchronized ground truth data for the sensor trajectory, captured with a high precision motion capturing system, is available for all sequences. For the evaluations in this chapter, we will use the two datasets aimed at evaluation of SLAM systems, “Handheld SLAM” and “Robot SLAM”. The former consists of eleven sequences, partly recorded in an office building (prefixed with “fr1”) and partly recorded in an industrial hall (prefixed with “fr2”). The sequences of the “Handheld SLAM” category were recorded with a handheld RGB-D camera. The motion of the sensor covers all six degrees of freedom. The sequences include a wide range of challenges. The “fr1”

set of sequences contains, for example, fast motions, motion blur, quickly changing lighting conditions and short-term absence of salient visual features. Overall, however, the scenario is office-sized and rich on features. Figure 3.5 shows a voxel map our system created for the "fr1 desk" sequence. The "fr2" sequences, particularly those of the "Robot SLAM" category, combine many properties that are representative of highly challenging input. Recorded in an industrial hall, the monochrome floor contains few distinctive visual features. Due to the size of the hall and the comparatively short maximum range of the Kinect, the sequences contain stretches with hardly any visual features with depth measurements. Further, occurrence of repeated instances of objects of the same kind can easily lead to faulty associations. Some objects, like cables and tripods, have a very thin structure, such that they are visible in the RGB image, yet do not occur in the depth image, resulting in features with systematically wrong depth information. The repeatedly occurring poles have a spiral pattern that, similar to a barber's pole, suggests a vertical motion when viewed from a different angle. Several sequences also contain short periods of sensor outage. All sequences except one contain loops.

The "Robot SLAM" category consists of four sequences recorded with the foot of the sensor mounted rigidly on a Pioneer 3 robot. The motion of these sequences is therefore roughly restricted to planar motion and mostly steady. However, gaps in the floor and cables make the robot rock jerkily, leading to fast changes of the field of view. Note that, even though the sequences contain wheel odometry and the motion is roughly restricted to a plane, we make no use of any additional information in the presented experiments.

Appendix A.2 lists all sequences of the benchmark with public ground truth, with detailed properties such as the trajectory length and duration. Further information about the dataset can be found on the benchmark's web page [116] and our corresponding publication [117].

3.2.2 Error Metric

The benchmark also provides evaluation tools to compute error metrics given an estimated trajectory. Depending on the application, the quality of a SLAM algorithm manifests itself in different properties. For an online robot navigation task both the accuracy of the map and the localization of the robot are important. If the goal is to obtain the best possible map using a handheld sensor, only the quality of the map is of concern. Unfortunately, accurate ground truth for a 3D map is hard to obtain for real world scenarios. The error metric chosen in this work therefore does not directly assess the quality of the map but focuses on the reconstruction of the trajectory. One may argue that the map error and the trajectory error of a specific dataset depend on the given scene and the definition of the respective error functions. Nevertheless, it is reasonable to assume that the error in the map will, in general, be highly correlated to the error of the trajectory.

The first error metric, called the *relative pose error* (RPE), was proposed by Kümmerle

et al. [76]. The RPE is based on the difference of relative poses between the ground truth and the estimate for given time steps. For a trajectory estimate $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_1 \dots \hat{\boldsymbol{x}}_n\}$ and the corresponding ground truth $\boldsymbol{X}$ it is defined as

$$e_{\text{RPE}}(\hat{\boldsymbol{X}}, \boldsymbol{X}) = \frac{1}{N} \sum_{ij \in \mathcal{T}} trans(\hat{\delta}_{ij} \ominus \delta_{ij})^2 + rot(\hat{\delta}_{ij} \ominus \delta_{ij})^2, \tag{3.4}$$

where $\hat{\delta}_{ij}$ and δ_{ij} denote the relative transformation in time step ij in the estimate and ground truth, i.e., $\hat{\boldsymbol{x}}_j \ominus \hat{\boldsymbol{x}}_i$ and $\boldsymbol{x}_j \ominus \boldsymbol{x}_i$. Here and above, $\ominus$ denotes the motion difference operator. N is the number of relative transformations evaluated. The functions *trans* and *rot* compute (and possibly weight) the rotational and translational part of this difference. To avoid the scale-dependent summation of rotational and translational errors, they can be computed and assessed separately. The advantage of assessing the relative pose differences, particularly if nearby poses are used, is the independence of the order in which the errors are introduced.

Kümmerle et al. leave the question open how the relative transformations, i.e., the set of time steps $\mathcal{T}$, should be chosen in general. They suggest to make the choice dependent on the focus of the experimental evaluation. Choosing a set of adjacent poses emphasizes the local drift, while differencing poses that are further apart highlights the global consistency.

An alternative error measure provided with the benchmark is the absolute trajectory error (ATE), which measures the difference in terms of the absolute offset between corresponding poses. It is defined as

$$e_{\text{ATE}}(\hat{\boldsymbol{X}}, \boldsymbol{X}) = \frac{1}{n} \sum_{i=1}^{n} ||trans(\hat{\boldsymbol{x}}_i) - trans(\boldsymbol{x}_i)||^2, \tag{3.5}$$

i.e., the mean of the squared Euclidean distances between the corresponding poses. To make the error metric independent of the coordinate system in which the trajectories are expressed, the trajectories are aligned in an optimization step that finds the unique minimum of the above squared error between corresponding points (using the same techniques as for the alignment of visual features). The correspondences of poses are established using the timestamps. In contrast to the RPE, the ATE thus emphasizes the global consistency. However, the results of our SLAM system show that the RPE and the ATE are strongly correlated, as visible in Figure 3.7. An advantage of the ATE over the RPE is the intuitive visualization of the error, as shown in Figure 3.8. We chose to use the ATE in our experiments, as it condenses the error into only one number and has a single, unambiguous definition. We take the square root of the mean squared error (RMSE), to obtain a result in meter instead of square meter, which is more intuitive.

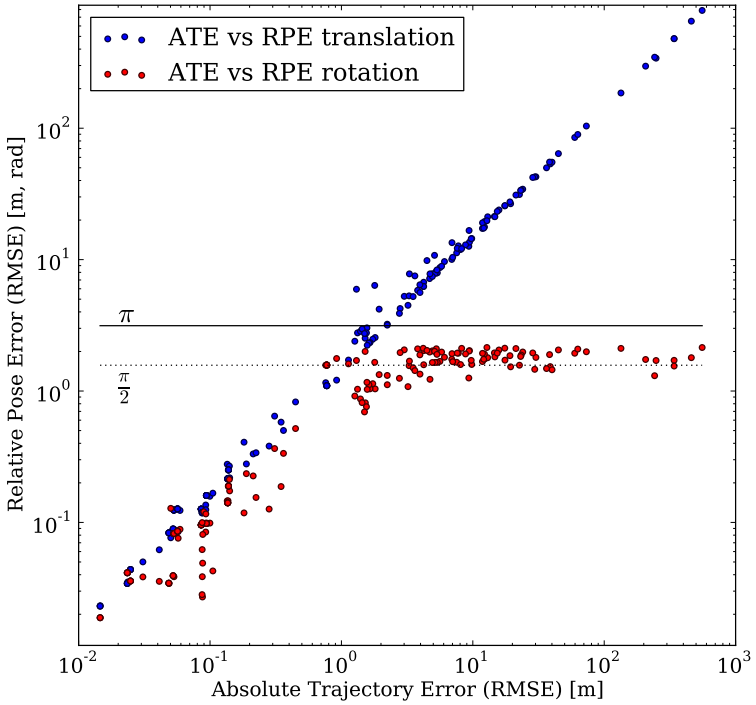
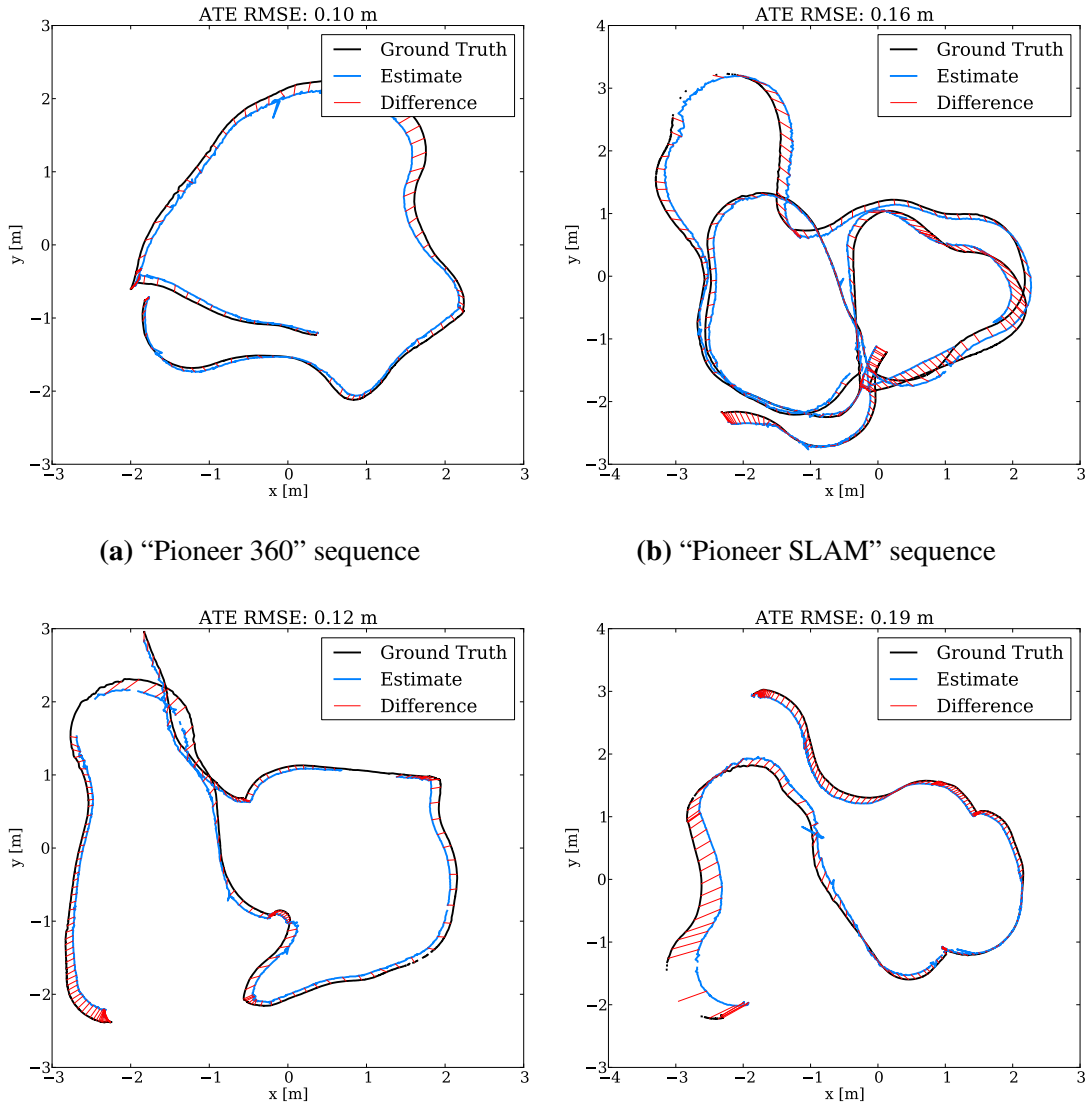


Figure 3.7: Comparison of SLAM error metrics. The strong correlation between the "Absolute Trajectory Error" "and the Relative Pose Error" metrics is clearly visible. The rotational error converges near $\frac{\pi}{2}$, which is the expected average for randomly distributed rotations.

3.2.3 Experimental Setup

For the evaluation of the contributions presented in the following sections of this chapter, we compare the quantitative results when using the proposed techniques to those of the baseline system. If not denoted otherwise, we aggregate the results over the sequences of a dataset (i.e., "Handheld SLAM", "Robot SLAM" or both). The variation of the results is shown by box plots, as the distribution of the error is usually not symmetric. While the matching strategy and RANSAC introduce randomness, the variation due to these factors is low. Since a SLAM system has many parameterized trade-offs, we also often aggregate over various parameter settings. This serves to evaluate the generality of the results. In the course of our work, we often found that certain techniques are beneficial in special settings only. We avoided to incorporate results that only hold for such special cases and discuss this property where applicable.

We used an Intel Core i7 CPU with 3.40 GHz, and an nVidia GeForce GTX 570 graphics card (only used for the GPU implementation of SIFT), for all experiments. Except where otherwise noted, the presented results were obtained offline, processing every recorded frame, which makes the qualitative results independent of the performance



(a) "Pioneer 360" sequence

(b) "Pioneer SLAM" sequence

(c) "Pioneer SLAM2" sequence

(d) "Pioneer SLAM3" sequence

Figure 3.8: The plots show top-views of the ground truth and the estimated trajectory of the "Robot SLAM" dataset. The red lines represent the (non-squared) summands in the absolute trajectory RMSE (Equation 3.5) and thus illustrate the error terms used. Only every tenth difference is shown for clarity.

of the used hardware. However, the presented system has also been successfully used for real-time mapping. The results with such a configuration can be found in Section 3.7, where we compare our approach against other RGB-D SLAM approaches for which a quantitative evaluation on the datasets of the benchmark has been published.

Appendix A states the accuracy achieved with our RGB-D SLAM approach, when combining all the techniques proposed in this chapter. To serve as a useful reference

for performance comparisons, we state the trajectory error for every sequence of the benchmark, as obtained with a single parameterization.

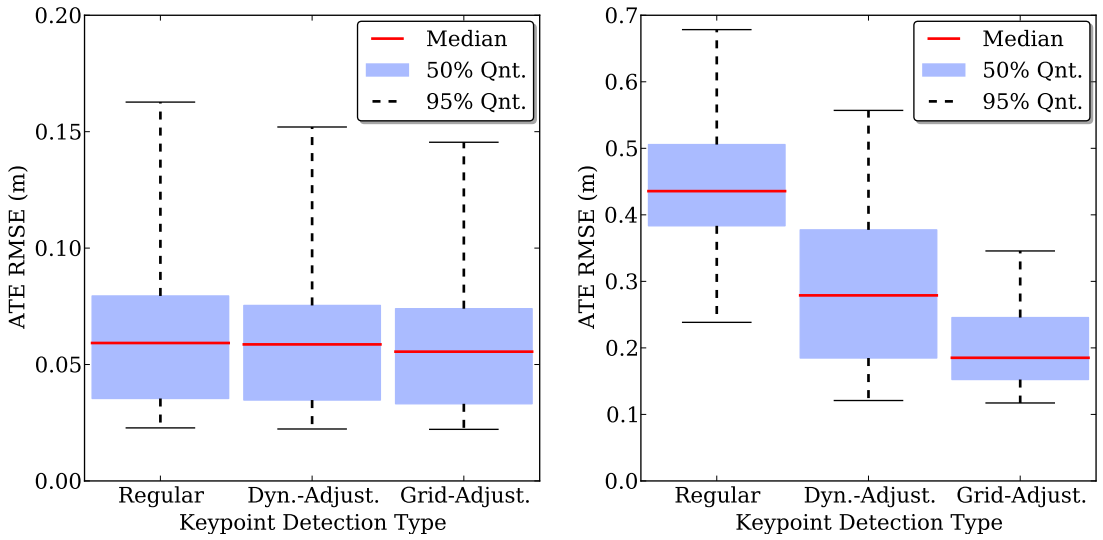
3.3 Improved Feature Detection and Matching

As mentioned in Section 3.1.2.1, the performance of a feature based SLAM system greatly depends on the used algorithms and values of the parameters used for keypoint detection, description and matching. This section discusses techniques to improve the state-of-the-art implementations in the OpenCV library [10].

3.3.1 Keypoint Detection

The keypoint detectors of SIFT, SURF and ORB use a threshold on the saliency of keypoints to limit the number of keypoints computed. Low thresholds lead to high computational cost for feature-rich images. High thresholds lead to a lack of features in images, e.g., with motion blur. OpenCV offers the functionality for dynamically adapting the threshold, but it proved to be too costly for online performance. In our published evaluations [33, 38], we therefore used the default thresholds of the OpenCV implementations. For challenging scenarios, however, we found that these thresholds led to suboptimal results. Huang et al. [60] reported a reduced failure rate for RGB-D odometry using an adaptive threshold for FAST keypoints. We therefore investigated the keypoint detection process and found likewise that substantial improvements could be achieved by adaptive thresholds for sequences with motion blur and changes in lighting, e.g., when the camera is pointed at a window. We further found that the default implementation of the threshold adjuster in OpenCV is not suited for real-time video processing. The threshold adaptation steps are small, so that many iterations are required. The detection is unnecessarily rerun in case of too many features, instead of discarding the excessive keypoints with the weakest response. Moreover, the adjusted threshold is not retained for the next frame. By introducing changes to avoid these inefficiencies, the dynamic adaptation can be used online without major runtime penalty.

To obtain a uniform feature distribution, Huang et al. [60] split the image into patches of 80×80 pixels and retain only the best 25 keypoints per patch. We also divide the input image into a grid (2×2 or 3×3 cells) to avoid the agglomeration of the keypoints, which is mostly a problem for the ORB detector. In addition, we parallelize the keypoint detection of the cells and use independent thresholds for each part. A global threshold always needs to adapt to the cell with the least salient texture, which leads to the costly computation of unnecessary many keypoints in the remaining cells. These additions are particularly important for SURF, where the keypoint detection dominates the overall runtime. To not lose any keypoints, the grid cells need to overlap by the maximum keypoint diameter.



(a) "fr1" Sequences **(b)** "Robot SLAM" dataset

Figure 3.9: Evaluation of the impact of the improved keypoint detection. Left: The benefits are small for the (already very accurate) results of the feature-rich "fr1" scenarios in the "Handheld SLAM" category. Right: For more challenging scenarios, as found in the "Robot SLAM" category, the error is substantially reduced. Results are accumulated over the feature types SIFT (CPU), SURF and ORB.

Again, the basic functionality for the subdivision is available in OpenCV, yet without provisions for the overlap, therefore losing keypoints at the inner borders of the grid, and without the ability of parallel processing or individually adaptive thresholds. As a consequence of these techniques, the trajectory error for the challenging "Robot SLAM" dataset is substantially reduced, as shown in Figure 3.9b. For the feature-rich office scenarios, shown in Figure 3.9a, the effect of these measures is less pronounced but nevertheless visible.

Figure 3.10 shows the improvements per feature type. Particularly the initially worse accuracy of ORB greatly gains on the other feature types and even outperforms them in some sequences.

3.3.2 Feature Matching

Matching non-binary feature descriptions, such as those for SIFT and SURF, is commonly approached by using the Euclidean distance between the feature vectors. Recently, Arandjelović and Zisserman [5] propose to use the Hellinger kernel to compare SIFT features. They report substantial performance improvements for object recognition. In our experiments we could observe improved matching of features, which led to a reduction of the trajectory error of up to 25.8% for individual scenarios. However, for most sequences in the used datasets, the improvement was not significant, as the error is dominated by

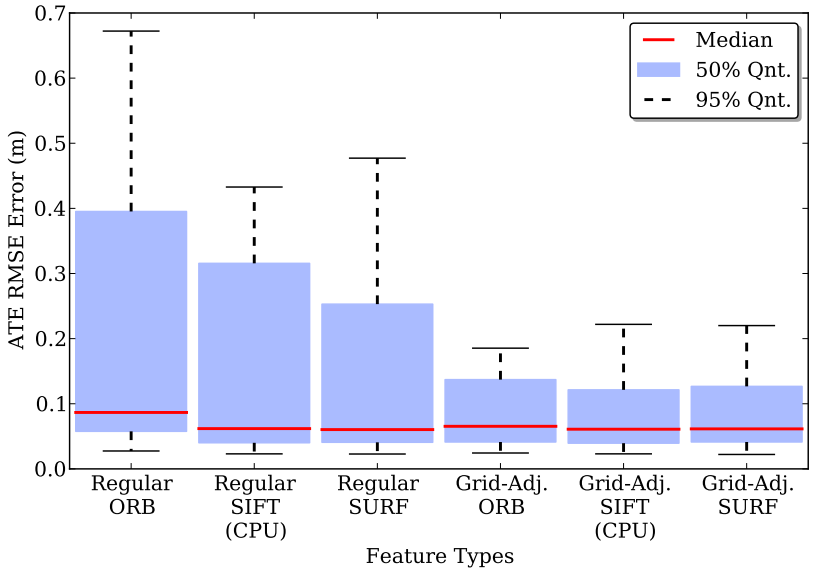


Figure 3.10: Evaluation of the impact of the improved keypoint detection per feature type. It is clearly visible that all feature types benefit from the improvements. Particularly the ORB feature attains approximately equal performance to SURF and SIFT (CPU), while retaining the advantage of its low runtime cost. The plots are accumulated over the datasets used in Figure 3.9, i.e., "fr1" and "Robot SLAM".

effects other than those from erroneous feature matching. As the change in distance measure neither increases the runtime noticeably, nor affects the memory requirements, we suggest the adoption of the Hellinger distance.

For matching ORB features we found that the nearest neighbor ratio, as proposed by Lowe for SIFT (see Section 3.1.2.1, Equation 3.1) to distinguish inliers from outliers, is not as stable as for SIFT. For SIFT and SURF features, this ratio followed a similar distribution in the scenarios we evaluated, so that a fixed threshold can be applied (values between 0.5 and 0.8 have worked well for us). In contrast, the ambiguity of ORB features increases strongly, e.g., with the occurrence of motion blur, such that the threshold results either in a low amount of matches in difficult situations or many outliers otherwise. Having too few matches substantially hurts the transformation estimates, hence a strict threshold for the ratio should be avoided. Many outliers impact the required iterations for RANSAC. We therefore threshold the matches at a permissive ratio of 0.95 but retain only the $M = 150$ matches with the best ratio.

After introducing the above changes, ORB features offer the best trade-off between speed, accuracy and robustness for online processing. The high extraction speed shifts the performance bottleneck to the nearest neighbor search in the feature matching. For SIFT and SURF, we used the FLANN library, which also provides *locality sensitive hashing* [46] (LSH), a nearest neighbor search which yields good results for binary features. We typically only extract up to 800 features per frame. For such small amounts of features

LSH is slower than brute force matching. To obtain real-time performance while keeping the accuracy and robustness high, we modified the computation of the hamming distance for ORB descriptors. The hamming distance is the number of differing bits. The binary XOR operation on two bit sequences results in a sequence in which the bits are set where the input bits differ. The hamming weight (the number of bits set, also called popcount) of this sequence then yields the hamming distance. There are many fast methods that compute the hamming weight, e.g., the method of Wegner [122] which is linear in the number of set bits, and the use of lookup tables. The lookup table method uses the bit sequence for indexing an array, which contains the precomputed hamming weights at the appropriate positions. This method executes in constant time but requires excessively large tables for long bit sequences. Standard ORB features require 32 bytes (256 bits), thus a single table is infeasible ($2^{256} \approx 10^{77}$ bytes). The standard implementation for the computation of the hamming distance in OpenCV uses a lookup table with a table size of 256, so that the hamming distance is aggregated byte-wise. We reimplemented the computation of the hamming distance using the 64 bit *POPCNT* CPU instruction, which is available, e.g., in the Intel Core i7 architecture since 2008. This reduces the distance computation for 32 byte ORB features to 4 XOR operations, 4 POPCNT operations and 3 summations. In our experiments, using our implementation of the Hamming distance computation in the brute force search for matching features yields a speedup by a factor of 8 compared to the respective OpenCV implementation of brute force search. Compared to LSH, the performance is improved by a factor of 9.6 to 15 (using 2600 and 600 features per frame respectively).

3.4 Exploiting the Graph Neighborhood for Loop Closure Search

In this section we improve our initial strategy for matching the current set of features to those of earlier frames, as described in Section 3.1.2.3. Even though we only search for loop closures in the set of keyframes, the computational expense grows linearly with the number of estimates. For multi-core processors, this is mitigated to a certain degree, since the individual frame-to-frame estimates are independent and can therefore be easily parallelized. However, a comparison of a new frame to all predecessor frames is not feasible and the possibility of estimating a valid transformation is strongly limited by the overlap of the field of view, the repeatability of the keypoint detector and the robustness of the keypoint descriptor.

Therefore, we require a more efficient strategy for selecting candidate frames for which we estimate the transformation. Recognition of images in large sets of images has mostly been investigated in the context of image retrieval systems [93] but also for large scale

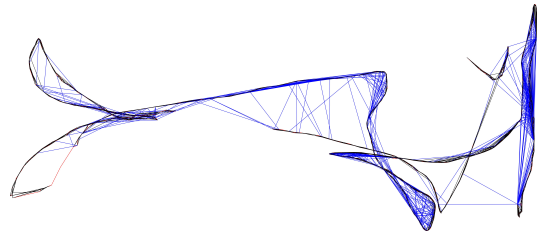
SLAM [20]. While these methods are required for datasets spanning hundreds of kilometers, they require an offline training step to build efficient data structures. Other research groups proposed to use the current trajectory estimate as a basis to compute a potential geometric relationship, i.e. distance or view overlap between frames, to decide whether the frames should be matched. This is very effective for systems with frequent closure of small loops. However, loop closing is most valuable when the loop is big and a lot of drift has accumulated. In these cases, approaches based on the geometry of the trajectory will likely fail.

We therefore propose an efficient, straightforward-to-implement algorithm to suggest candidates for frame-to-frame matching. It is based on the available graph structure and complementary to the approaches described above. We employ a strategy with three different types of candidates. Firstly, we apply the egomotion estimation to n immediate predecessors. To efficiently reduce the drift, we secondly search for loop closures in the geodesic graph-neighborhood of the previous frame. We compute a minimal spanning tree of limited depth from the pose graph, with the sequential predecessor as the root node. We then remove the n immediate predecessors from the tree to avoid duplication and randomly draw k frames from the tree with a bias towards earlier frames. We therefore guide the search for potentially successful estimates by those previously found. In particular, when the robot revisits a place, once a loop closure is found, this procedure exploits the knowledge about the loop by preferring candidates near the loop closure during sampling.

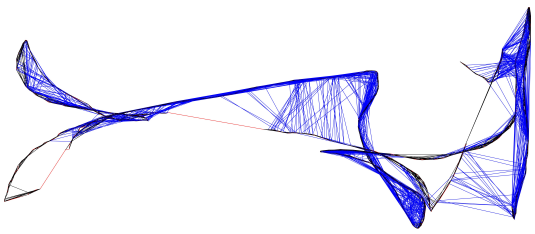
Figure 3.11 shows a comparison between a pose graph constructed without and with sampling of the geodesic neighborhood. The extension of found loop closures is clearly visible. The graphs have both been created while matching each frame to 9 others. For the graphs on the left we compared $n = 3$ immediate predecessors and $k = 6$ randomly sampled keyframes. The graphs on the right have been created with $n = 2$, $k = 5$ and $l = 2$ sampled frames from the geodesic neighborhood.

3.5 Statistical Graph Pruning for Increased Robustness

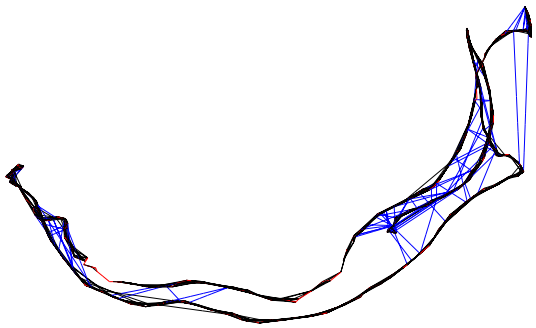
The graph optimization backend is a crucial part of our SLAM system. It significantly reduces the drift in the trajectory estimate. However, in some cases, the graph optimization may result in a distorted trajectory. Common causes for this are, e.g., wrong "loop closures" due to repeated structure in the scenario, or highly erroneous transformation estimates due to systematically wrong depth information. Fast motions may lead to such problems in current sensor hardware, because of rolling shutters and missing synchronization of the color and infrared camera's shutter. Thin structures such as cables and thin poles or pipes are often visible in the color image but not in the depth image. This leads to sets of visual features with a consistent offset in their position and motion. The



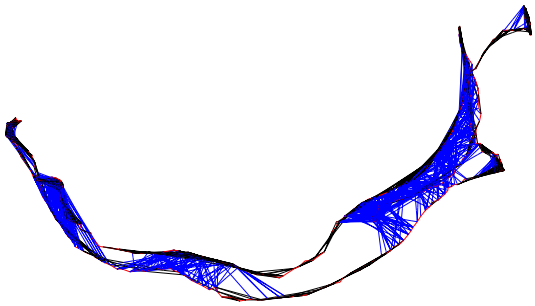
(a) Pose graph for "fr1 floor", regular approach



(b) Pose graph for "fr1 floor", with graph-distance based sampling



(c) Pose graph for "fr1 desk2", regular approach



(d) Pose graph for "fr1 desk2", with graph-distance based sampling

Figure 3.11: Pose graph for the sequences "fr1 floor" and "fr1 desk2". Left: Transformation estimation only to consecutive predecessors and randomly sampled frames. Right: Additional exploitation of previously found matches using the geodesic neighborhood. In both runs, the same overall number of candidate frames for frame-to-frame matching were processed. On the challenging "Robot SLAM" dataset, this reduces the average error by 26 %.

validity of transformations can therefore not be guaranteed in every case. However, given several constraints, of which only some are affected by these errors, the residual error of the edges in the graph after optimization allows to detect such inconsistencies. We therefore propose to prune inconsistent transformations in the graph. For this we use a threshold on the summands of the error function after the initial convergence, i.e., on

$$e_{ij}^2 = \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})^\top \boldsymbol{\Omega}_{ij}\ \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij}) \tag{3.6}$$

(see also Equation 3.3). Each summand represents the Mahalanobis distance between the measurement and the estimate. After pruning the edges, we continue the optimization. We successively iterate the method from coarse to fine, so that the initial optimization of the graph is the basis for pruning with a permissive threshold. The result is then used as the initial guess for the next pruning step with a stricter threshold. This iterative approach prunes the edges with the highest error first. This allows the estimates that were affected by these edges to converge according to the remaining edges. If pruning were done in

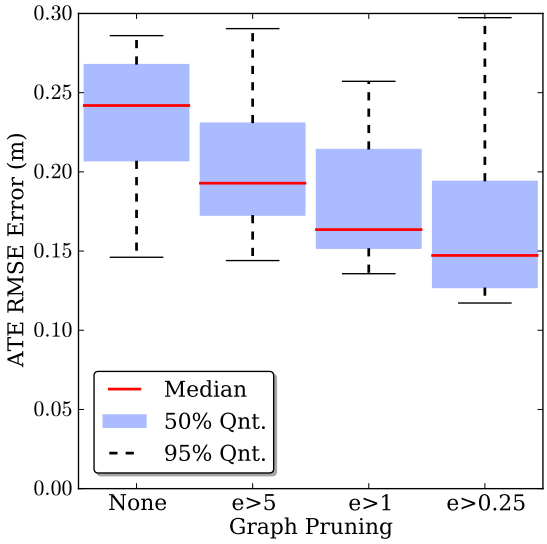


Figure 3.12: Our approach for pruning edges yields additional robustness against outliers. "None" denotes regular optimization with the Huber kernel. The remaining results denote successive optimization iterations, where edges with residual error higher than e are pruned from the graph. The error is given by the Mahalanobis distance of the optimized pose offset to the original pose offset associated with the edge. The results are accumulated over the challenging "Robot SLAM" dataset and the feature types SURF, SIFT (CPU), and ORB.

a single step, the error in the estimate, caused by the erroneous edges, could lead to the premature pruning of "good" edges. This procedure can be interpreted as a robust kernel (see Section 2.4.1) with constant error outside of a threshold, which is initially set to infinity and iteratively reduced after convergence. However, as we entirely remove the edges from the graph, the computational cost of the optimization is decreased.

Interestingly, Fischler and Bolles [43] describe (and dismiss) a similar approach in the context of low dimensional parameter estimation techniques, where one first uses all data to find an initial guess, then removes the furthest outlier and reiterates. They report unsatisfactory results for a line estimation problem and propose RANSAC instead. However, RANSAC is not feasible for selecting the "good" edges of the graph. The reason is that hypotheses that include large loop closures – which are required for reducing drift – will not have a larger consensus set than those without.

Figure 3.12 shows experimental results on the "Robot SLAM" dataset, which demonstrate the improvement in accuracy obtained using different thresholds. For these sequences, the presented approach is a crucial step towards reliable mapping. For lower thresholds than shown ($<$0.25), the estimates become unreliable. While the final solution then often converges to the edges with the "best" measurement, the average of several good edges is often superior. In our experiments we found that the thresholds shown in Figure 3.12 consistently work well across scenarios.

3.6 A Method for Verifying the Registration of Depth Images

Motion estimation techniques such as ICP and feature correspondence based alignment with RANSAC are core components of state-of-the-art SLAM systems. It is therefore crucial that the estimates are reliable. Given a high percentage of inliers after optimization, both methods may be assumed successful. However, a low percentage does not necessarily indicate an unsuccessful transformation estimate and could be a consequence of a small overlap between the frames or few visual features, e.g., due to motion blur, occlusions or lack of texture. Hence, both methods lack a reliable failure detection.

We therefore propose a method to verify a transformation estimate, independent of the estimation method used. Our method exploits the availability of structured dense depth data, in particular the contained dense free-space information. We propose the use of a beam-based environment measurement model (EMM). An EMM can be used to penalize pose estimates under which the sensor measurements are improbable given the physical properties of the sensing process. In our case, we employ a beam model, to penalize transformation estimates for which observations in one depth image should actually be occluded by observations of the other depth image. Consider for example a wall that is about one meter in front of the sensor and we perceive this wall twice, from the same pose. However, for some reason the transformation estimate between the poses is misplaced by half a meter towards the wall. If we assume the estimate to be valid, we would assume there are two walls. However, in that case the nearer wall should have occluded the further wall, therefore the observation of the further wall is in conflict with the transformation estimate.

EMMs have been extensively researched in the context of 2D Monte Carlo Localization and SLAM methods [119], where they are used to determine the likelihood of particles on the basis of the current observation. Beam-based models have been mostly used for 2D range finders such as laser range scanners, where the range readings are evaluated using ray casting in the map. While this is typically done in a 2D occupancy grid map, a recent adaptation of such a beam-based model for localization in 3D voxel maps has been proposed by Oßwald et al. [94]. Unfortunately, the EMM cannot be trivially adapted for our purpose. First, due to the size of the input data, it is computationally expensive to compute even a partial 3D voxel map in every time step. Second, since a beam model only provides a probability density for each beam [119], we still need to find a way to decide whether to accept the transformation based on the observation. The resulting probability density value obtained for each beam does not constitute an absolute quality measure. Neither does the product of the densities of the individual beams. The value for, e.g., a perfect match will differ depending on the range value. In Monte Carlo methods, the probability density is used as a likelihood value that determines the particle weight. In

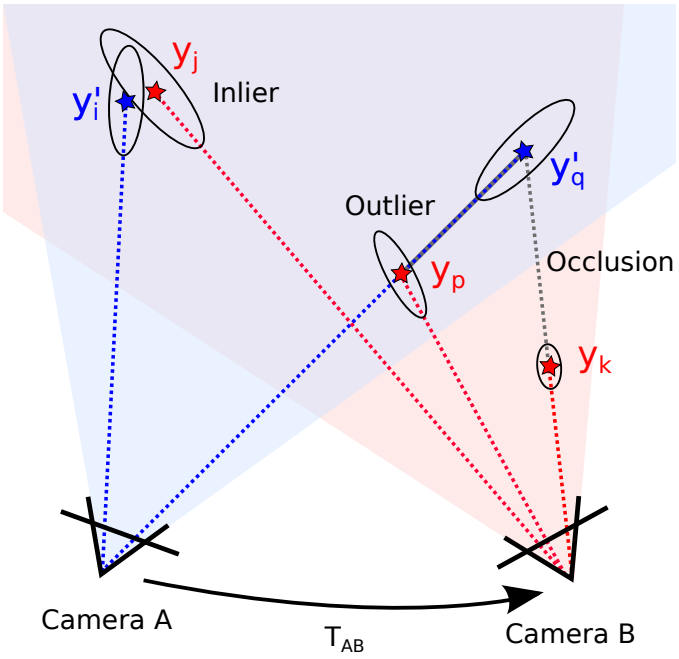


Figure 3.13: Two cameras and their observations aligned by the estimated transformation T_{AB}. In the projection from camera A to camera B, the data association of $\boldsymbol{y}'_i$ and $\boldsymbol{y}_j$ is counted as an inlier. The projection of $\boldsymbol{y}'_q$ cannot be seen from Camera B, as it is occluded by $\boldsymbol{y}_k$. We assume that each point occludes an area of one pixel. Projecting the points observed from camera A to camera B, the association between $\boldsymbol{y}'_i$ and $\boldsymbol{y}_j$ is counted as inlier. In contrast, $\boldsymbol{y}_p$ is counted as outlier, as it falls in the free space between camera A and observation $\boldsymbol{y}'_q$. The last observation, $\boldsymbol{y}_k$ is outside of the field of view of camera A and therefore ignored. Hence, the final result of the EMM is 2 inliers, 1 outlier and 1 occluded.

the resampling step, this weight is used as a comparative measure of quality between the particles. This is not applicable in our context as we do not perform a comparison between several transformation candidates for one measurement. We thus need to compute an absolute quality measure.

3.6.1 Environment Measurement Model

To compute an absolute quality measure for a given estimate of the transformation between two depth images, we propose to use a procedure analogous to statistical hypothesis testing. In our case the null hypothesis being tested is the assumption that after applying the transformation estimate, spatially corresponding depth measurements stem from the same underlying surface location.

To compute the spatial correspondences for an alignment of two depth images I'_D and I_D, we project the points $\boldsymbol{y}'_i$ of I'_D into I_D to obtain the points $\boldsymbol{y}_i$ (denoted without the prime). The image raster allows for a quick association of $\boldsymbol{y}_i$ to a corresponding depth

reading $\boldsymbol{y}_j \in I_D$. Since $\boldsymbol{y}_j$ is given with respect to the sensor pose it implicitly represents a beam, as it contains information about free space, i.e., the space between the origin and the measurement. Points that do not project into the image area of I_D or onto a pixel without valid depth reading are ignored. Figure 3.13 illustrates the different cases of associated observations.

For the considered points we model the measurement noise according to the equations for the covariances given by Khoshelham and Elberink [71], from which we construct the covariance matrix $\boldsymbol{\Sigma}_j$ for each point $\boldsymbol{y}_j$. The sensor noise for the points in the second depth image is represented accordingly. To transform a covariance matrix of a point to the coordinate frame of the other sensor pose, we rotate it using $\mathbf{R}$, the rotation matrix of the estimated transformation, i.e., $\boldsymbol{\Sigma}_i = \mathbf{R}^T \boldsymbol{\Sigma}_i' \mathbf{R}$.

The probability for the observation $\boldsymbol{y}_i$ given an observation $\boldsymbol{y}_j$ from a second frame can be computed as

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta \, p(\boldsymbol{y}_i, \boldsymbol{y}_j), \text{ with } \eta = p(\boldsymbol{y}_j)^{-1} \tag{3.7}$$

Since the observations are independent given the true obstacle location $\boldsymbol{z}$ we can rewrite the right-hand side to

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta \int p(\boldsymbol{y}_i, \boldsymbol{y}_j \mid \boldsymbol{z}) \, p(\boldsymbol{z}) \, d\boldsymbol{z}, \tag{3.8}$$

$$= \eta \int p(\boldsymbol{y}_i \mid \boldsymbol{z}) \, p(\boldsymbol{y}_j \mid \boldsymbol{z}) \, p(\boldsymbol{z}) \, d\boldsymbol{z}, \tag{3.9}$$

$$= \eta \int \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{z}, \boldsymbol{\Sigma}_i) \, \mathcal{N}(\boldsymbol{y}_j; \boldsymbol{z}, \boldsymbol{\Sigma}_j) \, p(\boldsymbol{z}) \, d\boldsymbol{z}. \tag{3.10}$$

Exploiting the symmetry of Gaussians we can rewrite this to

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta \int \mathcal{N}(\boldsymbol{z}; \boldsymbol{y}_i, \boldsymbol{\Sigma}_i) \, \mathcal{N}(\boldsymbol{z}; \boldsymbol{y}_j, \boldsymbol{\Sigma}_j) p(\boldsymbol{z}) \, d\boldsymbol{z}. \tag{3.11}$$

The product of the two normal distributions contained in the integral can be rewritten [97] so that we obtain

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta \int \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{y}_j, \boldsymbol{\Sigma}_{ij}) \, \mathcal{N}(\boldsymbol{z}; \boldsymbol{y}_c, \hat{\boldsymbol{\Sigma}}_{ij}) p(\boldsymbol{z}) \, d\boldsymbol{z}, \tag{3.12}$$

$$\text{where } \boldsymbol{y}_c = (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1})^{-1} (\boldsymbol{\Sigma}_i^{-1} \boldsymbol{y}_i + \boldsymbol{\Sigma}_j^{-1} \boldsymbol{y}_j)^{-1}, \tag{3.13}$$

$$\boldsymbol{\Sigma}_{ij} = \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j \quad \text{and} \quad \hat{\boldsymbol{\Sigma}}_{ij} = (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1})^{-1}. \tag{3.14}$$

The first term in the integral in Equation 3.12 is constant with respect to $\boldsymbol{z}$, which allows us to move it out of the integral

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta\, \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{y}_j, \boldsymbol{\Sigma}_{ij}) \int \mathcal{N}(\boldsymbol{z}; \boldsymbol{y}_c, \hat{\boldsymbol{\Sigma}}_{ij}) p(\boldsymbol{z})\, d\boldsymbol{z}. \tag{3.15}$$

Since we have no prior knowledge about $p(\boldsymbol{z})$ we assume it to be a uniform distribution. As it is constant, the value of $p(\boldsymbol{z})$ thus becomes independent of $\boldsymbol{z}$ and we can move it out of the integral. We will see below that the posterior distribution remains a proper distribution despite the choice of an improper prior [9]. The remaining integral only contains the normal distribution over $\boldsymbol{z}$ and, by the definition of a probability density function, reduces to one, leaving only

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \eta\, \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{y}_j, \boldsymbol{\Sigma}_{ij})\, p(\boldsymbol{z}). \tag{3.16}$$

Having no prior knowledge about the true obstacle also means we have no prior knowledge about the measurement. This can be shown by expanding the normalization factor

$$\eta = p(\boldsymbol{y}_j)^{-1} = \left(\int p(\boldsymbol{y}_j | \boldsymbol{z})\, d\boldsymbol{z} \right)^{-1} \tag{3.17}$$

$$= \left(\int \mathcal{N}(\boldsymbol{y}_j; \boldsymbol{z}, \Sigma_j) p(\boldsymbol{z})\, d\boldsymbol{z} \right)^{-1} \tag{3.18}$$

and using the same reasoning as above, we obtain

$$p(\boldsymbol{y}_j)^{-1} = \left(p(\boldsymbol{z}) \int \mathcal{N}(\boldsymbol{z}; \boldsymbol{y}_j, \Sigma_j)\, d\boldsymbol{z} \right)^{-1} \tag{3.19}$$

$$= p(\boldsymbol{z})^{-1} \tag{3.20}$$

Combining Equation 3.20 and Equation 3.16, we get the final result

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_j) = \mathcal{N}(\, \boldsymbol{y}_i \,;\, \boldsymbol{y}_j, \boldsymbol{\Sigma}_{ij}) \tag{3.21}$$

$$= \mathcal{N}(\Delta \boldsymbol{y}_{ij}; \boldsymbol{0}, \boldsymbol{\Sigma}_{ij}), \tag{3.22}$$

where $\Delta \boldsymbol{y}_{ij} = \boldsymbol{y}_i - \boldsymbol{y}_j$. We can combine the above three-dimensional distributions of all data associations to a $3N$-dimensional normal distribution, where N is the number of data associations. Assuming independent measurements yields

$$p(\boldsymbol{Y}_i \mid \boldsymbol{Y}_j) = \mathcal{N}(\Delta \boldsymbol{Y} \mid \boldsymbol{0}, \boldsymbol{\Sigma}) \in \mathbb{R}^{3N}, \tag{3.23}$$

where $\boldsymbol{\Delta} Y = (\ldots,\ \Delta \boldsymbol{y}_{ij}^\top,\ \ldots)^\top$ is a column vector containing the N individual terms $\Delta \boldsymbol{y}_{ij}$. The column vectors $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$ analogously contain the corresponding landmarks

of the two depth images, and $\mathbf{\Sigma}$ contains the respective covariance matrices $\mathbf{\Sigma}_{ij}$ on the (block-) diagonal.

The above formulation contains no additional term for "short readings" as given in [119], i.e., unexpectedly short measurements. The reason is that we expect a static environment during mapping and want to penalize such short readings, as it is our main indication for misalignment. In contrast, range readings that are projected behind the corresponding depth value, are common, e.g., when looking behind an obstacle from a different perspective. "Occluded outliers", the points projected far behind the associated beam (e.g., further away than three standard deviations) are therefore ignored. However, we do want to use the positive information of "occluded inliers", points projected closely behind the associated beam, which in practice confirm the transformation estimate.

A standard hypothesis test could be used for rejecting a transformation estimate at a certain confidence level, by testing the p-value of the Mahalanobis distance for $\mathbf{\Delta}Y$ for a χ^2_{3N} distribution (a chi-square distribution with $3N$ degrees of freedom). In practice, however, this test is very sensitive to small errors in the transformation estimate and therefore hardly useful. Even under small misalignments, the outliers at depth jumps will be highly improbable under the given model and will lead to rejection. We therefore describe a measure that varies more smoothly with the error of the transformation in the following.

3.6.2 Robust Hypothesis Testing

Analogously to non-parametric statistical hypothesis tests, we use the parametric hypothesis test only on the distributions of the individual observations Equation 3.22 and compute the fraction of outliers as a criterion to reject a transformation. Assuming a perfect alignment and independent measurements, the fraction of inliers within, e.g., three standard deviations can be computed from the cumulative density function of the normal distribution. The fraction of inliers is independent of the absolute value of the outliers and thus smoothly degenerates for increasing errors in the transformation while retaining an intuitive statistical meaning. Our experimental evaluation, described in the following section, shows that applying a threshold on this fraction allows to effectively reduce the number of highly erroneous transformation estimates and obtain a greatly improved quality of the map.

3.6.3 Implementation and Evaluation

Our system estimates the transformation of captured RGB-D frames to a selection of previous frames. After computing such a transformation, we compute the number of inliers, outliers, and occluded points, by applying the three sigma rule to Equation 3.22. As stated,

points projected within a Mahalanobis distance of three are counted as inliers. Outliers are classified as occluded if they are projected behind the corresponding measurement.

The data association between projected point and beam is not symmetric. As shown in Figure 3.13, a point projected outside of the image area of the other frame has no association. Nevertheless, in the reversed process it could occlude or be occluded by a projected point. We therefore evaluate both, the projection of the points in the new depth image to the depth image of the older frame and vice versa. To reduce the requirements on runtime and memory, we subsample the depth image. In the presented experiments we construct our point cloud using only every 8th row and column of the depth image, effectively reducing the cloud to be stored to a resolution of 80 by 60. This also decreases the statistical dependence between the measurements. In our experiments, the average runtime for the bidirectional EMM evaluation was 0.82 ms per depth image pair.

We compute the quality q of the point cloud alignment using the number of inliers I and the sum of inliers and outliers $I + O$ as $q = \frac{I}{I+O}$. To avoid accepting transformations with nearly no overlap we also require the inliers to be at least 25 % of the observed points, i.e., inliers, outliers and occluded points.

To evaluate the effect of rejecting transformations with the EMM, we ran experiments with eight minimum values for the quality q. To avoid reporting results depending on a specific parameter setting, we show statistics over many trials with varied parameters settings.

Analogous to the findings for the statistical pruning of edges presented in Section 3.5, our approach yields only minimal impact on the results for the "fr1" dataset. A q-threshold from 0.25 to 0.9 results in a minor improvement over the baseline (without EMM). The number of edges of the pose graph is only minimally reduced and the overall runtime increases slightly due to the additional computations. For thresholds above 0.95, the robustness decreases. While most trials remain unaffected, the system performs substantially worse in a number of trials. We conclude from these experiments, that the error in the "fr1" dataset does not stem from individual misalignments, for which alternative higher-precision alignments are available. In this case, the EMM-based rejection will provide no substantial gain, as it can only filter the estimates.

In contrast, the same evaluation on the four sequences of the "Robot SLAM" category results in greatly increased accuracy. The rejection of inaccurate estimates and wrong associations significantly reduces the error in the final trajectory estimates. As apparent in Figure 3.14a the use of the EMM decreases the average error for threshold values up to 0.9.

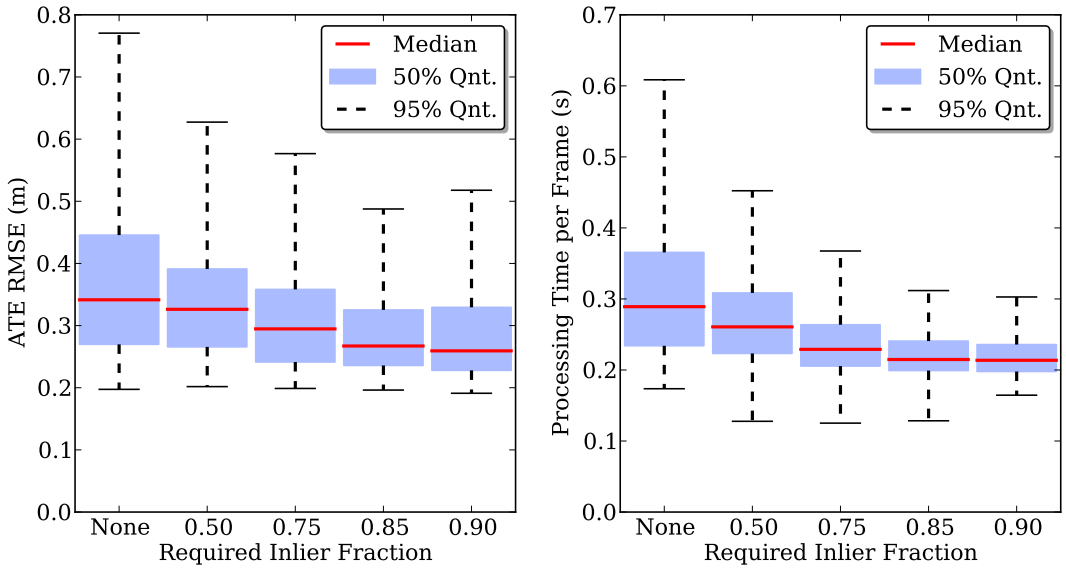


Figure 3.14: Evaluation of the proposed environment measurement model (EMM) for the "Robot SLAM" scenarios of the RGB-D benchmark for various quality thresholds. (a) The use of the EMM substantially reduces the error. (b) Due to the reduced number and increased consistency of the constraints, the graph optimizer converges faster, which greatly reduces the overall runtime.

3.7 Related Work

Because SLAM is such a fundamental problem for mobile robots, there has been intensive research on the subject for over two decades. The SLAM community has proposed a huge variety of SLAM approaches that cover all kinds of sensors, optimization techniques and map representations.

Wheeled robots often rely on laser range scanners, which commonly provide very accurate geometric measurements of the environment at high frequencies. To compute the relative motion between observations, many state-of-the-art SLAM systems use variants of the iterative-closest-point (ICP) algorithm [8, 44, 107]. A variant particularly suited for man-made environments uses the point-to-line metric [15]. Recent approaches demonstrate that the robot pose can be estimated at millimeter accuracy [103] using two laser range scanners and ICP. Disadvantages of ICP include the dependency on a good initial guess to avoid local minima and the lack of a measure of the overall quality of the match (see Section 3.6 for an approach to the latter problem). The resulting motion estimates can be used to construct a pose graph [49, 50, 66, 77]. Approaches that use planar localization and a movable laser range scanner, e.g., on a mobile base with a pan-tilt unit or at the tip of a manipulator, allow precise localization in 3D using a 2D sensor. In combination with an inertial measurement unit (IMU), this can also be used to create a

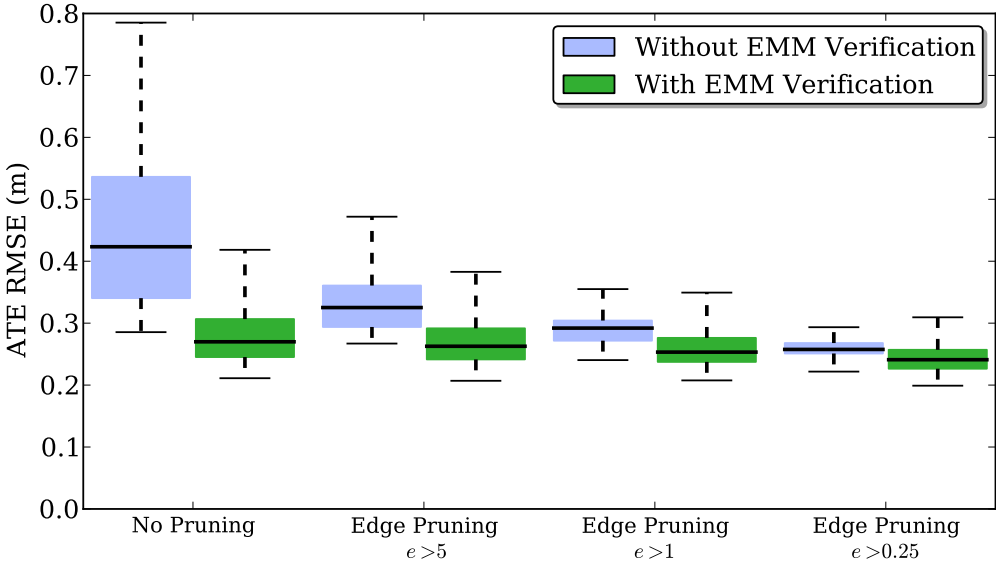


Figure 3.15: The EMM verification method presented in this section, and the statistical graph edge pruning presented in Section 3.5 both yield a higher accuracy. The best results are obtained when combining both approaches. The evaluation comprises 240 runs on the challenging "Robot SLAM" scenario, accumulating results from various parameterizations for the required EMM inlier fraction (as in Fig. 3.14), feature count (600-1200) and matching candidates (18-36).

map with a quadrocopter [51]. However, since a laser range scanner only covers a 2D slice of the world at any given time, creating a dense 3D model is much faster when using a 3D sensor.

Visual SLAM approaches [23, 40, 72, 113], also referred to as "structure and motion estimation" [65, 92] compute the robot's motion and the map using cameras as sensors. Stereo cameras are commonly used to gain sparse distance information from the disparity in textured areas of the respective images. Many techniques used in our SLAM system are adapted from those used for visual SLAM with stereo cameras.

RGB-D sensors provide high resolution visual and geometric information at the same time. This is particularly useful in robot navigation. The dense depth information lets us reliably map the free space, i.e., where the robot can pass. The combination of color with dense depth information allows us to estimate the robot motion based on correspondences of visual features, photoconsistency, and ICP variants.

In contrast to 2D point clouds as given by laser range scanners, for the Microsoft Kinect the number of points is increased by several orders of magnitude. This can be reduced by subsampling the point cloud; however, a sufficient representation of a 3D scene obviously requires substantially more data than a 2D slice of it. More problematic than the size of

the data sets is the limited field of view, which amplifies the problem of having sufficiently distinctive geometric structure. Ambiguous scenarios, analogous to the "long corridor" in 2D SLAM, are therefore common in RGB-D data. Further, the increased dimensionality of the search space from three to six degrees of freedom results in a greater dependency on a good initialization. In consequence of the mentioned problems, a straightforward application of ICP to RGB-D data will not achieve a comparable performance to that in the case of 2D laser range scanner data.

Nevertheless, ICP can be applied successfully for motion estimation in 3D. Generalized ICP [110] is an ICP variant particularly designed for 3D point clouds, and can employ a point-to-plane metric to reduce spurious motions along the principal planes of a scene, e.g., floor and walls. However, except in situations with highly unambiguous geometry and few visually salient keypoints, there has been no improved accuracy in the motion estimate of GICP as compared to the estimate from feature correspondences in our experiments.

RGB-D Odometry

The density and high frame rate of the depth information prompted a number of novel approaches for RGB-D odometry. Huang et al. [60] propose an odometry approach based on tracking FAST keypoints [104]. Since rotation may cause big motions in distant features, they first estimate the rotation and then search for the keypoint in a small window, using a 9×9 patch of intensity values for matching. To reduce the short-term drift, they keep one reference keyframe, which is only replaced when it goes out of view. Some of the techniques they use for keypoint detection are similar to ours, as detailed in Section 3.3.1. They published their software under the name *Fovis*. Whelan et al. [124] and Handa et al. [53] report results on Fovis which are included in Table 3.1 and Figure 3.18.

Steinbruecker et al. [112] propose a transformation estimation based on the minimization of an energy function. For frames close to each other, they achieve an improved runtime performance and accuracy compared to GICP. Using the distribution of normals, Osteen et al. [95] improve the initialization of ICP by efficiently computing the difference in orientation between two frames, which allows them to substantially reduce the drift. These approaches work well for small incremental changes between consecutive frames, an assumption that is justified by the high frame rate of the RGB-D sensors. Other groups propose adaptations of ICP to deal with the high amount of data more efficiently [25, 100] or to include the additional information from color [124] or visual features [56].

Dryanovski et al. [26] proposed a real-time capable approach using keypoints as landmarks in a global map. The landmarks are matched to new observations via locality instead of a feature descriptor. They propose an extended noise model to improve the estimation of the Mahalanobis distance between landmarks and new observations. Although this approach does not scale to large environments, they are able to close small loops.

The processing power of modern graphics hardware gave rise to several approaches that are leveraged by the massive parallelization. KinectFusion [91] was the first of a series of approaches for surface reconstruction based on a voxel grid containing the truncated signed distance [22] to the surface. The sensor pose relative to the world model is estimated by ICP variants, which register the global surface with the current measurement. Each measurement is directly fused into the voxel representation. This reduces the drift as compared to the frame-to-frame comparisons we employ, yet lacks the capability to recover from accumulating drift by loop closures. Real-time performance is achieved, but requires high performance graphics hardware. The size of the voxel grid has cubic influence on the memory usage, so that KinectFusion only applies to small workspaces. Kintinuous [124] overcomes this limitation by virtually moving the voxel grid with the current camera pose. The parts that are shifted out of the reconstruction volume are triangulated to form a mesh. This allows to deal with loop closures by deformation of the mesh [125]. Table 3.1 shows a comparison between the results reported by Whelan et al. [124] to our approach. Our approach achieves the best result on three of four sequences, using a real time capable configuration running on the CPU only.

RGB-D SLAM Systems

The first scientifically published SLAM system for Kinect-type RGB-D cameras was proposed by Henry et al. [55, 56] who use visual features in combination with GICP [110] to create and optimize a pose graph. Unfortunately neither the software nor the data used for evaluation have been made publicly available, so that a direct comparison cannot be carried out.

Meilland and Comport [88] combine voxel representations with the advantages of keyframes in their RGB-D SLAM approach. They propose to fuse multiple frames into extended keyframes to avoid the loss of information due to the incomplete overlap of the field of view, when using regular keyframes. Table 3.1 shows the results reported by them in comparison to those obtained with our approach. Zeng et al. [130] show that the memory requirements of the voxel grid can be greatly reduced using an octree to store the distance values. Hu et al. [59] recently proposed a SLAM system that switches between bundle adjustment with and without available depth, which makes it more robust to lack of depth information, e.g., due to distance limitations and sunlight.

Kerl et al. first proposed to optimize a photoconsistency error term to align successive RGB-D frames [69] for RGB-D odometry. They further proposed a robust approach to combine the photoconsistency error with an error term based on the dense depth data [70]. Their approach runs in real-time on a CPU, using the same hardware as in our experiments. To benefit from loop closures, they augment this visual odometry by keyframe based visual SLAM with a pose graph. A comparison of this SLAM approach with ours is shown in Figure 3.16. The results of Kerl et al. are highly accurate, but their

Sequence	Our Approach Real-Time	Kintinuous ICP+RGB-D	Multi-Keyframe VP	Fovis
	Median ATE			
fr1 desk	**0.016 m**	0.069 m	0.018 m	0.221 m
fr2 desk	**0.074 m**	0.119 m	0.093 m	0.112 m
fr1 room	**0.076 m**	0.158 m	0.144 m	0.238 m
fr2 large no loop	0.826 m	0.256 m	**0.187 m**	0.273 m
	Maximum ATE			
fr1 desk	0.080 m	0.234 m	**0.066 m**	0.799 m
fr2 desk	**0.111 m**	0.362 m	0.116 m	0.217 m
fr1 room	**0.124 m**	0.421 m	0.339 m	0.508 m
fr2 large no loop	1.574 m	0.878 m	**0.317 m**	0.897 m

Table 3.1: Comparison of the trajectory error of our approach with the Kintinuous ICP+RGB-D approach (Whelan et al. [124]), the Fovis Visual Odometry approach (as reported in [124]) and the multi-keyframe approach using view prediction (Meilland et al. [88]). The compared approaches are running in real-time. We therefore compare against a real-time configuration of our approach, which performs at 30 Hz on average. Note that our approach and Fovis only use the CPU, while the other approaches require a powerful GPU. Nevertheless, we outperform the other approaches on three of the four sequences w.r.t. the median. We compare median and maximum ATE here, as these values are given in the respective publications.

approach fails on the "fr1 floor" sequence, presumably because of their heavy reliance on the visual odometry which breaks at the short sensor outages in that sequence. Meilland and Comport [88] propose a GPU based real-time tracking approach, which is also based on optimization of both photometric and geometric constraints. Their approach is used to compute a super-resolution map from a sequence of RGB-D frames.

Maier et al. [85] investigated the performance of full bundle adjustment, i.e., including the landmarks into the graph, with an otherwise similar approach to ours. Further they show that first optimizing local sub-maps and then aligning these is faster with low impact on the accuracy. As they present results on our benchmark dataset, we performed a comparison, which is shown in Figure 3.17. Overall, they achieve an accuracy that is comparable to our approach, with our approach being faster when using a real-time configuration. Unfortunately they do not evaluate their system on the more challenging sequences. The processing times were obtained on comparable hardware: Intel Core i7-2600 CPU with 3.40 GHz (ours) versus Intel Core i7-3770 CPU with 3.40 GHz (theirs). No GPU has been used in all cases.

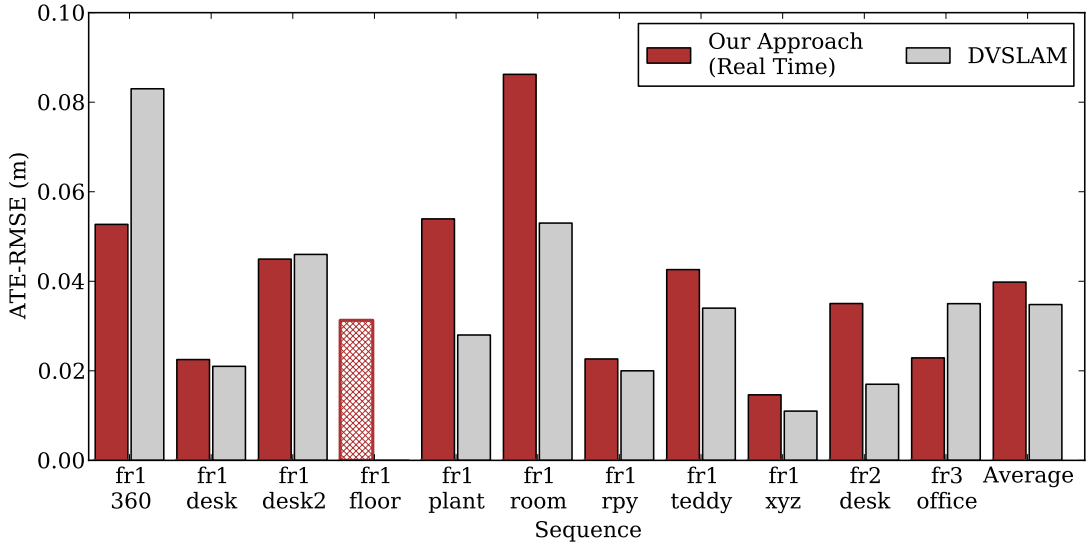


Figure 3.16: Comparison with the results reported by Kerl et al. [70] on our benchmark dataset. Their approach obtains better results on these sequences on average, but fails on the "fr1 floor" dataset (not included in the averaged results).

RGB-D Datasets

Besides our RGB-D benchmark, other researchers have published RGB-D datasets, often adopting the tools and error metrics of our benchmark, but focusing on different aspects in the recorded data. Handa et al. [53] created a dataset for evaluation of RGB-D odometry and surface reconstruction. Since ground truth for the map itself is difficult to obtain, they create 3d models in simulation and use a raytracing software to create color and depth data. The sequences are created by using various camera trajectories obtained in the real world as trajectory for the rendering process. Handa et al. add typical noise to the depth and color. Unfortunately they do not consider effects that result from a rolling shutter, asynchronous depth and color images, motion blur, and the lack of accurate timestamps. They provide ground truth trajectories for eight synthesized sequences, which can be used with the evaluation tools of our benchmark. They published the ATE-RMSE for several visual odometry approaches, which allows us to compare them to our RGB-D odometry (i.e., using the graph optimization backend only to optimize a small graph containing the poses which were successfully matched). As shown in Figure 3.18, on average our approach is the most accurate. However, the high error rate for some of the approaches in the publication of Handa and Whelan et al. [53] does not fit the accuracy reported on real datasets by Whelan et al. [124]. Therefore, one could argue that the way the data is synthesized might cause these problems, e.g., by producing highly ambiguous visual features. Further it is unclear why ICP should become worse when including the color information (ICP and ICP+RGB-D in Figure 3.18).

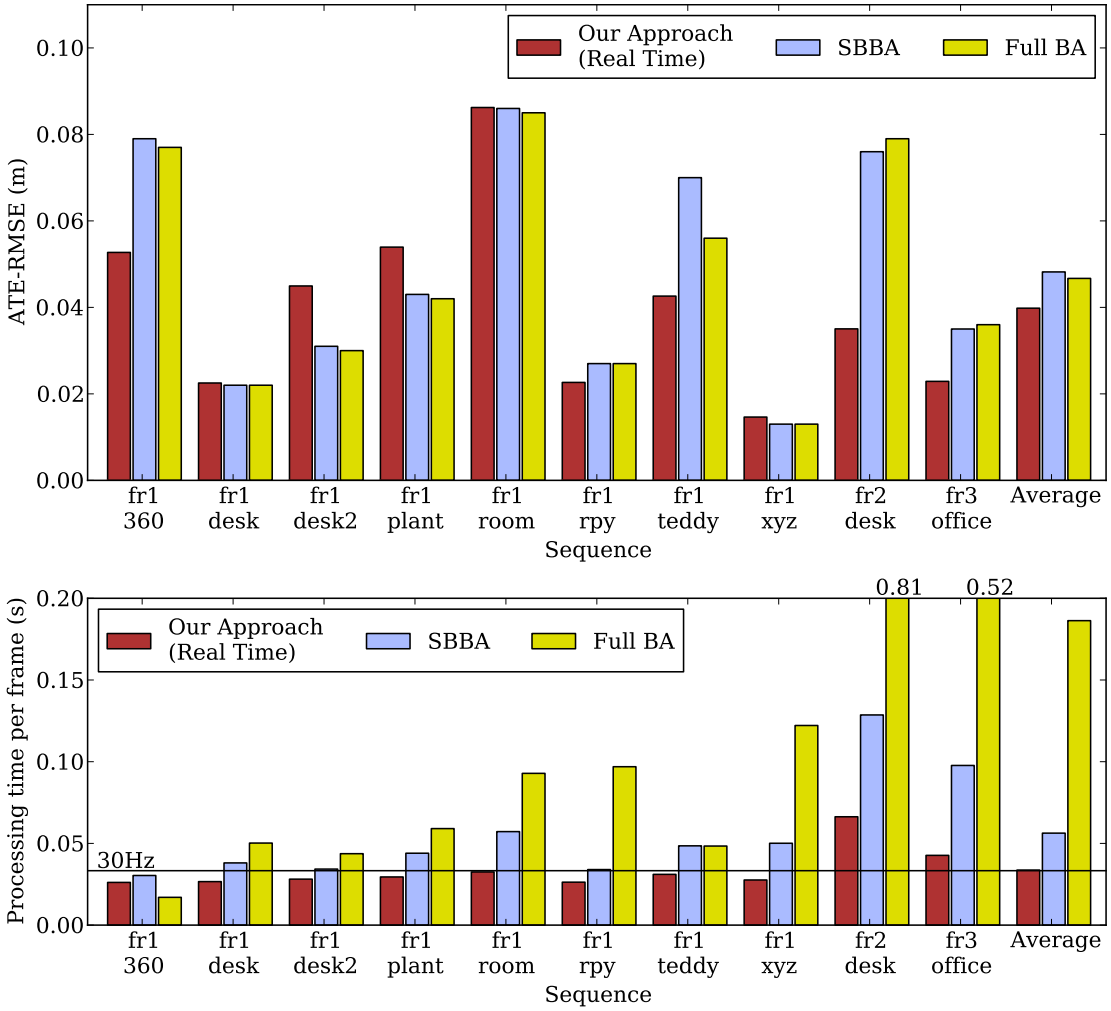


Figure 3.17: Comparison with the results reported by Maier et al. [85] on our benchmark dataset for full bundle adjustment (Full BA) and their sub-map based bundle adjustment (SBBA) approach. In real-time configuration our result provides a similar accuracy to SBBA and Full BA at lower processing times on comparable hardware.

A further dataset that includes RGB-D data was captured at the MIT Stata center [42] and made available online[2]. The dataset was captured with the sensors of a Willow Garage PR2 robot (see Figure 3.6). It contains several long sequences with various combinations of sensor data, some of them with ground truth. For an 229 m long sequence of this dataset ("2012-04-06-11-15-29"), consisting of nearly 20.000 frames, we obtained an ATE-RMSE of 1.65 m at a processing rate of 5.0 Hz.

[2]http://projects.csail.mit.edu/stata/

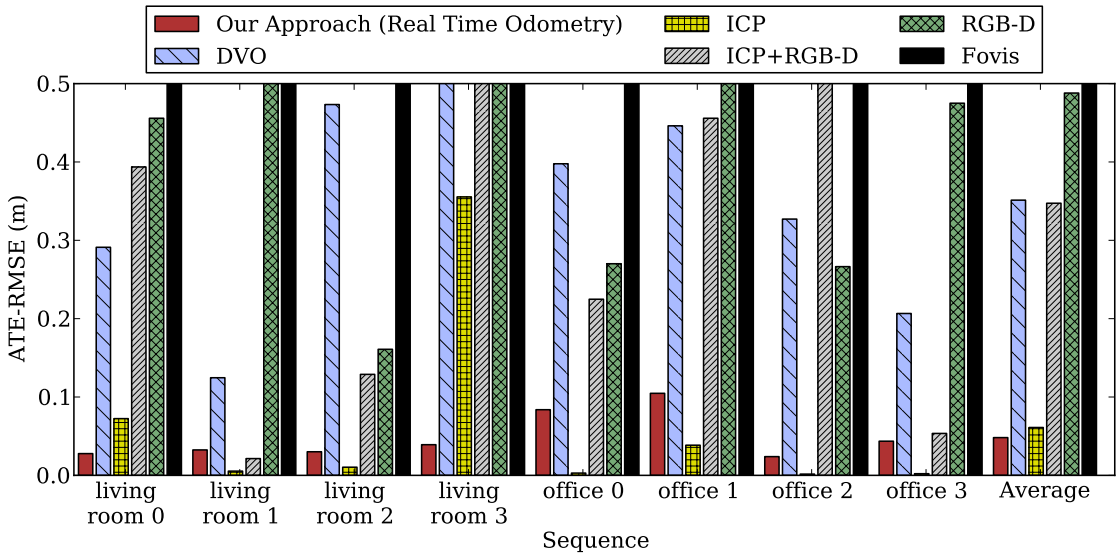


Figure 3.18: Comparison of our approach with the results reported by Handa et al. [53] on their synthetic benchmark dataset using the RGB-D odometry approaches DVO [69], Fovis [60], RGB-D [112], ICP as used in KinectFusion [91] and Kintinuous [123], and its extension ICP+RGB-D [124]. Our approach is configured analogous to the experiments in Figures 3.16 and 3.17, but without final optimization of the full pose graph. Since the size of the simulated scenarios are room-sized, we consider an error of more than 0.5 m as failure.

3.8 Conclusion

In this chapter, we investigated the problem of simultaneous localization and mapping with an RGB-D sensor. We first developed a state-of-the-art SLAM system that exploits the sensor's depth and color information to compute the constraints for a pose graph using sparse visual flow. Second, we created a benchmark for RGB-D SLAM systems, that provides datasets with ground truth and metrics to measure the accuracy. We publicly released the benchmark and the RGB-D SLAM system to allow others to reproduce our results and provide the community with a method to evaluate and compare algorithms quantitatively. In continued research, we proposed and evaluated several methods that improve the basic system with respect to runtime, accuracy and reliability. We increased the number of successful motion estimates, by exploiting the existing graph structure of our optimization backend for geodesic neighbor search. On the other hand, we showed that pruning the edges of the graph based on their statistical error effectively removes deficient motion estimates and thus improves the final result. We further proposed an environment measurement model to evaluate the quality of registration results directly in the SLAM frontend, exploiting the dense depth information. By detecting and rejecting faulty estimates based on this quality measure and the above graph pruning, our approach deals robustly with highly challenging scenarios. We performed a detailed experimen-

tal evaluation of the critical components and parameters based on the benchmark, and characterized the expected error in different types of scenes. We furthermore provided detailed information about the properties of the RGB-D SLAM system that are critical for its performance.

Compared to stereo cameras, RGB-D sensing based on projection of structured light reduces the reliance on texture. However, compared to cameras with fisheye lenses or parabolic mirrors, their field of view is very limited. In combination with the minimum and maximum range, this may lead to periods without usable visual features. Our mapping approach is robust to short sensor outages, but requires overlapping field of views with common feature observations to connect frames in the graph. In the next chapter we address this problem by extending our approach to multiple views, as an absence of features in all views is less likely to occur than for a single view.

Chapter 4

Multiple View RGB-D Perception

Contents

The typically restricted field of view of visual sensors often imposes limitations on the performance of localization and simultaneous localization and mapping (SLAM) approaches. In this chapter, we investigate the use of multiple RGB-D cameras for mapping and develop a solution for extrinsic self-calibration of the sensors via SLAM. Further, we propose to create a catadioptric RGB-D sensor by combining an RGB-D camera with two planar mirrors. We let the mirrors split the field of view such that it covers opposite views, e.g., front and rear view of a mobile robot. For this sensor, we extend the extrinsic self-calibration to exploit the knowledge about the mirror arrangement to reduce the degrees of freedom of the problem. In experiments on real-world data, we demonstrate that incorporating multiple views yields substantial benefits for the accuracy of RGB-D SLAM. We evaluate the convergence and accuracy of the self-calibration approach and show the benefits of the reduced degrees of freedom approach.

. . . . .

RGB-D sensors based on the projection of structured light have leveraged many robotic applications. The dense depth information at high frequencies brought indoor perception to a new level. However, one of the major limitations of RGB-D cameras, particularly in the context of SLAM, is the small field of view. It is limited not only by the narrow opening angle, but also by the minimum and maximum range. In many applications it is therefore desirable to use more than one sensor to increase the observable area. For example, in RGB-D SLAM with a single sensor, the estimation of the motion becomes difficult if the robot is too close to obstacles, therefore lacking depth information. On the other hand, in large empty spaces, such as a hall it may lack reliable depth information because few things are within the maximum range. Further, indoor environments often contain areas of high ambiguity in geometry and texture, e.g., only flat monochrome surfaces with right-angled edges. In such environments, a smaller effective field of view means a lower probability to perceive salient structures or texture. For handheld SLAM the human can deal with this problem by taking care to avoid these situations. For an autonomous robot, however, this is not straightforward, as it does not yet have the required knowledge about its environment to plan a specialized path. With multiple sensors, the robot is less prone to such situations. On the downside, multiple sensors come at a cost. Besides the additional financial cost, also weight and power consumption are multiplied.

However, many robots are already equipped with multiple sensors for safety reasons, e.g., for dynamic collision avoidance, where the robot is usually required to perceive the accessibility of its path. If the robot can go forward and backward – or is even

omnidirectional – this cannot be solved with a single RGB-D sensor. If multiple sensors are available, it is desirable to incorporate them in the mapping process.

To fuse the observations of multiple sensors, they need to be projected to a common coordinate system. This requires to determine the relative poses of the sensors, i.e., to calibrate them extrinsically. This is often a tedious procedure, either approached by manual measuring or by using common observations of a calibration pattern. The latter procedure requires an overlapping field of view, which would be an unfortunate configuration for RGB-D cameras, as the interference of the projected infrared patterns deteriorates the depth measurements.

In this chapter, we present our work on the subject of multi-view RGB-D perception. In brief, our contributions are

- an extension of the system presented in Chapter 3 to perform SLAM jointly with the measurements of two RGB-D cameras and the evaluation of the impact on the performance.
- A catadioptric extension for RGB-D cameras to split the field of view of a single camera, attaining similar benefits of using two cameras with only low additional cost.
- A novel approach to self-calibration of multiple viewpoints of RGB-D cameras via SLAM, using the egomotion estimation of each sensor and, if available, common landmark observations.
- The introduction of additional constraints for this self-calibration approach, which exploit the knowledge about the structure of the catadioptric extension, making the calibration of the views quicker, more robust and allowing calibration from planar motion only.

4.1 SLAM with Multiple RGB-D Sensors

Performing localization or SLAM with consumer grade RGB-D cameras as discussed in Chapter 3 has been a topic of intensive research in recent years [21, 38, 56, 125]. Many approaches are adaptations of algorithms commonly used with stereo cameras or lidars (laser range scanners). Lidars typically have a horizontal field of view of 180° to 360° degrees and usually provide accurate depth measurements in a range between a few centimeters and several dozen meters. In contrast, for the most popular RGB-D sensors, i.e., the Microsoft Kinect and the Asus Xtion pro live, the distance in which the sensor provides depth measurements is generally limited to a range of about 0.6 m to 8 m with a rather restricted field of view of 43° vertically by 57° horizontally. Whereas for lidars, one can usually assume some meaningful geometric structure to be in the

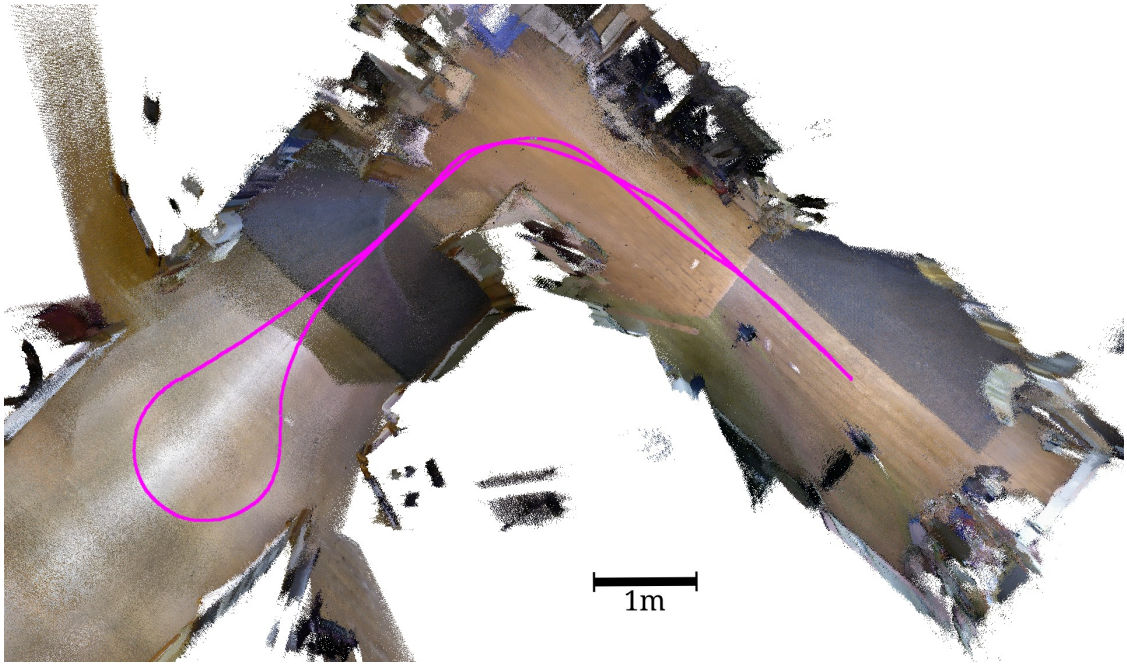


Figure 4.1: Top-view of the environment of the experiment with the trajectory of the robot.

sensor's field of view (e.g., more than half of the room for a field of view of 180°), in the context of RGB-D cameras we often need to deal with (visually and geometrically) ambiguous structure, e.g., when perceiving only a flat part of a wall or one corner of a room. Accordingly, an extension of the field of view of RGB-D sensors would be highly beneficial. The easiest way to achieve this is by using multiple sensors.

We therefore extended the approach discussed in Chapter 3 to investigate the impact of a second RGB-D Camera. With available extrinsic calibration, this can be achieved in a straight-forward manner by processing the frames from the cameras in turn. We modified the match-making algorithm from Section 3.4 such that frames from the other sensors will not be considered for the n direct predecessors. Instead we include an edge representing the extrinsic calibration between the respective sensor's nodes. Because current RGB-D cameras cannot be synchronized, we increase the uncertainty of the calibration to reflect the difference in the time stamps.

Experimental Evaluation

To quantify the performance gain from using a second RGB-D sensor in a typical SLAM application, we mounted two RGB-D cameras back to back on a Pioneer robot and drove it through an indoor office environment. The environment and the path of the robot are shown in Figure 4.1. The trajectory of the robot starts and ends in the same position with opposite bearing, as would be encountered in a fetch-and-carry task. As autonomous

robots usually need to monitor their path, we oriented the sensors to look forward and backward. While the forward-only field of view of the robot greatly overlaps between going back and forth, the perspective is substantially different. When using both views, the features perceived in the first part of the trajectory are seen from a similar perspective by the respective other sensor on the way back.

We reconstruct the trajectory of the robot offline from a recorded dataset, using the system presented in Chapter 3, once using only the forward-facing sensor and once using both. We do not use any further sensing for the trajectory reconstruction. We compare the respective trajectories with ground truth obtained from 2D Monte-Carlo localization using a SICK LMS-200 laser scanner. To compute the deviation from the ground truth, we use the root-mean-square of the absolute trajectory error (ATE-RMSE), as described in Section 3.2.2. For a trajectory estimate $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_1 \dots \hat{\boldsymbol{x}}_n\}$ and the corresponding ground truth $\boldsymbol{X}$ the ATE-RMSE is defined as

$$e_{\text{ATE-RMSE}}(\hat{\boldsymbol{X}}, \boldsymbol{X}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \|trans(\hat{\boldsymbol{x}}_i) - trans(\boldsymbol{x}_i)\|^2}, \tag{4.1}$$

i.e., the root-mean-square of the Euclidean distances between the corresponding ground truth and the estimated poses. To make the error metric independent of the coordinate system in which the trajectories are expressed, the trajectories are aligned such that the above error is minimal. The correspondences of poses are established using the timestamps from the sensor image and the range scan.

The robot's path is about 18 m long. The translational root mean squared error (ATE-RMSE) using only the front-facing camera for SLAM is 0.209 m. Incorporating additionally the information from the back-facing sensor, the error with respect to the ground truth is only 0.070 m, a substantial reduction of approximately two thirds. The respective trajectories are visualized in Figure 4.2. This result strongly suggests the use of multiple sensors, where possible.

4.2 A Catadioptric Extension for RGB-D Cameras

As demonstrated in Section 4.1, a substantial improvement can be achieved by the use of multiple sensors. This, however, comes with the inherent increase of weight, power consumption and financial costs. In this section, we propose a novel catadioptric setup for Kinect-style RGB-D cameras that is of low cost (less than 15 €), and requires neither significant computational resources nor higher power consumption and substantially relaxes the limitations from the small field of view.

Catadioptric sensors have been extensively used in the robotics and computer vision community for localization, visual odometry and SLAM [99]. Various shapes of mirrors have been used to increase the field of view, including parabolic, hyperbolic and

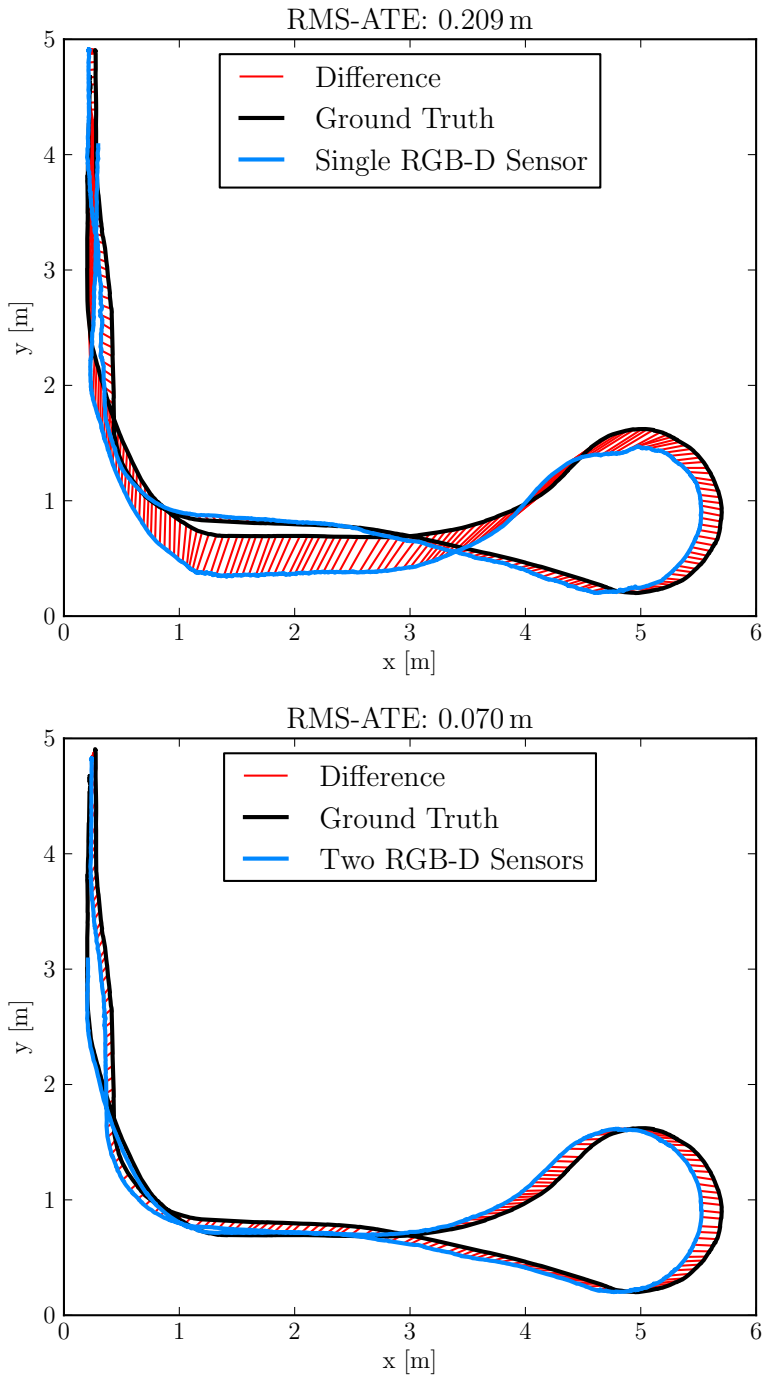
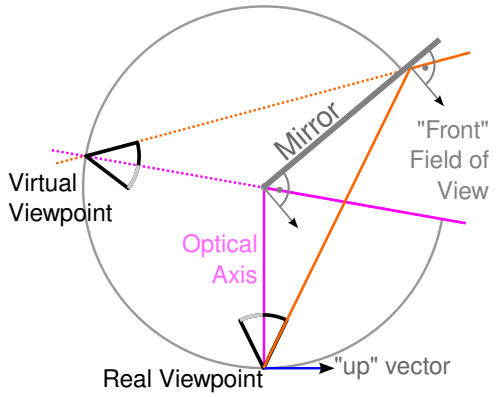
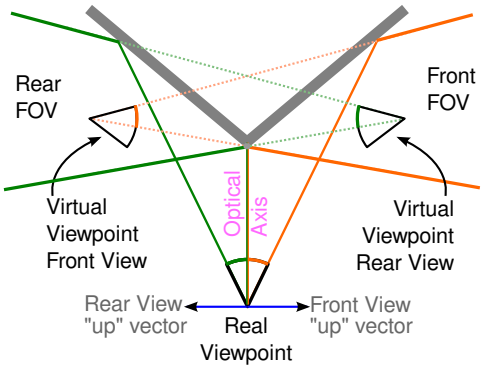


Figure 4.2: Comparison of the trajectory reconstruction of RGB-D SLAM using a single front-facing sensor (top) with the use of two sensors facing front and back (bottom). The additional information reduces the error by 66.5 %.



(a) Creation of a virtual viewpoint. For clarity, only one mirror and the respective projection of the viewpoint is shown.



(b) For two planar mirrors, the focal point is projected to two virtual viewpoints. The "up" vector needs to be flipped for the lower part of the original image.

Figure 4.3: Illustration of the concept of the virtual viewpoints.

spherical [128]. Also, mirrors have been used to capture stereo images with a single camera [47]. To the best of our knowledge, however, we are the first to propose a combination of RGB-D cameras with mirrors.

4.2.1 Design

The most influential design choice of a catadioptric sensor is the shape of the mirror(s).

While omnidirectional (non-RGB-D) cameras with parabolic mirror have been proven beneficial in many applications [128], the concept cannot be easily transferred to RGB-D cameras. The main benefits of parabolic mirrors are the 360° field of view perpendicular to the optical axis and the good compromise between detail in the close range and a big vertical field of view (if the optical axis is vertical).

An RGB-D sensor consists of an infrared projector, an infrared camera and a color camera. The infrared camera perceives the projected pattern and computes dense distance values from the disparity of the projected and perceived patterns. Due to the alignment of the infrared projector and camera, the disparity can be efficiently computed, by searching for a projected point in a small horizontal stretch of the infrared image. Using parabolic mirrors would come with many difficulties. Most importantly, it would require three mirrors. Hence, some parts of the field of view would not be visible to all optical devices to due occlusion by the neighboring mirrors. Measures to avoid crosstalk through the reflection of the neighboring mirrors would be required. Further, it would be computationally more expensive to compute the disparity image, as one would need to compensate for the curvature introduced by the mirrors of both infrared devices. In contrast, when using a planar mirror, the mode of operation of the camera is not affected. We therefore propose

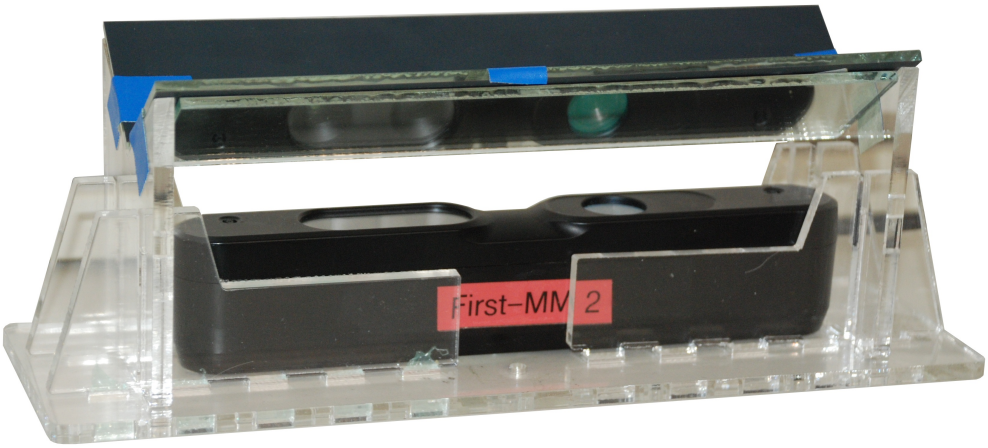


Figure 4.4: The assembled catadioptric sensor using an Asus Xtion PRO Live, two mirrors and a mount made of transparent acrylic sheet. Figure 4.5 depicts the CAD model of the mirror mount.

to split the field of view of the camera, with two planar mirrors, creating two *virtual viewpoints* roughly opposite to each other. Figures 4.3a and 4.3b illustrate the formation of the virtual viewpoints. Further splits might be beneficial in specific scenarios to allow the robot to measure other aspects of its environment, e.g., floor or ceiling. Aligning the edge between the mirrors with the horizontal axis of the RGB-D camera minimizes the loss of depth information such that almost the full sensor resolution can be used. Setup this way, only the visibility of the edge where the mirrors are linked will negatively affect the image. In our experiments we created two field of views of about 20° vertically by 57° horizontally. Pointed in opposite directions, the two field of views are particularly beneficial for robots moving in the plane, as the sensor data gained by the horizontally extended perception yields much more information relevant to the planar motion than the sacrificed perception in the vertical direction. Figure 4.4 shows the prototype of our proposed catadioptric sensor. However, the proposed sensor extension can be also beneficial for robots moving with more degrees of freedom as, e.g., the ambiguity between translation and rotation is alleviated. As a further advantage of the proposed setup, the intrinsic calibration of the device is not changed, so that one can apply the standard procedure or use the factory default settings. We choose the tilt of the two mirrors such that the centers of the two half fields of view are roughly horizontal. Note that ideal planar mirrors do not introduce any distortion to the depth perception principle, regardless of their pose.

The choice of the distance between camera and mirrors is a trade-off between loss of image area and the size of the device. The closer we mount the mirrors to the camera, the larger will be the unusable projection of the gap between the mirrors be in the images. If they are further away, the mirrors need to be larger, and the size and weight of the overall device grows. In our implementation the mirrors are about 2 cm in front of the camera

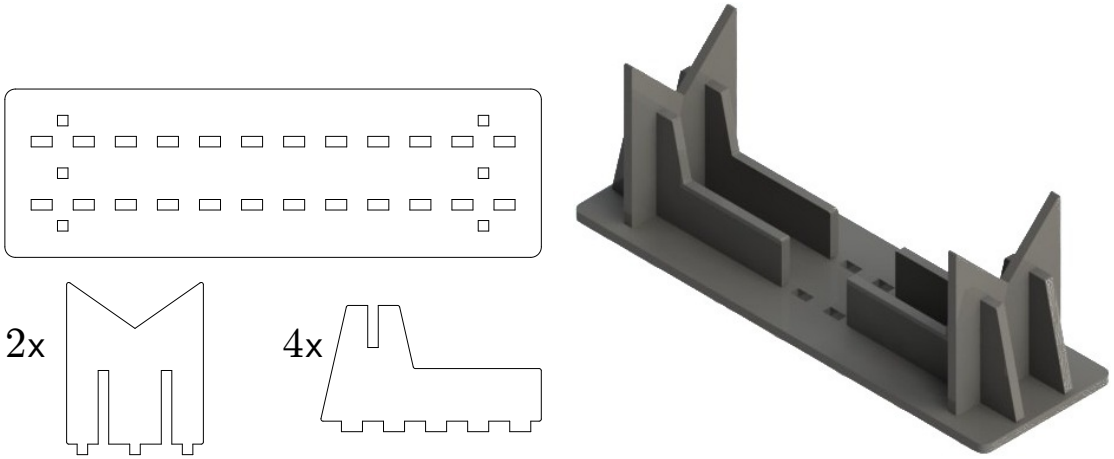


Figure 4.5: The CAD model of the mirror mount for an "ASUS Xtion PRO Live". The model is available online[1]. The structure can easily be created with a laser cutter from a 30 cm × 30 cm × 0.5 cm sheet of acrylic glass.

and we crop out a rectangle with a height of 50 pixels from the center of the image. Note that our goal here is a setup that is generally applicable. We therefore do not address, e.g., the possible reduction of the minimum sensing distance with respect to the center of the robot. While this could easily be achieved by increasing the distance between mirrors and camera, it would come at the cost of equally reduced maximum range and, as mentioned above, a bigger size of the device. To hold the mirrors and the sensor, we designed a multi-component structure which we can easily assemble by manually plugging the individual components together. Figure 4.5 shows the CAD model of the structure, which we used to cut the pieces from a 5 mm thick sheet of polymethylmethacrylate (PMMA) in a laser cutter. While it would be possible to use coated PMMA for the mirrors as well, we recommend to use glass mirrors. Due to the great planarity and rigidity of glass no noticeable distortions are introduced to the images.

To readily offer the developed device to the research community, we publish the used CAD model[1].

4.2.2 SLAM with the Catadioptric RGB-D Sensor

To use the catadioptric sensor for RGB-D SLAM, we can use the exact same procedure as described in Section 4.1. However, the two virtual views originate from the same sensor image, which means they are synchronized (except for a minor difference due to the rolling shutter). This allows us to use the extrinsic calibration to project the visual features of both views into a common local coordinate system. This has the advantage that more matches are available for each transformation estimation. Particularly, having

[1] see http://ais.informatik.uni-freiburg.de/projects/datasets/catadioptric-rgbd/

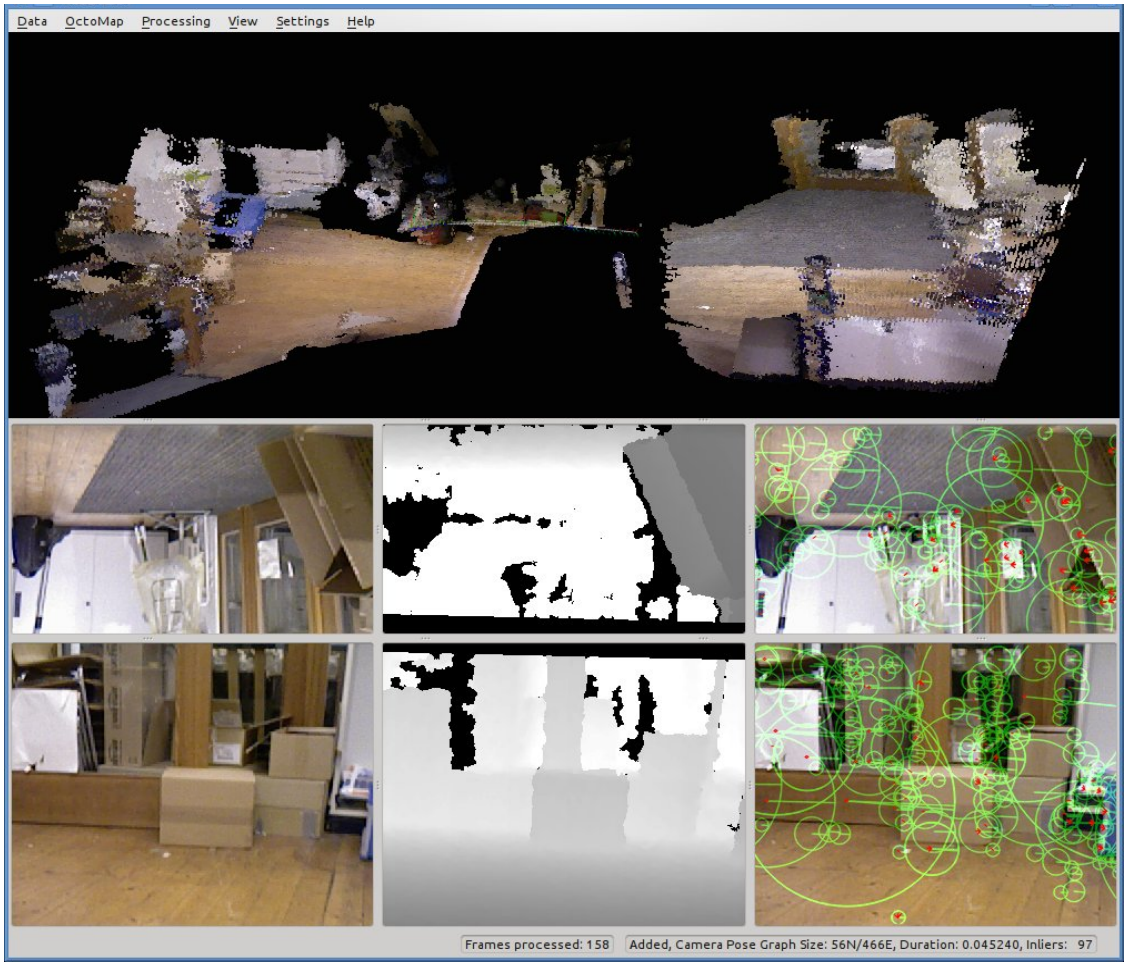


Figure 4.6: The RGB-D SLAM software processing input from the catadioptric sensor. Top: 3D model created from both views using the estimated calibration. Bottom rows: color and depth input for the two views and the respective visual features.

matches on both sides of the robot makes the difference between translation and rotation less ambiguous (when the robot translates parallel to the image plane, the features on both sides will move towards the same direction; if it revolves, they will move to opposite directions). Figure 4.6 shows the adapted software processing the split views.

To quantify the performance gain from using the proposed sensor extension, and compare it to the results of Section 4.1, we use the same experimental setup as before. We compare the trajectory obtained from our RGB-D SLAM system with data from the catadioptric sensor to ground truth, which we obtain from our pioneer robot. The robot is driving through the same indoor environment shown in Figure 4.1 and records the ground truth from 2D Monte-Carlo localization using the SICK LMS-200 laser scanner. Figure 4.7 depicts a Pioneer robot with the proposed catadioptric sensor and a map created from the data obtained from the catadioptric RGB-D sensor.

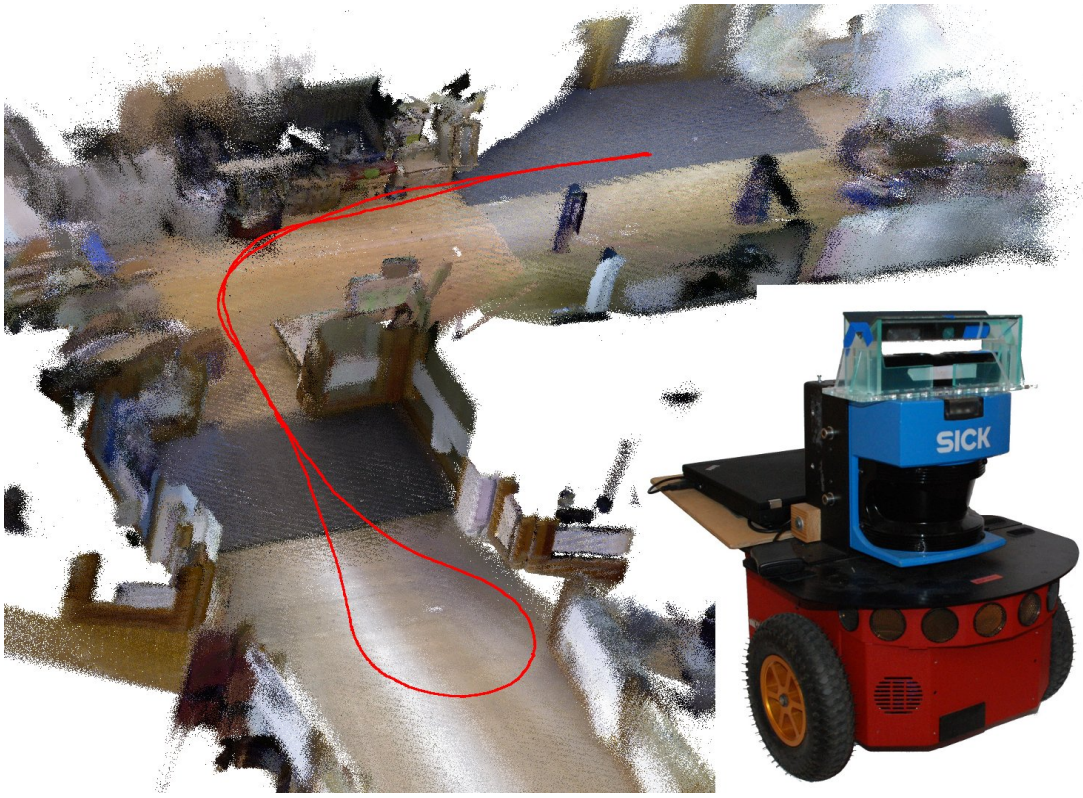


Figure 4.7: A resulting map and the trajectory estimate from our evaluation experiment, where we show that the proposed catadioptric sensor substantially improves the accuracy in a robot SLAM task. Note that only the RGB-D data was used to generate the map, while the laser range scanner and the odometry were used to obtain the ground truth.

To compute the deviation from the ground truth, we use the root-mean-square of the absolute trajectory error (ATE-RMSE), as described in Sections 3.2.2 and 4.1. We also compare against the results from Section 4.1. Ideally, we would compare the accuracy of the SLAM trajectory estimate from the same motion. However, RGB-D cameras actively project an infrared pattern which can lead to a crosstalk effect between the sensors. This effect has been shown to substantially deteriorate the measurements, particularly if the sensors are placed in close proximity [86]. We therefore refrain from running the catadioptric setup at the same time as the two regular sensors.

To guarantee comparability, we use the highly accurate laser-based trajectory following approach by Sprunk et al. [111], for which we teach a trajectory and let the robot repeat it, once with the two regular RGB-D cameras and once with the catadioptric device. For ground truth, we used the laser scanner to create a 2D map in the “teach” run and use the laser-based localization result for comparison in each repetition run. In our experiments, the deviation of the mobile robot from the taught trajectory was always below 0.02 m and 2.5°. Therefore, the input to both sensor setups is suited for an unbiased comparison. The

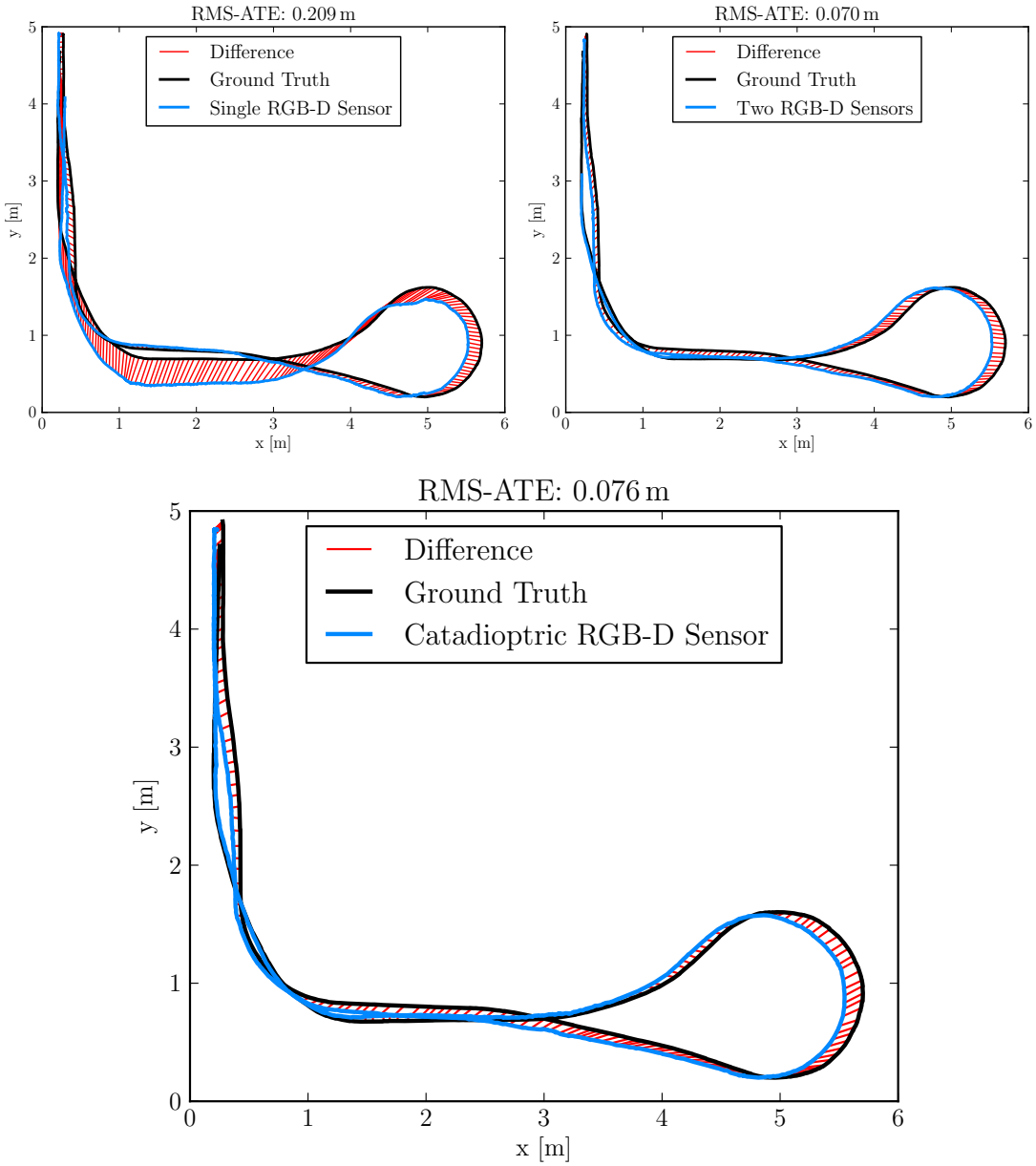


Figure 4.8: For a trajectory of about 18 m the ATE-RMSE obtained when using the proposed catadioptric RGB-D sensor (bottom panel) with respect to the ground truth is only 0.076 m. For comparison, in our earlier experiment (top panels), using one and two regular RGB-D cameras, we obtained 0.209 m and 0.070 m respectively. A larger version of the top panels can be found in Figure 4.2.

trajectory and the mapped environment are depicted in Figure 4.7. The ATE-RMSE for the catadioptric sensor is 0.076 m. In comparison, the corresponding error for mapping with the regular sensor, is 0.209 m. Thus the error is reduced by 64 % , which is almost as good as the result we obtained using two regular RGB-D cameras, where we obtained an error of 0.070 m. Figure 4.8 shows the trajectory from the respective RGB-D SLAM results in comparison with the ground truth.

4.3 Calibration of Multiple RGB-D Sensors via SLAM

To combine the perception of the individual viewpoints in the previous sections, we relied on the availability of their extrinsic calibration. In this section we present an approach to incorporate the extrinsic calibration into the graph optimization backend of the SLAM framework, such that it can be carried out during mapping – even online, if desired. In contrast to pattern-based calibration, our approach does not require an overlapping field of view or special tools. Our approach is able to perform a full calibration only from the egomotion estimates of the sensors. Additionally, common observations, e.g., after turning the sensors, are incorporated to constrain the calibration parameters. The backend, as described in Section 3.1.3, performs non-linear least-squares to estimate the maximum likelihood configuration of the sensor poses. Let us recapture the form of the least-squares problem:

$$\boldsymbol{F}(\boldsymbol{X}) = \sum_{ij \in \mathcal{C}} \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})^\top \, \boldsymbol{\Omega}_{ij} \, \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij}), \tag{4.2}$$

$$\boldsymbol{x}^* = \operatorname*{argmin}_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}). \tag{4.3}$$

As in Section 3.1.3, $\boldsymbol{X}$ is the trajectory, representing the individual poses $\boldsymbol{x}_i$ of the pose graph and $\mathcal{C}$ is the set of constraints (edges) in the graph. The error function $\mathbf{e}(\cdot)$ computes the difference between an expected measurement given the current state vector and a real measurement $\boldsymbol{z}_{ij}$. The error is $\mathbf{0}$ when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ perfectly explain the measurement. Finally, the information matrix $\boldsymbol{\Omega}_{ij}$ models the uncertainty in the error.

The measurements $\boldsymbol{z}_{ij}$ describe relative transformations between the camera poses at certain points in time. The transformation may relate subsequent poses, i.e., visual odometry, or span over large time intervals (loop closures).

We can elegantly include the unknown extrinsic calibration into this formulation. First, by matching frames from different sensors. When one camera observes a part of the environment that was previously seen in another camera, the above error function involves two different cameras and will be able to estimate their relative state accordingly, assuming uninterrupted motion estimates between the respective times of observation. Further, we define a second set of error functions that represents the calibration parameters cor-

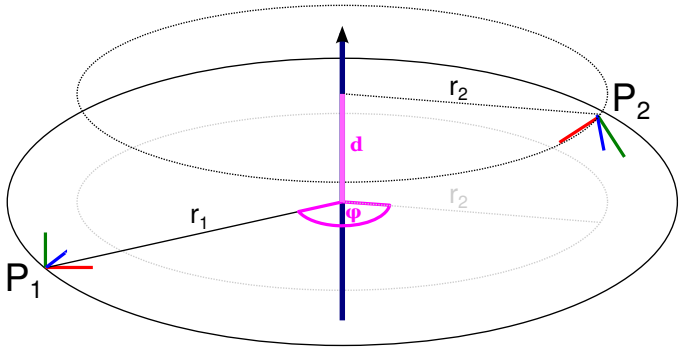


Figure 4.9: When rotating the sensors P1 and P2 jointly about the depicted axis, the degrees of freedom d and ψ (shown in magenta) are not observable from this motion.

responding to the setup of our mirrors. Without any further assumptions the calibration consists of estimating the offset $\boldsymbol{c} \in \mathrm{SE}(3)$ between the two cameras. Without loss of generality let us assume that the state vector contains ordered pairs of poses for the camera in each time step, i.e., $\boldsymbol{x} = \langle \boldsymbol{x}_1^{[1]}, \boldsymbol{x}_1^{[2]}, \ldots, \boldsymbol{x}_t^{[1]}, \boldsymbol{x}_t^{[2]} \rangle$, where $\boldsymbol{x}^{[j]}$ represents the poses of the camera j. The error for the offset parameter $\boldsymbol{c}$ is given by

$$\mathbf{e}_i(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{x}_i^{[2]} \ominus \left(\boldsymbol{x}_i^{[1]} \oplus \boldsymbol{c} \right), \tag{4.4}$$

where $\oplus$ corresponds to the motion composition and $\ominus$ to its inverse.

Given these two types of error functions, the joint estimation of the trajectory and the calibration is done by solving

$$\underset{\boldsymbol{x},\boldsymbol{c}}{\operatorname{argmin}} \left(\sum_{ij \in \mathcal{C}} \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})^\top \, \boldsymbol{\Omega}_{ij} \, \mathbf{e}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij}) + \sum_t \mathbf{e}_t(\boldsymbol{x}, \boldsymbol{c})^\top \, \boldsymbol{\Omega}_t \, \mathbf{e}_t(\boldsymbol{x}, \boldsymbol{c}) \right). \tag{4.5}$$

Note that while we only consider two cameras, the approach itself is able to handle multiple cameras. This latter type of constraint allows us to estimate the calibration from the egomotion estimate of the individual sensors. Intuitively, if two cameras are rigidly attached and perform a translational motion, we know that the motion vectors need to be parallel. This constrains the angle of the optical axes of the cameras to the motion vector, but not around it. The translational offset between the sensors is not observable from such a motion. If the sensors jointly rotate about an axis, we can observe the angle to the axis and the offset orthogonal to it, but not their relative rotation about the axis or offset along it. See Figure 4.9 for an illustration. To achieve full calibration of all degrees of freedom without loop closures across sensors, the sensors therefore need to be rotated about at least two axes. We evaluated the calibration from both types of constraints individually using two Asus Xtion Pro Live sensors mounted back-to-back, with the second one flipped upside down. We pivoted the sensors on a tripod in the middle of a small office and used

SLAM to determine the calibration parameters. The chosen arrangement of the sensors allows to accurately align the lenses, up to the variance in fabrication. The expected extrinsic offsets are zero, except in the rotation about the cameras' horizontal axis (the yaw angle in the plot), which should be 180°, and their relative translation along the optical axis (the y axis in the plot), which should be about -2 cm (we could not measure the exact offset between the focal points manually). The convergence of the individual degrees of freedom is shown in Figure 4.10. Note how the rotation about the roll (on the left side) is not sufficient to constrain the rotational calibration about that axis. As visible in the center column though, the constraints of the egomotion allow to effectively calibrate the sensors if the sensor rotates about more than one axis. In the experiment, we used the factory settings for the intrinsic calibration. However, we found that in some Xtion Pro devices the scaling of the depth has a systematic bias of up to 10 %. Therefore we performed a quick calibration of the depth scaling.

4.4 Calibration of the Catadioptric RGB-D Sensor

In this section we discuss the extrinsic calibration of the presented catadioptric RGB-D sensor. In principle, the calibration procedure from Section 4.3 can be directly applied to the input from the catadioptric sensor, to estimate the position of the two virtual viewpoints created by the mirrors. However, due to special properties of the proposed catadioptric device, the offset between the virtual camera viewpoints is constrained approximately to a manifold of less than six degrees of freedom. We can exploit this knowledge to reduce the degrees of freedom, which improves the convergence properties of the calibration.

Regarding the intrinsic calibration, we assume no additional distortion from the mirrors, due to the small size of the used mirrors and the great planarity and rigidity of glass. In our experiments we found that this assumption holds in practice. Therefore, the intrinsic calibration procedure does not differ from that of a regular RGB-D camera.

4.4.1 Reduction to Three Degrees of Freedom

As stated above, we mount the mirrors and the RGB-D camera such that the edge between the mirrors is aligned with the transversal axis (the axis going through the projector and the two lenses). When the mirror mount is produced with a laser cutter, the error in this alignment is negligible for our practical purpose. If we therefore assume that the edge between the mirrors is indeed parallel to the transversal axis of the camera, we obtain three degrees of freedom for our setup. The left side of Figure 4.11 illustrates the remaining degrees of freedom of this configuration. Under these assumptions the roll and yaw angles of the virtual viewpoints are identical. Further, the offset in the transversal axis must be zero. The remaining degrees of freedom are the relative pitch θ and the

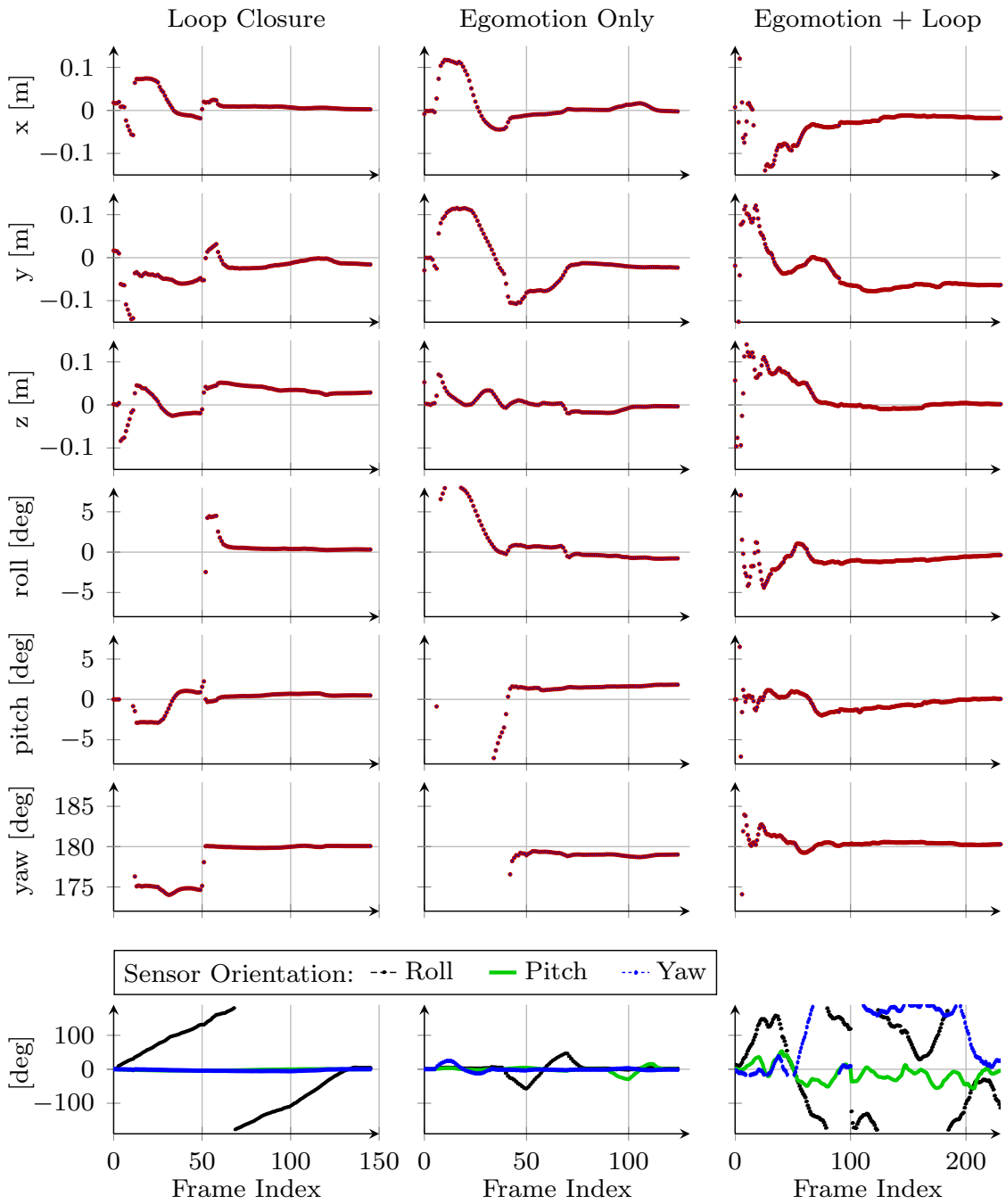


Figure 4.10: Convergence of the extrinsic calibration of two RGB-D sensors mounted back to back. The bottom row depicts the orientation of the camera facing to the front during the recorded motion. The rows above show the components of the computed relative transformation between the viewpoints. *Left*: Rotation only about the vertical axis (roll), with a loop closure between the viewpoints around frame 50. *Center*: Small rotation of the device around all axes, without loop closure. *Right*: Unconstrained motion with loop closures.

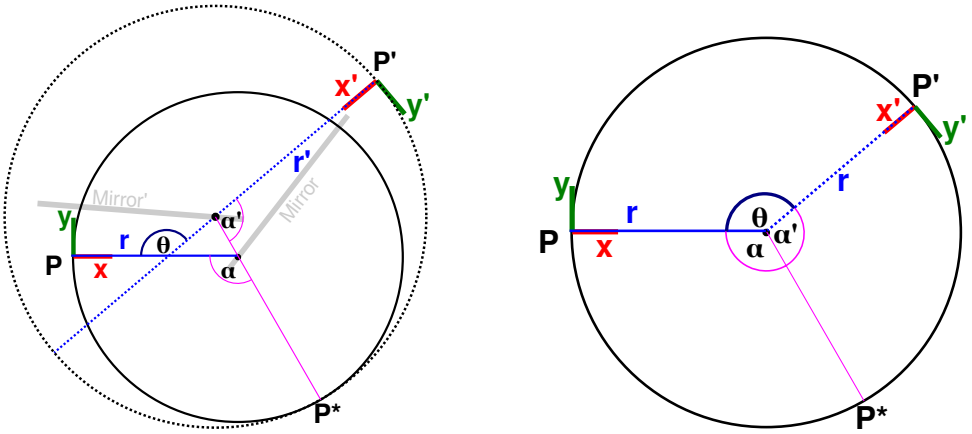


Figure 4.11: Sketch of the cross sections of the proposed catadioptric extension. P^* depicts the position of the real focal point, while P and P' depict the virtual focal points as projected by the two mirrors. *Left*: Assuming the intersection of the mirror planes to be parallel to the image plane (both orthogonal to the illustration) results in a displacement with three degrees of freedom (x', y', θ') between the virtual viewpoints. *Right*: Additionally assuming the edge between the mirrors to intersect the optical axis of the camera reduces the transformation between the virtual viewpoints to the two deg. of freedom r and θ.

translational offset $\boldsymbol{b} \in \mathbb{R}^2$. Thus, we obtain ${}^{\text{3DOF}}\boldsymbol{c} = \langle \boldsymbol{b}, \theta \rangle$ which we can transform to $\text{SE}(3)$ as follows:

$$ {}^{\text{SE(3)}}f_{\text{3DOF}}({}^{\text{3DOF}}\boldsymbol{c}) = \begin{pmatrix} \mathbf{R}_x(\theta) & (0, \boldsymbol{b})^\top \\ \mathbf{0} & 1 \end{pmatrix}. \tag{4.6} $$

The result of this conversion recovers the parameter $\boldsymbol{c}$ in its original space and can directly be plugged into Equation 4.5 to estimate the calibration.

This calibration method has the advantage that it is more robust to small variation in the placement of the camera with respect to the mirrors. The method even enables us to fully calibrate the virtual viewpoints with planar motions (which is impossible for two regular cameras) under the stated assumption, if we use the rotation to constrain the translational degrees of freedom and a translation to constrain the rotational offset. However, this means we would need to rotate about the axis along which the translation is known, which is the transversal axis - and therefore need to mount the device such that the wide aperture angle is vertical. Unfortunately, in a planar SLAM application using this configuration for online self-calibration would reduce the horizontal field of view by more than 62.5 %. Therefore the overlap of consecutive measurements would be substantially reduced when the robot rotates.

4.4.2 Reduction to Two Degrees of Freedom

Let us now assume that the intersection of the mirror planes is parallel to the transversal axis and it intersects with the optical axis of the projector and the lenses. We then obtain only two degrees of freedom for the virtual viewpoints, namely r and θ. See the right side of Figure 4.11 for an illustration of the remaining degrees of freedom.

Let ${}^{\text{2DOF}}\boldsymbol{c} = \langle r, \theta \rangle$ be the two dimensional parameter. We can convert this to a 4×4 transformation matrix $\in \text{SE}(3)$ as follows:

$$ {}^{\text{SE(3)}}f_{\text{2DOF}}({}^{\text{2DOF}}\boldsymbol{c}) = \begin{pmatrix} \mathbf{R}_x(\theta) & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{4.7} $$

where $R_x(\theta)$ corresponds to a 3×3 rotation matrix that represents a rotation by θ around the x-axis and $\mathbf{t} = (r - r\cos(\theta),\ r\sin(\theta),\ 0)^\top$. Again, the result of Equation 4.7 can be directly considered in Equation 4.5 to perform the calibration.

Under the assumptions stated above, these two parameters can be determined from planar motion only, for any placement of the sensor, as the introduced constraints allow the computation of the unobserved degrees of freedom.

4.4.3 Experimental Evaluation

As in Section 4.3, there are two particularly interesting cases in the calibration of the proposed sensor. First, we will examine the convergence using egomotion alone. Second, we will evaluate the impact of a loop closure between the two virtual viewpoints. Additionally, there is a third case, interesting in particular for robots moving in the plane: the convergence when restricting the sensor to a planar motion. For the evaluation of the convergence, we recorded data while rotating the sensor about the optical axis, the transversal axis and the "up" vector of the real camera. Figure 4.12 shows the convergence of the individual components of the displacement. The initial guess for the optimization is the identity, i.e., both viewpoints are at the same place. For the methods with restricted degrees of freedom, we compute the shown displacement components as described in Section 4.4. As expected, on the left column we can see that only the two degrees of freedom calibration method is capable to compute a stable estimate for the displacement for a planar motion without loop closure (before frame 50). The three degrees of freedom calibration method does not converge, because the camera would need to rotate around the transversal axis. However, as discussed in Section 4.4, this would be an unfavorable configuration for a SLAM setting with a wheeled robot moving in the plane.

The loop closure between the virtual viewpoints at frame 50 introduces sufficient constraints on all degrees of freedom. For rotational motion about more than one axis we observe quick convergence for all methods, with slightly less stable estimates for the

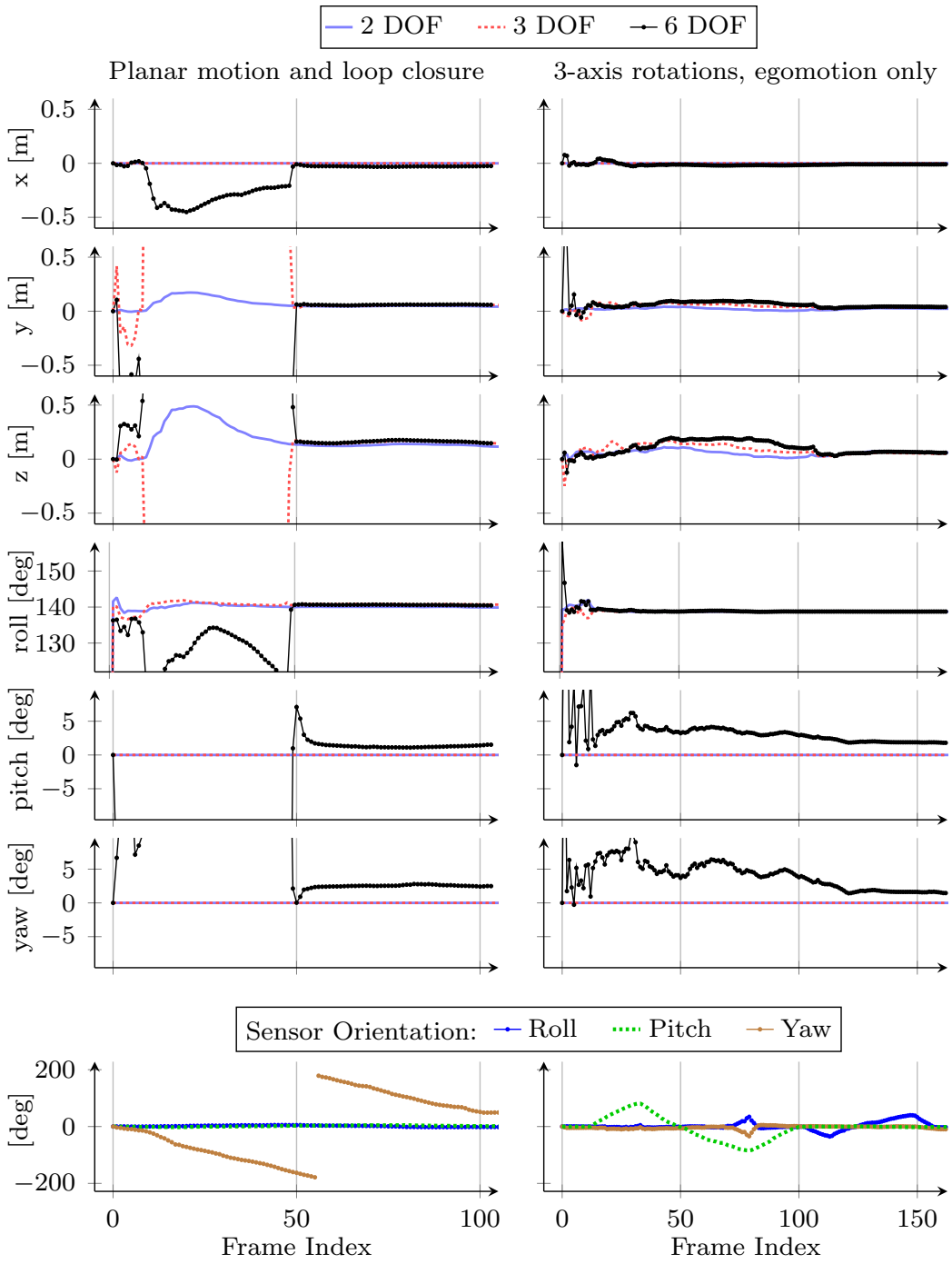


Figure 4.12: Convergence of the calibration parameters for the proposed methods. The bottom row depicts the orientation of the first virtual viewpoint during the recorded motion. The rows above show the components of the computed relative transformation between the virtual viewpoints. *Left*: Rotation only about the vertical axis, with a loop closure between the viewpoints around frame 50. *Right*: Rotation of the device around all axes, without loop closure.

methods with higher degrees of freedom. The estimated roll (the angle between the virtual cameras) corresponds well to the manually measured angle between the mirrors. Since the location of the focal point of the RGB-D camera is affected by the geometry of the lens and also inside the housing of the camera, we are only able to obtain a rough estimate of the translational components manually. The estimated translation of the virtual cameras is in line with what we expect given the approximate manual measurement. Furthermore, a visual inspection of the point cloud data of the catadioptric extension revealed the accuracy of the range data, for example, both virtual cameras observe the ground plane at the expected location and orientation. Using the parameters from the different calibration methods in the experimental setting from Section 4.2.2 showed no substantial impact on the trajectory error. This indicates that the remaining error in the extrinsic calibration of the views is negligible in SLAM compared to other sources of error.

4.5 Related Work

Transforming the measurements of multiple sensors into a common coordinate frame is called extrinsic calibration. This is often equivalent to the estimation of the relative poses of the sensors. A known extrinsic calibration, for example, allows us to transform the motion of one sensor into the motion of the other sensor. In principle, a device can be calibrated by precise manufacturing according to a model. For example, if the proposed catadioptric device is produced using the CAD model and a "perfect" laser cutter, the relative geometry of the mirrors is precisely known and can be used to compute the angular offset between the two virtual camera viewpoints. In practice, however, the extrinsic calibration often needs to be estimated. In the last decades several approaches for online calibration of range sensors and cameras [79, 80, 89, 96, 109] and multi-camera systems [14] have been proposed. Roy and Thrun [105] and Scaramuzza [109] proposed methods for online self-calibration of the kinematic model parameters for the odometry of a mobile robot using the onboard sensor. Kümmerle et al. [78] showed that the calibration of odometry and the according kinematic model parameters to an onboard sensor can be included in the graph optimization framework. Zienkiewicz et al. [132] automatically calibrate the position of the camera from visual odometry. Given the trajectory of a vehicle, Maddern et al. [84] perform an extrinsic calibration of LIDAR sensors by optimizing the Rényi Quadratic Entropy of the point cloud as the robot traverses the environment.

Brookshire and Teller presented an unsupervised approach that only requires egomotion estimates for calibrating the offset between range sensors either in 2D [11] or 3D [12]. To determine the extrinsic calibration of two (or more) cameras with an unconstrained rigid motion, we need to constrain all six degrees of freedom by appropriate motions. In their analysis, Brookshire and Teller [12] find that rotational motion around at least two different axes is required to obtain the required constraints. The calibration

procedures described in this chapter exploit both the egomotion and common observations to estimate the calibration parameters. Analogously to Kümmerle et al. [78] we incorporate these information sources elegantly as constraints in the graph optimization backend of our SLAM approach.

The calibration of the measurements of an individual sensor is called intrinsic calibration and concerns parameters that govern the image creation, i.e., focal length, principal point, and distortion model parameters. For RGB cameras, there are well-known techniques from the field of computer vision for estimating these parameters [54]. For an RGB-D camera we need to perform the intrinsic calibration for the color camera, the infrared camera, and the infrared projector. To this end, Herrera et al. [57] use a checkerboard to calibrate the intrinsics of such a camera. Recently, Teichman et al. [118] presented an approach to calibrate the depth-measurements of an RGB-D camera given a SLAM estimate.

4.6 Conclusions

In this chapter, we extended the mapping approach of Chapter 3 to multiple viewpoints. In a real-world experiment we found that the addition of a second RGB-D sensor can result in substantial benefits for the reconstruction accuracy. We presented a novel catadioptric extension to RGB-D cameras by using two planar mirrors to split the field of view such that it covers both front and rear view. In further experiments we demonstrated that this catadioptric extension yields similar benefits to a second sensor, at much lower cost.

We furthermore devised an approach for the extrinsic calibration of multiple viewpoints via SLAM, by introducing constraints for the estimation of the calibration parameters into the graph optimization backend. Further, we show that we can exploit the knowledge about the structure of the catadioptric sensor to introduce additional constraints that reduce the degrees of freedom of the estimation problem. Besides better convergence, this allows for full calibration of the viewpoints from planar motion, e.g., during operation of a wheeled robot.

Chapter 5

Interactive Perception and Manipulation of Doors

Contents

For a robot to navigate a typical indoor environment, opening of doors is a fundamental skill. Current state-of-the-art approaches for robotic door opening rely on a fixed grasp of the door handle. The grasp is required because these approaches are not aware of the dynamics of the door and therefore cannot predict what happens when the door is released in motion. In contrast, humans use prior knowledge and tactile perception to judge a door's dynamics. This allows humans to push or pull the door with only a point contact and release the door for switching the manipulating hand or changing the contact point. Inspired by these skills, we present a novel approach to learn a model of the dynamics of a door from observations with a depth sensor. The learned model enables the realization of dynamic door-opening strategies and reduces the complexity of the door opening task. These strategies reduce the degrees of freedom required of the manipulator and facilitate motion planning, as compared to approaches maintaining a fixed grasp. Additionally, execution is faster, because the robot merely needs to push the door long enough to achieve a combination of position and speed for which the door will stop at the desired state. Using our approach, the model of the dynamics can be learned from observing a human teacher or by interactive experimentation. We further present an approach to estimate the dynamics of an unknown door within the first interaction using tactile perception. We demonstrate the advantages in experiments on a real robot, where the robot precisely swings a door open to a desired opening angle.

. . . . .

In the previous chapters we investigated how a mobile robot operating in indoor environments can reliably create a map, e.g., as a basis for navigation. As most buildings are sectioned into rooms, which are separated by doors, opening doors is a fundamental skill to map and navigate in such environments. Current state-of-the-art approaches for robotic door opening typically use a gripper to obtain a firm grasp of the door handle to maintain control over the door while unlatching and opening it [13, 16, 67, 98]. For humans, however, there is a variety of ways to open a door, which we unconsciously choose and execute depending on the situation. For latched doors, we first need to turn a knob or handle. In the following opening motion, though, we generally do not maintain a firm grasp on the handle, as that would restrict our motion when passing through the doorway.

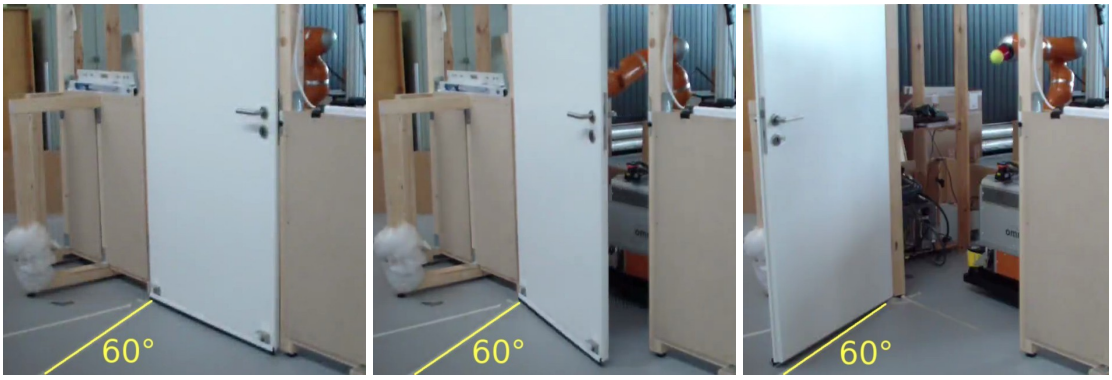


Figure 5.1: Execution of a door swing to 60°. Left: The robot starts pushing the closed door. Center: The robot releases the door when it predicts the momentum to be sufficient to reach the target state. Right: The final state of the door is very close to the marked target state. See Table 5.1 for quantitative results.

As we perceive the dynamics of the door in the course of opening, we can predict its trajectory when released. This ability makes us highly flexible during the execution. For instance, we can release the moving door as soon as its kinetic energy suffices to reach the desired state. Further, it allows us to dynamically switch the contact points and even the manipulating hand without interruption.

The aim of the work presented in this chapter is to advance robotic manipulation of doors towards the described flexibility by incorporating information from novel perception methods. More specifically, we want to learn a model of the dynamics of a door from perception and let the robot use this information during manipulation. This is in contrast to current state-of-the-art approaches to robotic door opening, which do not make explicit use of knowledge of the dynamics of the manipulated doors. Most approaches assume quasi-static motion, i.e., slow enough that inertial forces are negligible. This assumption substantially reduces the possible execution speed and the door needs to be released at rest. As a further consequence, the end effector needs to be in contact until the desired door state is reached. Therefore door opening with a fixed grasp is a challenging problem, as it requires a large dexterous workspace and requires high-dimensional motion planning, particularly when the base needs to be moved. In practice, this makes state-of-the-art approaches only feasible for robots with high reachability and many degrees of freedom.

In our experiments, we show that the learned model of the dynamics of the door enables a door manipulation approach in which the robot accelerates the door with its end effector exactly as long as required to swing it to a desired angle. This strategy represents the extreme case of the ability to release the door before it comes to rest. It therefore serves well as a showcase of the achievable accuracy of the predictions. Figure 5.1 shows an experiment where the robot executes a door swing to an opening angle of 60°. This

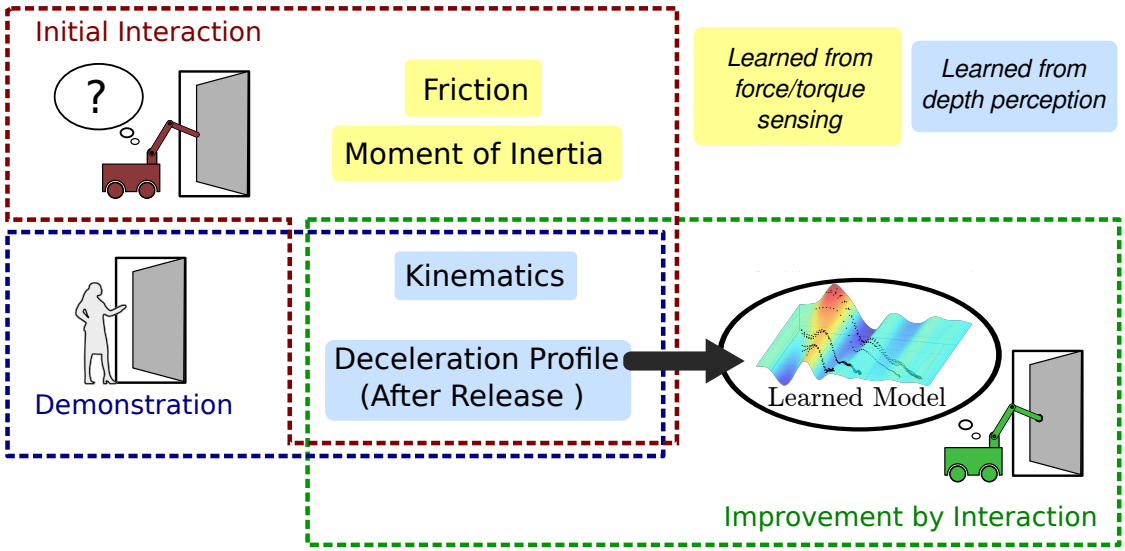


Figure 5.2: The proposed methods for learning the aspects of the dynamics model of a door allow the robot to learn from demonstration and to interactively learn by experimentation.

approach is applicable even on robots with few joints and limited reachability.

In the remainder of the chapter, we present our work on the perception and manipulation of doors in Detail. Our main contributions are as follows.

- We present a novel approach to **learn the kinematic model of a door from observation with a depth sensor**. The approach exhibits a quick approximate convergence, such that the kinematic model can be effectively used after observing the moving door only for a few degrees.

- We present an approach to **learn a model of the dynamics of a door from sensor observations** that allows to make accurate predictions of the door's dynamic behavior. The model can be used during the manipulation to predict the motion of the object at any time. Because the model captures the physical properties of the door, it generalizes over different starting conditions and desired stopping positions.

- We further propose an approach to **let the robot interactively bootstrap the model** during the first opening of an unknown door using tactile perception.

- We integrated the proposed perception and manipulation approaches on a real robot and **experimentally evaluate the resulting manipulation skills** in several door opening tasks. We demonstrate that the learned model of the dynamics can be used to open a door quickly with accurate results.

An overview of the proposed approaches is shown in Figure 5.2. In the following Section we will shortly recapture the physics of a moving door. In Section 5.2 we describe how

the robot learns the dynamics of the door from depth perception. Section 5.3 describes an approach to quickly obtain an initial estimate of the dynamics using tactile perception. Our experiments with a real robot are detailed in Section 5.4. In Section 5.5 we discuss related work, followed by a conclusion in Section 5.6

5.1 Articulated Object Dynamics

To predict the effects of the robot's actions on an articulated object we need to learn a model of the dynamic behavior of the manipulated object. In this section we will briefly review the physics on which the model is based and show how to efficiently learn the parameters for our physical model. For comprehensibility, we will concentrate on the physics of hinged doors at first, discussing the differences to the conceptually very similar model for sliding mechanisms in Section 5.1.2.

5.1.1 Rotational Motion

To predict the dynamic behavior of a door from its physical properties, assuming the standard model of friction, it is sufficient to describe the door by its *moment of inertia* I and the *kinetic friction* τ_f, which describes the resisting force due to friction within the door hinges when the door is moving.

According to the law of conservation of energy, assuming the friction torque is constant and neglecting other effects than friction, the reduction in rotational kinetic energy needs to be equal to the work of friction, i.e.,

$$\tfrac{1}{2} I \left(\omega_t^2 - \omega_{t'}^2\right) = \tau_f \left(\theta_t - \theta_{t'}\right), \tag{5.1}$$

where ω_t is the angular velocity at time t and θ_t is the corresponding opening angle. Note that the direction of friction forces is always opposite to the motion, i.e., τ_f always has the opposite sign of the current velocity ω. Given the angular velocity ω, we can compute the stopping distance $\Delta\theta = \theta_t - \theta_{t'}$ by setting $\omega_{t'}$ to zero,

$$\Delta\theta = \frac{I {\omega_t}^2}{2\tau_f} \tag{5.2}$$

This allows us to determine the position to release the door (i.e., stop accelerating it, if we do not maintain a grasp) during manipulation such that the door reaches the desired angle. Vice versa, we determine the required velocity at a given release angle θ_0 in order

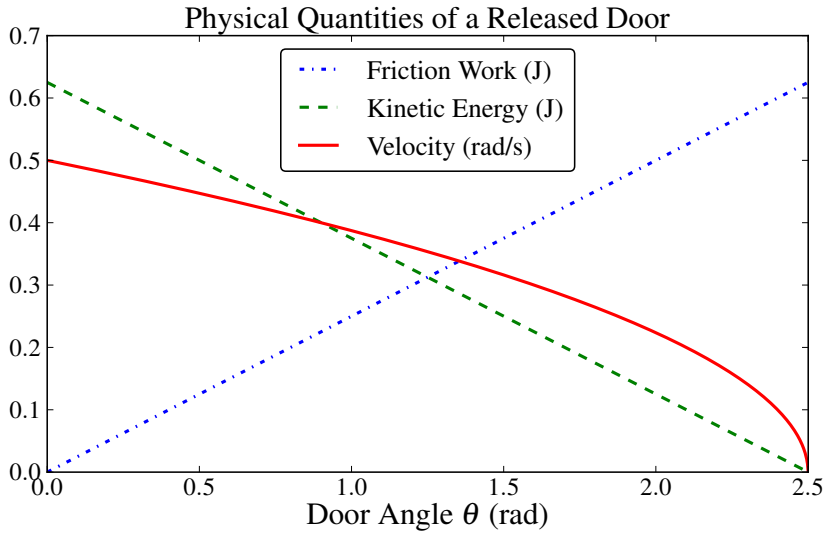


Figure 5.3: Door trajectories according to the standard friction model, assuming constant friction. See also the trajectory estimates from real data in Figure 5.5. Note that the velocity has a different physical unit than friction work and kinetic energy and will in general not be in the same scalar range. However, the depicted values are similar to those measured in our experiments.

for the door to stop at θ_T as

$$\omega = \sqrt{2\frac{\tau_f}{I}(\theta_T - \theta_0)} \tag{5.3}$$

Since I and τ_f only appear in form of the fraction $\alpha = \tau_f / I$ in Equation 5.2 and 5.3, we can also use the acceleration α instead. Given the current position θ_0, velocity ω_0 and the (constant) deceleration from friction, we can compute the trajectory of a door over time as

$$\theta(t) = \tfrac{1}{2}\alpha t^2 + \omega_0 t + \theta_0. \tag{5.4}$$

Figure 5.3 shows example trajectories generated with the described physical model for different starting velocities.

Note that the deceleration of the door is not only influenced by static and kinetic friction but also by other effects such as air drag and gravity (for non-upright doors). In this work we will summarize all these decelerating forces under the name friction.

5.1.2 Linear Motion

The physical models of sliding doors and other prismatic configurations such as drawers, is straightforward to adapt from the rotational case presented above. The equality of kinetic energy and friction work given in Eqn. 5.1 for linear motion is

$$\tfrac{1}{2}m\,(v_t^2 - v_{t'}^2) = F_f\,(x_t - x_{t'}), \tag{5.5}$$

where m is the mass of the object and v_t, x_t denote the linear velocity and position at time t. F_f denotes the friction force of the sliding mechanism. The stopping distance Δx for a given linear velocity v is thus computed by

$$\Delta x = \frac{mv^2}{2F_f}. \tag{5.6}$$

The required velocity at a position x_0 to reach $x_0 + \Delta x$ is

$$v = \sqrt{\frac{2F_f\Delta x}{m}}. \tag{5.7}$$

The kinetic friction of a sliding object can be found during constant-velocity motion as $F_f = F_{ee}^T\hat{\boldsymbol{x}}$, the dot product of the measured force vector and a unit vector along the sliding direction. Instead of the moment of inertia, we estimate the mass m during acceleration from

$$m = \frac{\int_0^T (F_{ee} - F_f)^T\hat{\boldsymbol{x}}\,\mathrm{d}t}{v_t - v_0}. \tag{5.8}$$

5.2 Perception of Doors with a Depth Sensor

As described in the beginning of this chapter, we want to learn a model of the dynamics of a door to be able to predict the effects of the actions of the robot. In this section we propose perception methods that allow the robot to learn how friction decelerates the door from observations with a depth sensor. With these methods, the robot can learn the model of the door either passively by observing other agents operating the door, or by perceiving it during interactive experimentation.

As a requirement for converting the position, velocity and applied force of the end effector of the robot to angular position, velocity and torques of the door, we need to obtain a precise estimate of the hinge location of the door. Sturm et al. [115] and others proposed methods to estimate the kinematics of articulated objects from sensor data, e.g., by tracking a fitted rectangle, a marker on the door, or the end effector pose. Here we propose an approach that requires only an arbitrary point and normal on the surface of the

door. This allows us to compute the geometry using only a planar depth sensor such as a laser range scanner or a single scan line from a depth camera. Without loss of generality, we assume the depth sensor to provide Cartesian point measurements $\{\boldsymbol{p}_i \in \mathbb{R}^2\}_0^N$ as obtained, e.g., by back-projecting the measurements of a horizontal laser range scan or projection of a point cloud from an RGB-D camera to the horizontal plane.

5.2.1 Estimating the Door State

We assume the robot is positioned in front of the door, e.g., by using the methods of Anguelov et al. [4], Limketkai et al. [81] or Rusu et al. [108], We then filter the measurements with a bounding box to make sure we observe the door only. The dimensions of the box depend on the uncertainty in the estimate of the robot's relative position to the door. We then apply a statistical filter to the remaining measurements to reject points far from their neighbors, e.g., as possibly obtained at the edge of the door.

To estimate the angle of the door, we subtract the mean $\overline{\boldsymbol{p}}$ from the remaining measurements and determine the door normal of the data set using the principal components analysis (PCA) [9]. In 2D, the normal $\boldsymbol{n} = [n_y, n_x]^T$ is given by the eigenvector with the smaller eigenvalue. We obtain the current state θ of the door from $\boldsymbol{n}$ as

$$\theta = \operatorname{atan2}(n_y, n_x) \tag{5.9}$$

5.2.2 Estimating the Hinge Position

Maximum Likelihood Estimation

From repeated observation of a moving door, we get several measurements for the mean point $\overline{\boldsymbol{p}}_j$ and the door normal $\boldsymbol{n}_j$. We track the direction of the door normal, to make sure that it always points in the same direction. After at least three measurements, we can apply a least squares optimization to compute the location of the hinge in closed form. We define the vector $\boldsymbol{h} = [h_x, h_y, h_r]^T$, where $[h_x, h_y]$ is the hinge location and h_r is the distance of the hinge to the line at the surface of the door. This parameterization explicitly allows the hinge to be non-collinear to the door surface, as there may be a substantial offset of the hinge orthogonal to the perceived surface. Each measurement must satisfy

$$[\boldsymbol{n}_j^T,\ 1]\boldsymbol{h} = \boldsymbol{n}_j^T \overline{\boldsymbol{p}}_j. \tag{5.10}$$

Defining the vector $\boldsymbol{g} = [\dots \boldsymbol{n}_j^T \overline{\boldsymbol{p}}_j \dots]^T$ and the matrix $\boldsymbol{M} = [\dots [\boldsymbol{n}_j^T,\ 1]^T \dots]^T$, we can directly compute the hinge parameters as

$$\boldsymbol{h} = \boldsymbol{M}^+ \boldsymbol{g}, \tag{5.11}$$

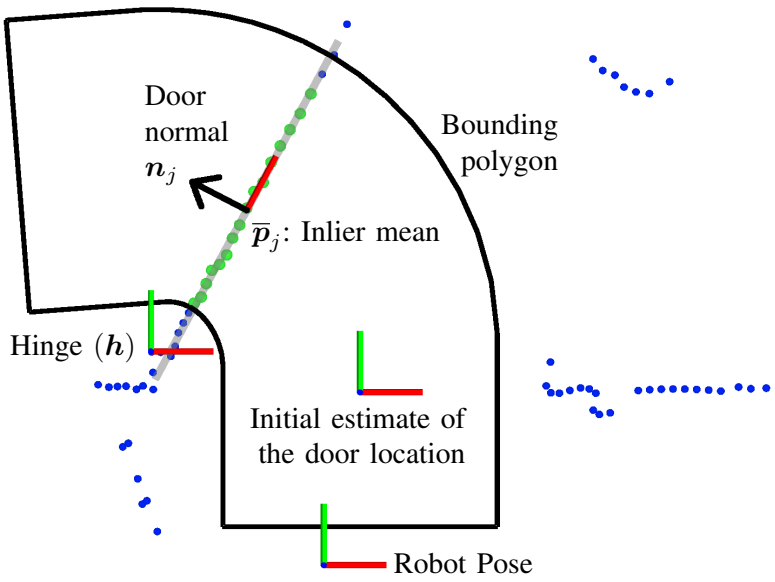


Figure 5.4: Online estimation of the door state and its kinematics from laser range scans (top view). Laser measurements are shown as blue (outliers) and green (inliers) dots. The bounding polygon is initialized as a small box at the initial estimate of the door location and expanded online.

where $\boldsymbol{M}^+ = \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}$ is the pseudo inverse.

To ensure tracking of the door over the full range of angles, we use the available state estimates to extrude the bounding box into a ring segment. Figure 5.4 illustrates the results after observing a full opening motion.

Accelerated Convergence

The described approach accurately determines the actual center of rotation, but requires observations of a variety of door states before converging. Observing a $90°$ door swing results in a highly precise estimate. However, for an observed motion of only few degrees, the estimate is far less accurate and stable than for the estimation algorithm of a collinear hinge as, e.g., [126], due to the higher degrees of freedom. If the robot interacts with an unknown door and we want to avoid the difficulties associated with whole body motion, i.e., the coordinated motion of base and manipulator, it is important to be able to get a good approximation of the hinge as early as possible. Assuming the workspace of the robot manipulator to be smaller than the required workspace for moving the door to the desired final state, the motion used for estimating the geometry will reduce the range available for accelerating the door to reach the desired state.

We therefore desire to stabilize the initial estimates. A common solution for this is the use of a prior. Unfortunately, a reasonable prior for the position of the hinge would be bimodal – with the modes on each side of the door plane – which cannot be represented

in the above formulation. We therefore use the degenerate measurement $\boldsymbol{n}_0 = [0, 0]$, with which the first constraint (Equation 5.10) becomes

$$[0,\ 0,\ 1]\boldsymbol{h} = 0 \tag{5.12}$$

$$h_r = 0. \tag{5.13}$$

This effectively sets a prior of zero on the radius h_r without constraining the hinge location parameters $[h_x, h_y]$. In practice, this makes the initial estimates as stable as in the collinear estimation case, yet the algorithm will still converge to the correct location without noticeable delay or deviation.

5.2.3 Learning the Dynamics

To learn the dynamic behavior of the door from depth perception, we use the door state estimation method from Section 5.2.1 to obtain the trajectory $\Theta = \{\theta_{t0}, \ldots, \theta_{tN}\}$ of the door and determine the deceleration of the door caused by friction. This is applicable both, when learning from human demonstration and during interactive experimentation. We want the model to generalize with respect to the starting position and velocity of the door, such that the robot can predict the motion of the door at any time. In practice, the deceleration of doors by friction and other effects may significantly change throughout the trajectory of the door. Hence, the total friction work over an angular distance changes depending on the point of release – and therefore the required kinetic energy for the desired stopping distance. We thus need to take the variation in deceleration over the course of the trajectory into account. To ensure generalization, we thus need to learn a *deceleration profile* $\alpha_f(\theta)$ of the door that depends on the opening angle.

Deceleration Estimation

Our measurements reflect the opening angle of the door. The deceleration is the second derivative of the angle with respect to time. However, direct numerical differentiation of noisy data amplifies the noise and therefore leads to unusable deceleration values. To be robust to the noise in the observations, we apply the regression techniques presented in Section 2.3.

For an unknown door, we have no generally applicable information on the global shape of the deceleration profile. However, as the forces which the door is subject to, e.g., friction in the hinge, air drag and gravity, typically do not change abruptly, we may assume that, locally, the deceleration is well approximated by a term constant or linear with respect to time. As a consequence from the equations of motion, we know that the door trajectory can then be locally approximated by a second or third order polynomial. We therefore apply the locally weighted regression approach described in Section 2.3.2,

which allows us to obtain the deceleration as one of the coefficients of the parametric model.

Given a trajectory Θ of N timestamped measurements θ_t we compute coefficients $\boldsymbol{x}$ of the locally fit curve at time t' by minimizing the squared error function

$$F(\boldsymbol{x}) = \sum\nolimits_t w_{t'}(t)\ (\theta_t - \Phi_t^T \boldsymbol{x})^2, \tag{5.14}$$

where $w_{t'}(t)$ is the tricube weight function centered at t' (see Section 2.3.2), and Φ_t is the vector of the polynomial basis functions, as defined in Section 2.3.1, Equation 2.42. Following Equation 2.50, the parameter vector $\boldsymbol{x}$ that minimizes the weighted squared error at t' is determined by solving the system of linear equations

$$\underbrace{\sum\nolimits_t \theta_t w_{t'}(t)\ \Phi_t^T}_{1\times d} = \boldsymbol{x}^T \underbrace{\left(\sum\nolimits_t w_{t'}(t)\ \Phi_t \Phi_t^T\right)}_{d\times d}. \tag{5.15}$$

The deceleration at t is then directly obtained as the corresponding component of $\boldsymbol{x}$. In our experiments, the assumption of locally static deceleration led to satisfactory results. Figure 5.5 shows two plots of the resulting decelerations and velocities for a set of demonstrated trajectories for a single door. The dependency of the deceleration on the position of the door is clearly visible in both plots. More subtle, but also noticeable is the dependency of the deceleration on the velocity, e.g., by the different magnitude of the variation in the lower plot.

Prediction from a Single Trajectory

The regression approach presented above allows us to compute the deceleration as a function of the angle, $\alpha_f(\theta)$, in the range of the observed positions. Based on this, we can predict the door trajectory given the current position θ_0 and velocity ω_0. By numerically integrating $\alpha_f(\theta)$, we can use Equation 5.3 to compute the future velocity at angle θ as

$$\omega^2 = \omega_0^2 + 2\sum_{i=1}^{N} \alpha_f(\theta_i)\,\theta_s, \tag{5.16}$$

where $\theta_s = \frac{1}{N}(\theta - \theta_0)$ is the discretization step size and $\theta_i = \theta_0 + (i + \frac{1}{2})\theta_s$ is the position where the deceleration is sampled for the ith discretization step. The stopping distance can be computed by summing until the right hand side becomes zero.

While the above approach is described only for hinged doors, it is directly applicable to the case of sliding doors, by exchanging the measurements and predictions of angle, angular velocity and deceleration to position, linear velocity and deceleration respectively.

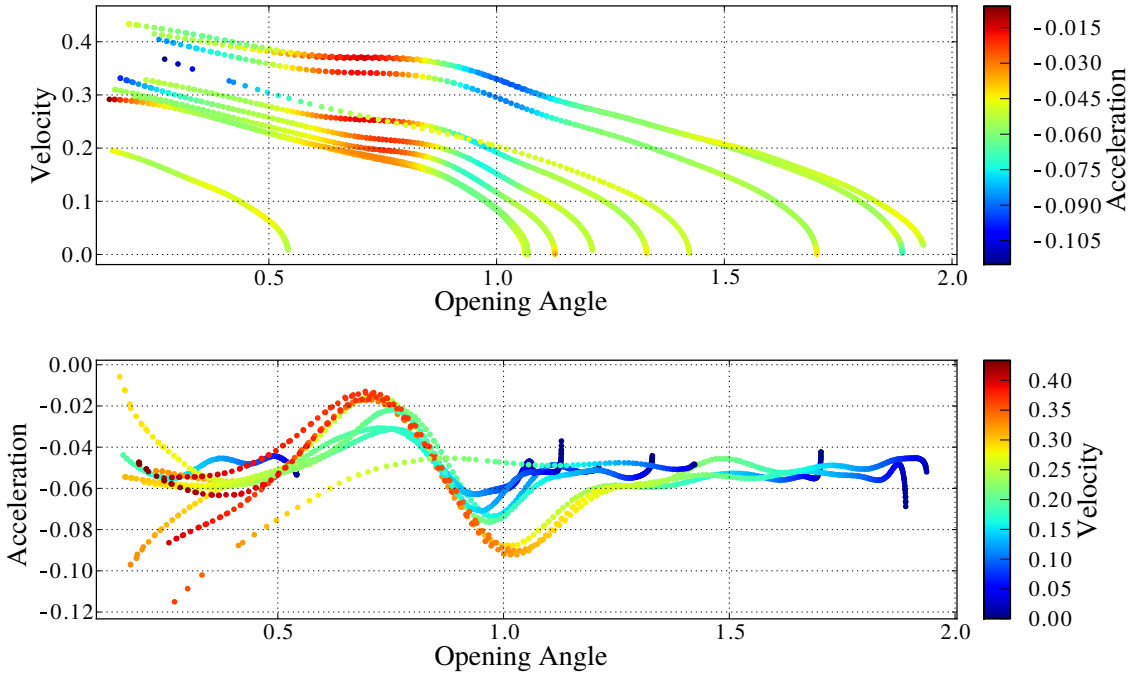


Figure 5.5: Opening angle (rad), angular velocity (rad/s) and angular acceleration (rad/s^2) extracted from several demonstrations with the same door. The variation of the deceleration with respect to the opening angle is clearly visible in both plots. Less pronounced, but also well visible is the dependency of the deceleration on the velocity. In the bottom plot this is manifest by the higher amplitude of the curve for faster motion.

Prediction from Multiple Observations

Using more than one trajectory observation to learn the model allows us to substantially improve our predictions. As visible from the experiments depicted in Figure 5.5, the velocity-dependent deceleration forces, i.e., viscous friction and air drag, are not negligible. Integrating several measurements in our deceleration profile allows us to model this dependency.

Again, we use regression to model the deceleration of the door from friction. However, in contrast to the estimation of the deceleration from the timestamped poses in the trajectory, the magnitude of the deceleration itself does neither follow a simple parametric model with respect to the opening angle, nor with respect to the velocity. While there are physical models of viscous friction and air drag, our experiments did not fit well to these. This might be due to dependencies on latent factors, e.g., room geometry, which substantially influences the air drag. However, even though we have no prior on the underlying process of the friction profile, we can assume that the deceleration values are highly correlated for similar conditions, i.e., inputs $\boldsymbol{y} = [\theta, \omega]^T$.

To learn the model of the deceleration $\alpha(\boldsymbol{y})$, we therefore use a Gaussian process (GP) [102]. A GP is a non-parametric regression method, see Section 2.3.3 for a detailed discussion. Instead of choosing a parametric model and its degrees of freedom, Gaussian processes let us directly specify the prior for the model in terms of the correlation between input locations. Further, GPs have a beneficial extrapolation behavior for generalization w.r.t. unseen input locations, which allows us to make predictions about new situations that are similar to, but outside the robot's experience, e.g., a faster movement or a bigger opening angle. This is particularly beneficial when the robot interactively learns about a new door.

A GP is defined by the mean function $\mu(\boldsymbol{y})$ and the covariance function $k(\boldsymbol{y}_i, \boldsymbol{y}_j)$. To define the GP for the deceleration profile of the door, we use the mean of all observed decelerations from the training dataset to compute a constant mean, i.e., $\mu = \frac{1}{N}\sum_{i=1}^{N} \alpha_i$. We subtract the mean from the observed values to obtain the vector of training target values $\boldsymbol{\alpha} = [\dots(\alpha_i - \mu)\dots]^T$. We use the squared exponential covariance function to compute the elements of the training data's covariance matrix $K(\mathcal{Y}, \mathcal{Y})$ from the training locations $\mathcal{Y} = \{\dots, \boldsymbol{y}_i, \dots\}$ as

$$K_{ij} = k_{SE}(\boldsymbol{y}_i, \boldsymbol{y}_j), \tag{5.17}$$

where

$$k_{SE}(\boldsymbol{y}_i, \boldsymbol{y}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{y}_j)^T \begin{bmatrix} l_\theta^2 & 0 \\ 0 & l_\omega^2 \end{bmatrix}^{-1} (\boldsymbol{y}_i - \boldsymbol{y}_j)\right). \tag{5.18}$$

Since the deceleration can vary quickly over a few degrees, we set the length scale l_θ to 5°. The velocity dependent variations were found to be roughly linear, with varying slope over θ. We thus enforced approximate linearity with $l_\omega = 1$, which is longer than the range of encountered velocities. To predict the acceleration given an angle and a velocity $\boldsymbol{y}_*$ we infer the mean $\bar{\alpha}_*$ of the conditional distribution $p(\alpha_* \mid \boldsymbol{y})$ as

$$\bar{\alpha}_* = k(\boldsymbol{y}_*, \mathcal{Y})(K(\mathcal{Y}, \mathcal{Y}) + \sigma_n^2 I)^{-1}\boldsymbol{\alpha}, \tag{5.19}$$

where $k(\boldsymbol{y}_*, \mathcal{Y}) \in \mathbb{R}^N$ is a row vector of the covariances of the new input $\boldsymbol{y}_*$ with the training dataset inputs $\mathcal{Y}$. The vector $\tilde{\boldsymbol{\alpha}} = (K(\mathcal{Y}, \mathcal{Y}) + \sigma_n^2 I)^{-1}\boldsymbol{\alpha}$ does not change during the opening motion and is therefore precomputed, reducing the online computations to N-fold evaluation of $k(\boldsymbol{y}_*, \mathcal{Y})$ and the dot product with $\tilde{\boldsymbol{\alpha}}$. To maintain real-time performance for big training datasets, we downsampled the trajectory data in accordance with the length scale of the locally weighted regression function (in Equation 2.47), which we used to estimate the deceleration from the observed trajectory.

The learned GP is used to numerically integrate the estimates of the deceleration analogous to Equation 5.16 with added dependency on the velocity. Figure 5.6 shows two profiles learned from observations.

As for the learning approach for a single trajectory, the transfer to the case of sliding doors is easily achieved by exchanging the measurements and predictions of angle, angular velocity and acceleration by position, linear velocity and acceleration respectively.

5.3 Interactive Learning of the Dynamics from Tactile Perception

In the previous sections we have presumed a human teacher that initially demonstrates the door motion to the robot. In this section, we present a method that lets the robot interactively bootstrap the dynamics model for unknown doors using tactile perception. The robot can estimate the friction $\boldsymbol{\tau}_f$ and the door's moment of inertia I using force or torque sensing during the contact phase of the first opening action. These estimates allow it to determine when to release the door using Equation 5.2. The friction in the hinge can be estimated by moving the door at constant angular velocity. For a constant velocity motion of the door, the torque applied to it by the end effector $\boldsymbol{\tau}_{ee}$ is equal to the friction torque $\boldsymbol{\tau}_f$. We can use the linear force $\boldsymbol{F}_{ee}$ at the end effector to compute the applied torque

$$\boldsymbol{\tau}_f = \boldsymbol{\tau}_{ee} = \boldsymbol{F}_{ee} \times \boldsymbol{r}, \tag{5.20}$$

where the vector $\boldsymbol{r}$ is perpendicular to the axis of rotation and connects said axis with the contact point of the door and the end effector of the robot. Since $\boldsymbol{r}$ is computed from the relative position of the hinge and the end effector, we require an estimate of the hinge axis. This can be estimated online by the method described in Section 5.2.2. In our experiments, the estimate obtained before releasing the door was always within a few centimeters of the final estimate. Since our sensor data is noisy, we average over the measurements from an interval of about 5°.

To determine the moment of inertia of the door, we need to estimate $\boldsymbol{\omega}$, the angular velocity, of the door. Analogous to the torque, we can compute the estimate from the linear velocity $\boldsymbol{v}_{ee}$ of the end effector,

$$\boldsymbol{\omega} = \boldsymbol{v}_{ee} \times \boldsymbol{r}. \tag{5.21}$$

Given the velocity $\boldsymbol{\omega}$ and the friction $\boldsymbol{\tau}_f$ we can compute the moment of inertia I by

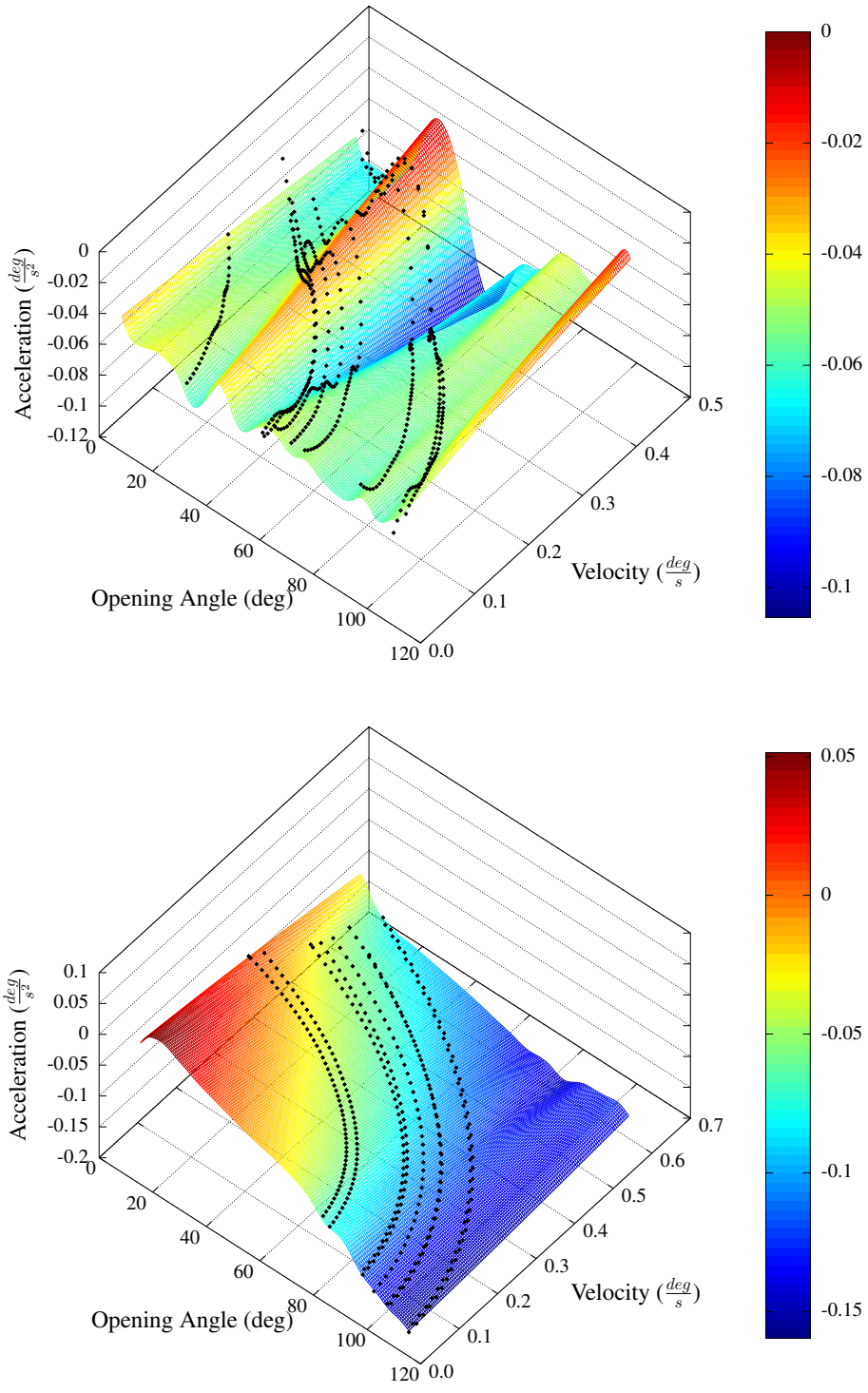


Figure 5.6: The resulting acceleration profile for two different doors, as modeled by the learned Gaussian process reflects the variations in deceleration over the position and the angular velocity.

accelerating the door. When accelerating the door, we can compute I from the relation

$$\alpha = \frac{\boldsymbol{\tau}_{ee} - \boldsymbol{\tau}_f}{I}. \tag{5.22}$$

To be robust to sensor noise, we integrate the measurements over time. Without further information, we assume the friction to be constant over the trajectory of the door and independent of the velocity.

$$\int_0^T \alpha(t)\,\mathrm{d}t = \int_0^T \frac{\tau_{ee}(t) - \tau_f}{I}\,\mathrm{d}t \tag{5.23}$$

$$\omega(T) - \omega(0) = \frac{1}{I}\int_0^T \tau_{ee}(t) - \tau_f\,\mathrm{d}t \tag{5.24}$$

$$I = \frac{\int_0^T \tau_{ee}(t)\,\mathrm{d}t - \tau_f T}{\omega(T) - \omega(0)}. \tag{5.25}$$

Given force and position measurements at the end effector at times $\{t_0, \ldots, t_i, \ldots, t_T\}$, we can compute the corresponding effective torques $\tau_i = |\boldsymbol{F}_i \times \boldsymbol{r}_i - \boldsymbol{\tau}_f|$. The door's moment of inertia can then be estimated by

$$I = \frac{\sum_{i=1}^{T} \tau_i (t_i - t_{i-1})}{\omega(T) - \omega(0)}. \tag{5.26}$$

Inserting the estimates for I and $\boldsymbol{\tau}_f$ into Equation 5.2, we can predict the final opening angle during the manipulation. However, the accuracy of this method depends greatly on how well the assumption of constant friction is met.

5.4 Experimental Evaluation

To evaluate the presented approach, we investigate the ability of the robot to learn and apply the model of the door dynamics to swing closed doors to a desired opening angle.

5.4.1 Experimental Setup

Robot

We use the DLR Light Weight Robot (LWR), a seven degree of freedom manipulator, with a passive end effector. It is mounted on a KUKA omniRob mobile base. We use a Hokuyo UTM-30LX laser range scanner to observe the door at 40 Hz. The setup is shown in Figure 5.7. The LWR is a powerful 7-DOF manipulator with a spherical wrist. Because the links are designed to be collinear when the manipulator is stretched out, the motion range of the three "elbow" joints needs to be limited to a minimum of $\pm 60^\circ$

Figure 5.7: The DLR Light Weight Robot with custom end effector used in our experiments, opening the metal door.

to avoid self-collision. This results in a limitation of the dexterous workspace near the robot, which makes motion planning difficult. For our approach this is not problematic, as we do not require complex motions and the LWR is powerful enough to accelerate even heavy doors with high friction within its workspace. We let the robot push the door using a linear position-controlled motion, such that the door achieves a velocity sufficient to reach the goal state. In the experiments we position the robot in front of the door and assume its relative position to be roughly known.

Doors

We demonstrate our approach on substantially different door types. The first door is a metal door attached to a concrete wall of the building (see Figure 5.7). The door is comparably heavy but smooth-running. The second door is made of a veneer on a wooden framework and is attached to a freestanding wooden frame (see Figure 5.1). It weighs only 27.8 kg. The friction is very low in the beginning, but the hinges are slightly misaligned, which increases the friction substantially towards 90°. To increase the variety, we conducted a further set of experiments with a brush seal attached to the wooden door

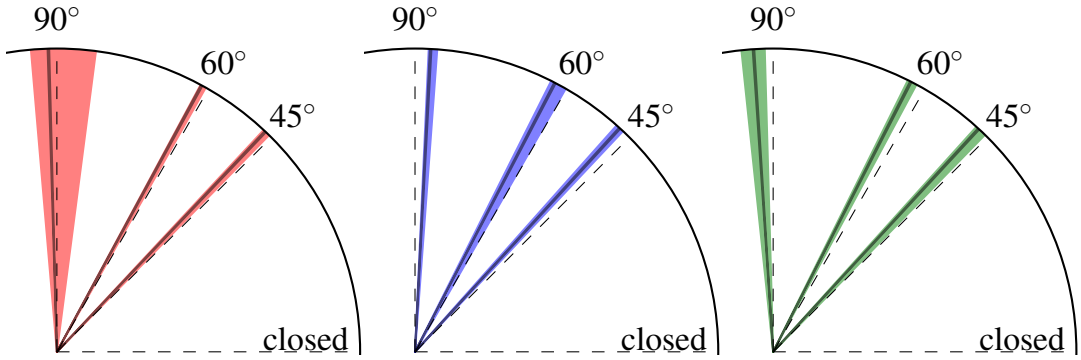


Figure 5.8: Visualization of the stopping angles of the doors in our experiments with three doors. For each door the robot has been commanded to open five times to 45, 60 and 90 degrees. The colored circle segments show the range from minimum to maximum, the solid black lines designate the mean, the dashed lines the respective commands. The dynamics model was learned from three observations for each door.

to increase the friction and an additional weight of 1.6 kg firmly attached to the handle. We will refer to these two configurations of the wooden door as "A" and "B".

5.4.2 Learning from Human Demonstration

We first evaluate the accuracy and precision of the opening task using models learned from observing a human demonstrator opening the doors. We recorded three demonstration of pushes with varying stopping angle for each door. We segmented the recorded data, as only the part from release to stop follows Equation 5.4. When the robot learns by experimentation, the release time is known. In case of human demonstrations, we determine the release time by using a wireless mouse to push the door. The demonstrator holds the mouse, such that the button makes the contact with the door during pushing. The release time (when the free motion of the door starts) is then obtained by the button release event. The stopping time is determined by searching for the maximum angle of the trajectory within some seconds of the release time. With this procedure, demonstrations are captured within seconds without requiring custom hardware. From the segmented data, the robot estimates the deceleration using locally weighted regression and creates the deceleration profile using the Gaussian process. The robot was then commanded to open each door to 45°, 60° and 90°. To evaluate the performance, each opening was repeated five times, resulting in a total of 45 executions. The model has not been updated from the robot's own actions. The results are given in Table 5.1 and visualized in Figure 5.8. This way of door opening is quick in execution. In the experiments, the time the robot was in contact with the door to accelerate it ranges from 0.6 s to 2.0 s with an average of only 1.0 s. The release angle was between 6 and 16 degrees.

Command	90.0°	60.0°	45.0°
Metal Door	91.5° ± 4.7°	61.3° ± 0.4°	46.4° ± 0.6°
Wooden Door A	86.9° ± 0.6°	62.4° ± 1.2°	47.4° ± 0.6°
Wooden Door B	93.6° ± 1.6°	62.8° ± 0.7°	47.2° ± 0.8°
Summary	90.6° ± 4.0°	62.2° ± 1.0°	47.0° ± 0.8°

Table 5.1: Average stopping angle and the standard deviation of 45 door openings, five for each door and each of three target states.

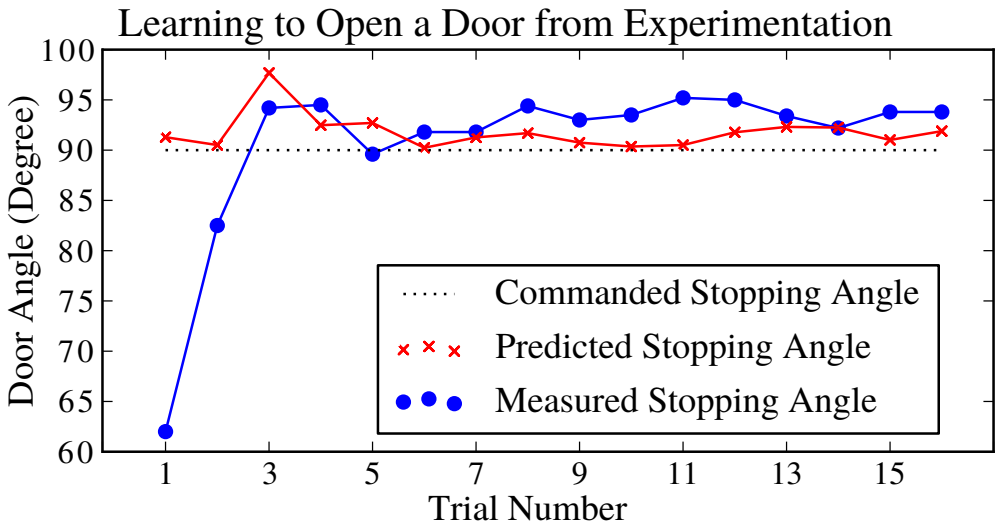


Figure 5.9: Unsupervised (purely interactive) learning of the dynamics of a door. Initially the robot has no knowledge of the door, therefore it estimates the friction and moment of inertia from tactile perception, as described in Section 5.3. From the second trial on the robot uses the depth sensor observations from previous trials.

5.4.3 Interactive Experimentation

While demonstrations with the method described in Section 5.2.3 are quickly done, the robot should be able to learn from its own actions. We evaluated the opening performance in a task sequence, where the robot generates and updates the model from its own experimentation. For the first opening, we apply the procedure described in Section 5.3, i.e., estimating the friction and moment of inertia from tactile perception. For this we use the torque sensors in the joints of the manipulator. To make the task challenging, we chose the door with the increasing friction, such that the initial estimate is guaranteed to fail. The robot therefore falls almost 30° short of opening the door to 90°. However, after only two observations, the result is within 5° of the desired opening angle. Figure 5.9 shows the whole sequence of 16 trials. The robot releases the door as soon as it predicts the target state to be reached. The prediction system may notice this too late, because of the time required to get the end-effector pose updates from the robot's operating system. In

this case the robot is aware that the door is overshooting and the estimate of how much is given by the red crosses in the figure. The experiment shows, that the robot is capable to learn a model of the door through unsupervised experimentation and that the predictions of this model are as accurate as for a model learned from observing a teacher.

To validate the tactile estimation approach independently of the approach based on depth-perception, we also verified the estimated moment of inertia. We do not have ground truth for the moment of inertia of the door, therefore we approximate it using the mass of the door and assuming a uniform mass distribution. While the latter assumption is violated by the mass of the door handle, the estimates for I obtained in our experiments approximately match the value computed this way.

5.5 Related Work

Perception and manipulation of doors has been intensively researched for over a decade. The following discussion of related work is grouped by the main subtasks involved.

5.5.1 Perception of Doors

Perception of Doors and Handles

The problem of door identification has been tackled by Anguelov et al. [4] who present an interesting approach based on laser range scans and a panoramic camera. They identify doors that have been observed in different opening angles by the laser range scanner. The identified doors allow them to learn how to distinguish walls and doors by color, such that similar doors can be identified in the camera data. While limited in determining the exact geometry of the doors, their approach provides valuable annotations of the doors in a map.

Limketkai et al. [81] propose a system to identify doors in 2D occupancy grid maps by learning common properties of the doors in the specific environment, such as the width and the indent from the wall. While based on strong assumptions, the method has the advantage to not rely on observing doors in different states. Rusu et al. [108] propose a system for identifying doors and extracting information about the geometry. They detect doors in 3D point clouds from a tilting laser range scanner by searching for offset planes that follow the standards for wheelchair accessible doors.

Assuming the door location to be roughly known, several methods have been proposed to detect the handle using visual features [73, 74] or a laser range scanner [101]. Others methods additionally obtain the exact location and dimensions of the door frame, e.g. using active vision with a stereo camera [3] or a tilting laser range scanner [108].

For quasi-static manipulation, a detailed investigation of *friction profiles* of articulated household objects has been manually conducted by Jain et al. [64]. They propose to apply

this information to, e.g., detect when a mechanism is blocked. In our work, we found that the friction forces in moving doors substantially vary with the speed of the door, and therefore proposed to learn a *deceleration profile* that considers the velocity of the door. We present a method for learning the deceleration profile only from observations with a depth sensor, or from tactile sensing.

Estimating the Kinematic Structure

There has been intensive research on accurately estimating the kinematic model of articulated objects, either during manipulation from the end effector [67, 98] trajectory, or by tracking the object using visual or depth cameras [68, 114]. Sturm et al. [115] propose an approach that is able to distinguish various kinematic structures, learning the most appropriate model for the mechanism at hand. However, these methods all require tracking of fixed points on the mechanism, e.g., the firmly grasped handle via forward kinematics, visual features or markers. We do not require a grasp, hence we cannot use the trajectory of the end effector. We therefore proposed a novel approach with the ability to learn the kinematics using a depth sensor such as a laser range scanner or an RGB-D camera in Section 5.2.2.

5.5.2 Manipulation of Doors

Unlatching the Door

Many approaches to door opening focus on unlatching the door and consider the door to be open after unlatching. Reliable results have been experimentally demonstrated, e.g., by Chitta et al. [16] and Jain and Kemp [63]. Complementary, we do not consider unlatching of the door in this work but assume the door to be unlatched and focus on quickly and accurately moving the door to a desired state.

Control Approaches

Several control algorithms that implement compliant control have been proposed. Compliant manipulation aims at applying force only along the constrained trajectory of the articulated object, while keeping lateral forces to a minimum. This may also be referred to as hybrid control, when combining position or velocity control to move along the trajectory and force control orthogonal to the trajectory to minimize lateral forces.

In the work of Meeussen et al. [87], the robot maintains a force in the forward direction of the end effector, which is assumed to follow the trajectory of the object due to the firm grasp and the compliant control. Some approaches learn the kinematic model during the manipulation [67, 98, 114] and use it for hybrid control. Reachability issues are usually addressed by moving the base, to increase the workspace and keep the manipulator

away from singular configurations [87, 98]. In contrast to motion planning approaches, control algorithms can not foresee whether the robot will succeed or end up in a dead-end configuration. Therefore the overall approach needs to ensure that the goal of the task is reachable.

The ability to predict the dynamics of the door, as provided by our approach, would be particularly beneficial in combination with control approaches that perform a quick motion. The robot could then release the grasp earlier. Either as soon as the kinetic energy suffices to reach the target state as in our experiments, or when the required momentum is below a safety threshold. This reduces reachability problems, as the contact phase is reduced.

Motion Planning Approaches

In contrast to control approaches, motion planning approaches plan the manipulation in advance by searching for a sequence of actions that will lead from an initial state to a desired goal state [16], [13]. The search for admissible intermediate states intrinsically allows to avoid issues with reachability, singular configurations and collisions. If a plan is found, it is therefore guaranteed that the robot will not be stuck in a local dead end. Unfortunately the planning space is high dimensional and highly constrained. Finding a valid plan in acceptable time is therefore a topic of intensive research [75, 120, 129]. In most planning approaches a rigid grasp of the object is assumed. To relax the constraints on the manipulator arising from the grasp, Diankov et al. [24] propose to use *caging grasps*, that allow small variations of the end effector position during the task, easing the search for transitional states. Nevertheless, motion planning in the context of door manipulation remains algorithmically complex and computationally highly demanding.

5.6 Conclusion

In this chapter we investigated the perception and modeling of doors with a depth sensor, to enable a robot to navigate indoor environments. We advance the state-of-the-art in learning of the kinematics of the door by a novel method based on depth data only. We further propose an approach to accurately estimate the deceleration of the door and learn a model of its dynamics. We demonstrate that the approach is applicable to learn from interactive experimentation or from observation of demonstrations by a human.

Using a non-parametric regression, we modeled the deceleration with respect to angle and velocity of the door. In experiments with a real robot, we demonstrated that we can exploit this model of the dynamics in a door opening task to implement a manipulation strategy that reduces the requirements on the dexterous workspace of the robot. Because the model allows to predict the trajectory of the door throughout the manipulation, the

requirement for a fixed grasp is removed and a point contact is sufficient for accurately moving the door to a desired angle. We showcased this by accelerating the door and releasing it as soon as possible, while precisely achieving a desired state.

The presented approaches open interesting opportunities for future research. Integration of the models into a planning approach would make planning and execution more flexible and would lead to a tighter integration with overall mission planning. Further we have chosen a fixed robot position and – except for the force applied – a fixed manipulator motion. Both should be chosen with respect to the position of the robot and the kinematic and dynamic properties of its manipulator; to ensure success, where possible, or to limit the maximum speed of the door.

Chapter 6

Conclusions

Truly autonomous robots must be capable to model their environment from their onboard sensors. The learned models are task-specific and essential for planning their actions. In this thesis, we presented our contributions to environment modelling, focusing on the perception of robots operating in indoor environments. We identified the creation of volumetric maps and modelling of doors as fundamental skills for indoor navigation of such robots. Therefore, we presented a system for simultaneous localization and mapping (SLAM) in 3D with RGB-D cameras, based on sparse visual features. We developed several novel techniques for SLAM frontend and backend that increase the accuracy and improve the robustness with respect to challenging input data. A thorough evaluation shows the benefits of our contributions on a public benchmark dataset, which covers a wide range of real-world scenarios. Further, we could establish that our system outperforms other state-of-the-art systems.

Compared to cameras with wide-angle lenses or with catadioptric extensions such as parabolic mirrors, RGB-D cameras have a limited field of view. This property, and the limited range of the depth perception may lead to a lack of usable data in SLAM applications. To extend the available field of view, our SLAM system can fuse data from multiple RGB-D sensors. We devised a novel approach to the extrinsic calibration of multiple RGB-D cameras, which is integrated into the SLAM backend and therefore allows to calibrate the sensors while mapping. Furthermore , we presented a novel extension for RGB-D cameras, which uses planar mirrors to split the image, creating two opposite viewpoints from a single sensor. In experiments on a real robot, we showed that the benefits of this catadioptric extension in a SLAM application are similar to those obtained with a second sensor.

For indoor navigation, an essential task for autonomous robots is the manipulation of doors, as their effective range of operation otherwise reduces to a single room. Current state-of-the-art approaches require complex manipulators with a large dexterous workspace, which is not feasible for consumer household robots. However, we showed that these requirements can be greatly lowered if the robot uses a manipulation strategy that considers the dynamics of the door. We consequently presented approaches to learn

a model of the kinematics and dynamics of a door. We developed a method that allows the robot to learn from demonstrations using a 3D sensor to perceive the door in motion. Further, we showed that the robot can bootstrap his model interactively using tactile sensing. In experiments with a real robot, we demonstrated that our approach enables the robot to open a door using only a simple forward motion with a point contact, effectively requiring contact to the door only for a range of a few centimeters. Despite these relaxed requirements, the robot is able to bring the door precisely to a desired state.

The approaches presented in this thesis form a novel foundation for indoor navigation, which allows to realize robotic applications with reduced cost. Thus, this work will be particularly beneficial for the progress of consumer robots.

Appendix A

Detailed Benchmarking Results

Here, we list the performance of our RGB-D SLAM implementation *RGBDSLAM v2*, as available on github[1]. We state various statistics on the error in the estimated trajectory, applying all contributions proposed in Chapter 3. Scientific publications typically select specific statistics to emphasize certain properties. In contrast, here we attempt to provide a comprehensive evaluation to serve as a reference which accompanies the error metrics, ground truth and datasets of the benchmark.

A.1 Dataset with Public Ground Truth

In this section, we list the results of all datasets for which the ground truth is publicly available, except for the calibration sequences which mostly consist of recordings of checkerboards. The development of our approach was mostly guided by experiments on the sequences of the "fr1" and the "pioneer SLAM" datasets. Nevertheless, as the table below shows, our approach is also highly accurate on most of the remaining sequences.

Sequence Name	rmse (m)	median (m)	mean (m)	std (m)	max (m)	FPS (Hz)
fr1 360	0.028	0.146	0.060	0.056	0.066	23.3
fr1 desk	0.014	0.083	0.021	0.019	0.025	26.2
fr1 desk2	0.021	0.105	0.029	0.024	0.036	34.6
fr1 floor	0.015	0.109	0.027	0.024	0.031	26.6
fr1 plant	0.027	0.275	0.051	0.048	0.057	20.6
fr1 room	0.035	0.201	0.086	0.077	0.092	10.8
fr1 rpy	0.011	0.069	0.021	0.019	0.024	18.2
fr1 teddy	0.022	0.417	0.050	0.048	0.055	10.3
fr1 xyz	0.008	0.062	0.013	0.011	0.015	25.8
fr2 360 hemisphere	0.151	1.633	0.256	0.243	0.297	36.5
fr2 360 kidnap	0.215	1.245	0.440	0.411	0.489	17.0
fr2 coke	0.048	0.250	0.064	0.050	0.080	17.1
fr2 desk	0.013	0.067	0.029	0.027	0.032	26.6

Continued on the next page

[1] http://felixendres.github.io/rgbdslam_v2

Sequence Name	rmse (m)	median (m)	mean (m)	std (m)	max (m)	FPS (Hz)
fr2 desk with person	0.007	0.079	0.017	0.017	0.019	10.5
fr2 dishes	0.041	0.172	0.052	0.038	0.066	33.3
fr2 flowerbouquet	0.022	0.098	0.031	0.023	0.038	29.6
fr2 flowerbouquet brownbackground	0.382	1.677	0.664	0.448	0.766	30.2
fr2 large no loop	0.480	2.330	1.317	1.199	1.402	18.4
fr2 large with loop	0.148	0.892	0.350	0.294	0.380	27.6
fr2 metallic sphere	0.138	0.811	0.214	0.180	0.255	28.3
fr2 metallic sphere2	0.032	0.174	0.054	0.048	0.063	28.5
fr2 pioneer 360	0.073	0.366	0.162	0.135	0.178	16.8
fr2 pioneer slam	0.068	0.619	0.140	0.129	0.155	25.8
fr2 pioneer slam2	0.069	0.328	0.188	0.216	0.201	18.5
fr2 pioneer slam3	0.121	0.665	0.262	0.227	0.289	26.9
fr2 rpy	0.007	0.038	0.014	0.013	0.016	27.2
fr2 xyz	0.008	0.043	0.013	0.011	0.015	29.8
fr3 cabinet	0.009	0.074	0.032	0.031	0.033	28.5
fr3 large cabinet	0.027	0.138	0.052	0.053	0.058	32.8
fr3 long office household	0.008	0.055	0.022	0.021	0.024	9.4
fr3 nostructure notexture far	0.036	0.155	0.070	0.071	0.079	33.7
fr3 nostructure notexture near withloop	0.074	0.289	0.128	0.112	0.148	27.8
fr3 nostructure texture far	0.028	0.268	0.038	0.031	0.047	32.0
fr3 nostructure texture near withloop	0.008	0.088	0.018	0.017	0.020	33.5
fr3 sitting halfsphere	0.010	0.086	0.018	0.016	0.021	25.7
fr3 sitting rpy	0.020	0.191	0.023	0.017	0.031	30.8
fr3 sitting static	0.003	0.023	0.005	0.005	0.006	34.4
fr3 sitting xyz	0.012	0.068	0.025	0.022	0.028	27.2
fr3 structure notexture far	0.011	0.064	0.025	0.024	0.027	30.3
fr3 structure notexture near	0.006	0.045	0.025	0.025	0.026	33.1
fr3 structure texture far	0.015	0.075	0.031	0.026	0.035	25.3
fr3 structure texture near	0.011	0.065	0.031	0.030	0.033	20.8
fr3 teddy	0.013	0.258	0.027	0.025	0.030	10.6
fr3 walking halfsphere	0.021	0.147	0.035	0.031	0.041	17.1
fr3 walking rpy	0.142	1.076	0.101	0.062	0.174	30.2
fr3 walking static	0.005	0.045	0.010	0.009	0.011	21.7
fr3 walking xyz	0.029	0.139	0.038	0.027	0.048	26.7

A.2 Benchmark Dataset Sequences

This section lists the sequences by category and states properties that are helpful in the interpretation of the results in Appendix A.1. The category name reflects the purpose of the sequences, e.g., evaluation of the performance for "Handheld SLAM". The duration, trajectory length and the covered area are obviously correlated to the error. The stated velocities influence the overlap between frames and – more severely – the motion blur. For further details about the sequences, e.g. a short description of each sequence, we refer the reader to the web page of the benchmark[2].

Sequence Name	**Duration (s)**	**Trajectory Length (m)**	**Average transl. velocity (m/s)**	**Average angular velocity (deg/s)**	**Trajectory dimensions (m × m × m)**
Category: Testing and Debugging					
freiburg1 xyz	30.09	7.112	0.244	8.920	0.46 × 0.70 × 0.44
freiburg1 rpy	27.67	1.664	0.062	50.147	0.15 × 0.21 × 0.21
freiburg2 xyz	122.74	7.029	0.058	1.716	1.30 × 0.96 × 0.72
freiburg2 rpy	109.97	1.506	0.014	5.774	0.21 × 0.22 × 0.11
Category: Handheld SLAM					
freiburg1 360	28.69	5.818	0.210	41.600	0.54 × 0.46 × 0.47
freiburg1 floor	49.87	12.569	0.258	15.071	2.30 × 1.31 × 0.58
freiburg1 desk	23.40	9.263	0.413	23.327	2.42 × 1.34 × 0.66
freiburg1 desk2	24.86	10.161	0.426	29.308	2.44 × 1.47 × 0.52
freiburg1 room	48.90	15.989	0.334	29.882	2.54 × 2.21 × 0.51
freiburg2 360 hemisphere	91.48	14.773	0.163	20.569	3.06 × 3.48 × 0.58
freiburg2 360 kidnap	48.04	14.286	0.304	13.425	4.26 × 3.44 × 0.12
freiburg2 desk	99.36	18.880	0.193	6.338	3.90 × 4.13 × 0.57
freiburg2 large no loop	112.37	26.086	0.243	15.090	3.63 × 5.07 × 0.20
freiburg2 large with loop	173.19	39.111	0.231	17.211	3.39 × 4.26 × 0.53
freiburg3 long office household	87.09	21.455	0.249	10.188	5.12 × 4.89 × 0.54
Category: Robot SLAM					
freiburg2 pioneer 360	72.75	16.118	0.225	12.053	4.24 × 4.38 × 0.06
freiburg2 pioneer slam	155.72	40.380	0.261	13.379	5.50 × 5.94 × 0.07
freiburg2 pioneer slam2	115.63	21.735	0.190	12.209	4.98 × 5.34 × 0.07
freiburg2 pioneer slam3	111.91	18.135	0.164	12.339	5.29 × 5.25 × 0.07

Continued on the next page

[2]http://vision.in.tum.de/data/datasets/rgbd-dataset/download

Sequence Name	Duration (s)	Trajectory Length (m)	Average transl. velocity (m/s)	Average angular velocity (deg/s)	Trajectory dimensions (m × m × m)
Category: Structure vs. Texture					
freiburg3 nostructure notexture far	15.79	2.897	0.196	2.712	0.24 × 2.95 × 0.05
freiburg3 nostructure notexture near withloop	37.74	11.739	0.319	11.241	2.98 × 3.65 × 0.34
freiburg3 nostructure texture far	15.53	4.343	0.299	2.890	0.12 × 4.33 × 0.09
freiburg3 nostructure texture near withloop	56.48	13.456	0.242	7.430	3.66 × 4.79 × 0.32
freiburg3 structure notexture far	27.28	4.353	0.166	4.000	0.87 × 3.89 × 0.06
freiburg3 structure notexture near	36.44	3.872	0.109	6.247	0.53 × 3.39 × 0.18
freiburg3 structure texture far	31.55	5.884	0.193	4.323	1.90 × 4.56 × 0.08
freiburg3 structure texture near	36.91	5.050	0.141	7.677	1.01 × 4.01 × 0.17
Category: Dynamic Objects					
freiburg2 desk with person	142.08	17.044	0.121	5.340	2.30 × 3.93 × 0.51
freiburg3 sitting static	23.63	0.259	0.011	1.699	0.12 × 0.08 × 0.05
freiburg3 sitting xyz	42.50	5.496	0.132	3.562	0.99 × 0.93 × 1.02
freiburg3 sitting halfsphere	37.15	6.503	0.180	19.094	1.46 × 0.88 × 1.19
freiburg3 sitting rpy	27.48	1.110	0.042	23.841	0.18 × 0.12 × 0.20
freiburg3 walking static	24.83	0.282	0.012	1.388	0.10 × 0.05 × 0.05
freiburg3 walking xyz	28.83	5.791	0.208	5.490	0.94 × 0.82 × 1.15
freiburg3 walking halfsphere	35.81	7.686	0.221	18.267	1.83 × 0.83 × 1.31
freiburg3 walking rpy	30.61	2.698	0.091	20.903	0.79 × 0.47 × 0.14
Category: 3D Object Reconstruction					
freiburg1 plant	41.53	14.795	0.365	27.891	1.71 × 1.70 × 1.07
freiburg1 teddy	50.82	15.709	0.315	21.320	2.42 × 2.24 × 1.43
freiburg2 coke	84.55	11.681	0.140	9.432	1.60 × 2.02 × 0.73
freiburg2 dishes	100.55	15.009	0.151	9.666	1.79 × 2.18 × 0.67
freiburg2 flowerbouquet	99.40	10.758	0.109	8.464	1.64 × 1.65 × 0.35
freiburg2 flowerbouquet brownbackground	76.89	11.924	0.157	10.598	1.76 × 1.99 × 0.70
freiburg2 metallic sphere	75.60	11.040	0.148	10.422	1.82 × 2.23 × 0.71
freiburg2 metallic sphere2	62.33	11.813	0.193	12.946	1.74 × 1.97 × 0.77
freiburg3 cabinet	38.58	8.111	0.216	10.248	2.72 × 2.50 × 0.44
freiburg3 large cabinet	33.98	11.954	0.362	8.747	3.70 × 5.44 × 0.28
freiburg3 teddy	80.79	19.807	0.248	20.410	2.06 × 2.00 × 1.05

List of Figures

Bibliography

[1] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, May 2013.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Commun. ACM*, 54(10):105–112, Oct. 2011.

[3] A. Andreopoulos and J. K. Tsotsos. Active vision for door localization and door opening using playbot: A computer controlled wheelchair for people with mobility impairments. In *Computer and Robot Vision, 2008. CRV '08. Canadian Conference on*, pages 3–10, May 2008.

[4] D. Anguelov, D. Koller, E. Parker, and S. Thrun. Detecting and modeling doors with mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3777–3784, 2004.

[5] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, 2008. ISSN 1077-3142.

[8] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

[9] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] J. Brookshire and S. Teller. Automatic calibration of multiple coplanar sensors. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[12] J. Brookshire and S. Teller. Extrinsic calibration from per-sensor egomotion. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[13] F. Burget, A. Hornung, and M. Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, May 2013.

[14] G. Carrera, A. Angeli, and A. J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[15] A. Censi. An ICP variant using a point-to-line metric. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, May 2008.

[16] S. Chitta, B. Cohen, and M. Likhachev. Planning for autonomous door opening with a mobile manipulator. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1799–1806, 2010.

[17] S. Choi, T. Kim, and W. Yu. Performance evaluation of ransac family. In *Proceedings of the British Machine Vision Conference*, pages 81.1–81.12. BMVA Press, 2009.

[18] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Pattern Recognition*, pages 236–243. Springer, 2003.

[19] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

[20] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 2010.

[21] J. Cunha, E. Pedrosa, C. Cruz, A. J. Neves, and N. Lau. Using a depth camera for indoor robot localization and navigation. *DETI/IEETA-University of Aveiro, Portugal*, 2011.

[22] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[23] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[24] R. Diankov, S. S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning with caging grasps. In *Humanoids*, pages 285–292, 2008.

[25] I. Dryanovski, C. Jaramillo, and J. Xiao. Incremental registration of rgb-d images. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1685–1690, 2012.

[26] I. Dryanovski, R. Valenti, and J. Xiao. Fast visual odometry and mapping from rgb-d data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2305–2310, May 2013.

[27] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[28] F. Endres, J. Hess, N. Franklin, C. Plagemann, C. Stachniss, and W. Burgard. Estimating range information from monocular vision. In *Workshop Regression in Robotics - Approaches and Applications at Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

[29] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard. Scene analysis using latent Dirichlet allocation. In *Proc. of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

[30] F. Endres, J. Hess, N. Engelhard, J. Sturm, and W. Burgard. http://www.ros.org/wiki/openni/Contests/ROS3D/RGBD-6D-SLAM, 2011.

[31] F. Endres, J. Hess, and W. Burgard. Graph-based action models for human motion classification. In *ROBOTIK*, May 2012.

[32] F. Endres, J. Hess, N. Engelhard, J. Sturm, and W. Burgard. 6D visual SLAM for RGB-D sensors. *at - Automatisierungstechnik*, 60:270–278, May 2012.

[33] 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 International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, May 2012.

[34] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. RGBDSLAM. http://ros.org/wiki/rgbdslam, July 2013.

[35] F. Endres, J. Trinkle, and W. Burgard. Interactive perception for learning the dynamics of articulated objects. In *Proceedings of the ICRA 2013 Mobile Manipulation Workshop on Interactive Perception*, Karlsruhe, Germany, May 2013.

[36] F. Endres, J. Trinkle, and W. Burgard. Learning the dynamics of doors for robotic manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

[37] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. RGBDSLAM v2. https://github.com/felixendres/rgbdslam_v2, July 2014.

[38] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3D mapping with an RGB-D camera. *IEEE Trans. on Robotics*, 30(1):177–187, Feb 2014.

[39] F. Endres, C. Sprunk, R. Kuemmerle, and W. Burgard. A catadioptric extension for RGB-D cameras. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[40] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Sydney, Australia, Dec 2013.

[41] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3D visual SLAM with a hand-held camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, 2011.

[42] M. F. Fallon, H. Johannsson, M. Kaess, D. M. Rosen, E. Muggler, and J. J. Leonard. Mapping the MIT stata center: Large-scale integrated visual and RGB-D SLAM. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012. URL http://projects.csail.mit.edu/stata/.

[43] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[44] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image Vision Comput.*, 21(13-14): 1145–1153, 2003.

[45] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a cloudless day. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, volume 6314 of *Lecture Notes in Computer Science*, pages 368–381. Springer, 2010.

[46] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[47] J. Gluckman and S. Nayar. Rectified Catadioptric Stereo Sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):224–236, Feb 2002.

[48] Google. https://www.google.com/atap/projecttango/, 2014.

[49] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[50] 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)*, Anchorage, AK, USA, May 2010.

[51] S. Grzonka, G. Grisetti, and W. Burgard. A Fully Autonomous Indoor Quadrotor. *IEEE Trans. on Robotics*, 8(1):90–100, 2 2012.

[52] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust statistics : the approach based on influence functions*. John Wiley & Sons, Inc., 1986.

[53] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[54] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[55] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*, Delhi, India, 2010.

[56] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. Journal of Robotics Research*, 31(5): 647–663, April 2012.

[57] D. Herrera, J. Kannala, and J. Heikkilä. Joint depth and color camera calibration with distortion correction. *IEEE Journal on Pattern Analysis and Machine Intelligence (PAMI)*, 34(10), Oct 2012.

[58] 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.

[59] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake. A robust RGB-D SLAM algorithm. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1714–1719, 2012.

[60] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[61] P. J. Huber. Robust statistics. 1981.

[62] S. Ito, F. Endres, M. Kuderer, G. D. Tipaldi, C. Stachniss, and W. Burgard. W-RGB-D: Floor-plan-based indoor global localization using a depth camera and WiFi. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[63] A. Jain and C. C. Kemp. Behavior-Based Door Opening with Equilibrium Point Control. In *RSS Workshop: Mobile Manipulation in Human Environments*, 2009.

[64] A. Jain, H. Nguyen, M. Rath, J. Okerman, and C. C. Kemp. The Complex Structure of Simple Devices: A Survey of Trajectories and Forces that Open Doors and Drawers. In *Proc. of the IEEE RAS/EMBS Intl Conf. on Biomedical Robotics and Biomechatronics (BIOROB)*, pages 729–736, 2010.

[65] H. Jin, P. Favaro, and S. Soatto. Real-time 3-d motion and structure of point-features: A front-end for vision-based control and interaction. In *Proc. of the IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[66] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, Dec. 2008.

[67] Y. Karayiannidis, C. Smith, F. Vina, P. Ögren, and D. Kragic. 'open sesame!' - adaptive force/velocity control for opening unknown doors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4040–4047, 2012.

[68] D. Katz and O. Brock. Extracting planar kinematic models using interactive perception. In *Unifying Perspectives in Computational and Robot Vision*, volume 8, pages 11–23. Springer US, 2008.

[69] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013.

[70] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[71] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[72] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE and ACM Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, 2007.

[73] E. Klingbeil, A. Saxena, and A. Ng. Learning to open new doors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2751–2757, 2010.

[74] D. Kragic, L. Petersson, and H. I. Christensen. Visually guided manipulation tasks. In *Robotics and Autonomous Systems, 40(2-3):193 – 203*, 2002.

[75] J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 995–1001, 2000.

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

[77] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

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

[79] J. Levinson and S. Thrun. Automatic calibration of cameras and lasers in arbitrary environments. In *International Symposium on Experimental Robotics*, 2012.

[80] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Proc. of Robotics: Science and Systems (RSS)*, 2013.

[81] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1471–1476, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[82] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60(2):91–110, 2004.

[83] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[84] 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.

[85] R. Maier, J. Sturm, and D. Cremers. Submap-based bundle adjustment for 3d reconstruction from rgb-d data. In *German Conference on Pattern Recognition (GCPR)*, Münster, Germany, September 2014.

[86] R. M. Martín and O. Brock. Deterioration of depth measurements due to interference of multiple rgb-d sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[87] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. P. Gerkey, and E. Berger. Autonomous door opening and plugging in with a personal robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 729–736, 2010.

[88] M. Meilland and A. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 3-8 November 2013. IEEE/RSJ.

[89] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis. 3d lidar–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The International Journal of Robotics Research*, 31(4):452–467, 2012.

[90] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[91] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.

[92] D. Nister. Preemptive RANSAC for live structure and motion estimation. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[93] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.

[94] S. Oßwald, A. Hornung, and M. Bennewitz. Improved proposals for highly accurate localization using range and vision data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012.

[95] P. R. Osteen, J. L. Owens, and C. C. Kessens. Online egomotion estimation of rgb-d sensors using spherical harmonics. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1679–1684, 2012.

[96] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. In *7th IFAC symposium on intelligent autonomous vehicles*, volume 7, 2010.

[97] K. B. Petersen and M. S. Pedersen. The Matrix Cookbook. http://www2.imm.dtu.dk/pubdb/p.php?3274, October 2008.

[98] L. Peterson, D. Austin, and D. Kragic. High-level control of a mobile manipulator for door opening. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2333–2338, 2000.

[99] C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin. A nonparametric learning approach to range sensing from omnidirectional vision. *Robotics and Autonomous Systems*, 58(6):762 – 772, 2010.

[100] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a depth camera: Parameter exploration for fast icp. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[101] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Ng. High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2816–2822, 2009.

[102] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[103] J. Roewekaemper, C. Sprunk, G. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard. On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[104] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, volume 1, pages 430–443, May 2006.

[105] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2292 – 2297, Detroit, MI, USA, 1999.

[106] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 13, 2011.

[107] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of the Intl. Conf. on 3-D Digital Imaging and Modeling*, Quebec, Canada, 2001.

[108] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz. Laser-based Perception for Door and Handle Identification. In *Proc. of the Intl. Conf. on Advanced Robotics (ICAR)*, 2009.

[109] D. Scaramuzza. *Omnidirectional Vision: from Calibration to Root Motion Estimation*. PhD thesis, Swiss Federal Institute of Technology Zurich (ETHZ), February 2008.

[110] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proceedings of Robotics: Science and Systems*, 2009.

[111] C. Sprunk, G. D. Tipaldi, A. Cherubini, and W. Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[112] F. Steinbruecker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[113] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, 2010.

[114] J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard. Operating articulated objects based on experience. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2739–2744, 2010.

[115] J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal on Artificial Intelligence Research (JAIR)*, 41:477–526, 2011.

[116] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. http://vision.in.tum.de/data/datasets/rgbd-dataset, October 2012.

[117] 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)*, Vilamoura, Portugal, 2012.

[118] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Proceedings of Robotics: Science and Systems*, 2013.

[119] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[120] J. Trinkle and R. Milgram. Complete path planning for closed kinematic chains with spherical joints. *International Journal of Robotics Research*, 21(9):773–789, 2002.

[121] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (13), 1991.

[122] P. Wegner. A technique for counting ones in a binary computer. *CACM*, 3(5):322–322, May 1960.

[123] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[124] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[125] T. Whelan, M. Kaess, J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[126] Wikipedia. Line-line intersection — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Line-line_intersection&oldid=541501982, 2013. [Online; accessed 4-March-2013].

[127] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). url-http://cs.unc.edu/ ccwu/siftgpu, 2007.

[128] Y. YAGI. Omnidirectional sensing and its applications. *IEICE Transactions on Information and Systems*, 82(3):568–579, 1999.

[129] Y. Yang and O. Brock. Elastic roadmaps - motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010.

[130] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, 2012.

[131] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1997.

[132] J. Zienkiewicz, R. Lukierski, and A. J. Davison. Dense, auto-calibrating visual odometry from a downward-looking camera. In *Proc. of the British Machine Vision Conference (BMVC)*, 2013.