Highly Accurate Mobile Robot Navigation

Christoph Sprunk

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Highly Accurate Mobile Robot Navigation

Christoph Sprunk

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. Cyrill Stachniss
	Rheinische Friedrich-Wilhelms-Universität Bonn
Tag der Disputation	27. November 2015

Zusammenfassung

Die herstellende Industrie erhöht durch die fortschreitende Automatisierung von Prozessen ihre Produktivität und reduziert gleichzeitig den Anteil körperlich beanspruchender Arbeit. Damit trägt Automatisierung einen wesentlichen Teil zu dem in industrialisierten Gesellschaften heute üblichen wirtschaftlichen Wohlstand und Lebensstandard bei. Die Automatisierung eines Produktionsprozesses erfordert heute üblicherweise eine aufwendige und dadurch kostenintensive Aufbau- und Inbetriebnahmephase, in der ein komplexes System bestehend zum Beispiel aus Förderbändern, Flurförderfahrzeugen, Roboterarmen und produktspezifischen Maschinen montiert und eingerichtet wird. Aus Gründen der Wirtschaftlichkeit wird industrielle Automatisierung daher nur bei der Produktion großer Mengen baugleicher Produkte eingesetzt. Infolge der Globalisierung nimmt jedoch die Geschwindigkeit von Marktveränderungen zu, was in der herstellenden Industrie zu kürzeren Zeitfenstern für die Anpassung ihrer Produktpalette und der zu produzierenden Stückzahlen führt. Zusammen mit dem Trend zur Individualisierung von Produkten entsteht hierdurch ein Bedarf für flexiblere, anpassbare Automatisierungstechniken, um die globale Wettbewerbsfähigkeit für kleine Losgrößen und sich verändernde Produkte zu erhalten. Hierfür sind kürzere Umrüstzeiten und geringere Kosten für den Auf- und Umbau von Automatisierungslösungen erforderlich. Für das Erreichen dieser Ziele ist autonome Navigation eine Schlüsseltechnologie.

In dieser Arbeit stellen wir Beiträge vor, die den Einsatz von Roboternavigation in industriellen Anwendungsszenarien zum Ziel haben. Unsere Verfahren stellen Materialtransport zwischen einzelnen Prozessschritten bereit und sind schnell und einfach in Betrieb zu nehmen. Da unsere Systeme lediglich auf Inkrementgeber an den Rädern der Fahrzeuge und auf Messdaten der vorgeschriebenen Sicherheitslaserscanner zurückgreifen, können wir auf zusätzliche unflexible Infrastruktur wie Reflektoren, Markierungen, oder Führungsdrähte und -Schienen verzichten. Wir stellen zunächst ein vollständiges Navigationssystem vor, das es omnidirektionalen Robotern ermöglicht, autonom, mit hoher Genauigkeit und unter Einhaltung sicherheitsrelevanter Rahmenbedingungen zu navigieren. Unser System berücksichtigt die Dynamik des Fahrzeugs bei der auf Optimierung basierenden Planung von glatten, krümmungskontinuierlichen Trajektorien. Aufgrund seiner Effizienz kann unser System die geplanten Trajektorien während der Fahrt regelmäßig aktualisieren, um auf Veränderungen in dynamischen Umgebungen zu reagieren. Wir präsentieren eine Methode, welche die Genauigkeit beim Anfahren von Zielpositionen in den Millimeterbereich erhöht. Hierzu vergleichen wir an der Zielposition aufgenommene, nicht diskretisierte

Referenzdaten der Sicherheitslaserscanner mit aktuellen Messdaten.

Für Anwendungsfälle, die erfordern, dass automatisierte Fahrzeuge fest vordefinierten Bahnen folgen, schlagen wir eine nutzerfreundliche, intuitive Methode zur Inbetriebnahme vor, die auf Demonstrationen des Benutzers basiert. Unser Navigationssystem lässt sich mit den so definierten virtuellen Spuren zu unterschiedlichen Autonomiegraden kombinieren. In Anwendungsszenarien, die ausschließlich vordefinierte Bahnen erfordern, reduzieren wir den Inbetriebnahmeaufwand weiter. Unser teach-and-repeat Framework fährt durch einmalige Demonstration eingelernte virtuelle Spuren mit einer Genauigkeit im Millimeterbereich ab, ohne dass es nötig ist, zuvor eine Karte der Umgebung zu erstellen.

Um zu einer realistischen Einschätzung von Navigationssystemen in Bezug auf ihre Leistungsfähigkeit, Robustheit und Zuverlässigkeit zu gelangen, stellen wir ein Testverfahren vor, das nicht nur einzelne Komponenten sondern vollständige Navigationssysteme durch Interaktion mit einer sorgfältig spezifizierten, sich verändernden Umgebung bewertet. Der Einsatz eines Referenzsystems, das die wiederholbaren Tests ebenfalls durchläuft, führt zur Vergleichbarkeit von Ergebnissen aus unterschiedlichen Instantiierungen unserer Testumgebung.

Wir haben mit allen vorgestellten Systemen umfangreiche Experimente durchgeführt. Die dabei festgestellte Genauigkeit und Zuverlässigkeit unserer Systeme führen uns zusammen mit ihrer Flexibilität und intuitiven Bedienbarkeit zu der Überzeugung, dass sie einen substanziellen Beitrag zur Anwendung von Navigation in der industriellen Automatisierung darstellen. Über den Einsatz in der Fabriklogistik hinaus könnten autonome Fahrzeuge, die mit Roboterarmen ausgestattet sind, in der Zukunft auch Manipulationsfähigkeiten zur Verfügung stellen. Hierdurch würde ein effizienter Konfigurationswechsel für ganze Prozessketten möglich, um zum Beispiel von der Herstellung von Fahrgestellen für Personenkraftwagen auf Fahrgestelle für Lastkraftwagen umzustellen, ohne dass hierfür eine aufwendige, manuelle Neuanordnung von im Fabrikboden verankerten Schweißrobotern nötig ist.

Abstract

The ongoing automation of manufacturing processes increases productivity and reduces physically demanding labor, a cornerstone for today's living standards and economic wealth in industrialized countries. However, industrial automation requires a costly setup phase. Therefore, it is only worthwhile when manufacturing large quantities of identical products. As globalization increases the pace of market changes and due to the trend towards product customization, there is a demand for more flexible, adaptable automation to retain global competitiveness for smaller batch sizes and changing products. Here, autonomous navigation is a key technology to reduce setup costs and increase flexibility.

In this thesis, we present contributions towards the deployment of mobile robot navigation in real-world industrial settings. We provide vehicle-based material transport between processes that is fast and intuitive to set up. Since our approaches only rely on wheel encoders and the mandatory safety scanners of an autonomous vehicle, they work without additional, inflexible infrastructure. We first propose a navigation system that enables omnidirectional vehicles to autonomously navigate with high accuracy while respecting safety constraints. Our system accounts for the dynamics of the vehicle when optimizing smooth, curvature continuous trajectories that are frequently updated in changing environments. We furthermore present a localization and docking method to achieve millimeter accuracy at target locations. For use cases that require automated vehicles to follow predefined routes, we propose an approach for intuitive instruction that relies on user demonstrations and can be combined with our navigation system to realize different levels of autonomy. For scenarios with predefined routes only, we further reduce the setup costs. Our teach-and-repeat framework needs no pre-built environment map and only a single, non-expert demonstration to track taught trajectories with millimeter accuracy.

For realistic estimates of the performance and reliability of navigation systems, we propose a benchmark protocol that instead of individual components evaluates complete systems. The tested system and a reference system interact with a carefully specified and scripted dynamic environment to ensure comparability between different evaluations.

We conducted extensive experiments for all our systems. Through the combination of high accuracy and reliability with flexibility and ease of use, they present a substantial contribution towards navigation for flexible automation. Autonomous vehicles can also be equipped with robotic arms, e.g., welding robots mounted on autonomous vehicles instead of bolted to the factory floor. Beyond logistics, navigation can thereby contribute to efficiently reconfigurable process chains and the vision of transformable factories.

Acknowledgements

I could not have completed this PhD thesis without the guidance and support of supervisor, colleagues, friends, and family. I am thankful for all those who made this thesis possible and in addition made my time as a PhD student in the Autonomous Intelligent Systems lab in Freiburg a pleasant and fun experience.

First, I want to thank Wolfram Burgard for giving me the chance to do a PhD in his lab, where he creates a stimulating and motivating environment filled with friendly and cooperative people. I appreciate his support that had the right balance between guidance and scientific freedom and provided me with valuable encouragement, advice and insights. Thank you also for the lessons in scientific writing, all the great ideas and the chance to attend so many interesting conferences all over the world.

I thank Cyrill Stachniss for being a reviewer for this thesis. I value our fruitful discussions during our common time in Freiburg and I am grateful for introducing me to the inner workings of European research projects.

I really enjoyed my collaboration with Boris Lau, Jörg Röwekämper, Markus Kuderer, Mladen Mazuran, Felix Endres, Marija Đakulović and Andrea Cherubini. Thank you for the hard work with nonetheless positive attitude during the long days before submission deadlines and for the cool things we made happen together. I am also grateful to the post-docs Cyrill Stachniss, Gian Diego Tipaldi, Luciano Spinello, and Giorgio Grisetti for their ever friendly ears and scientific as well as moral support.

A lot of my work was done on and for robotic platforms manufactured by KUKA Roboter GmbH, Augsburg. I am grateful for the good cooperation and mutual support, special thanks go to Patrick Pfaff, Rainer Bischoff and Klaus Miller.

I want to thank Jürgen Hess, Dominik Joho, Jörg Müller, Axel Rottmann, Boris Lau, Markus Kuderer, Nichola Abdo and Noha Radwann for being great and encouraging office mates, always open for scientific advice and discussions as well as the occasional bit of fun. Thank you also for making the candy jar happen together with me. Besides direct collaboration and office-sharing I want to thank Henrik Kretzschmar, Max Beinhofer, Pratik Agarwal, Michael Ruhnke, Bastian Steder, Rainer Kümmerle, Daniel Meyer-Delius, Kai Wurm, and Slawomir Sander for engaging discussions and the good time we had together. Jürgen Sturm and Jürgen Hess were great partners in discovering the wonderful world of the LaTeX packages tikz and pgfplots that I used for the figures in this thesis.

For their helpful comments and thoughtful feedback on earlier versions of this thesis I want to sincerely thank Markus Kuderer, Barbara Frank and Rainer Kümmerle.

For their patient support in administrative matters I thank Susanne Bourjaillat, Michael Keser, Evelyn Rusdea, Bettina Schug, Kristine Haberer and Daniela Wack.

In addition to supportive advisor and colleagues, support outside of the workplace played a non-negligible, important role for me. I owe special thanks to my family and Melanie for their continual and unconditional support and understanding. Thank you for providing a counterbalance and keeping me sane.

Finally, I gratefully acknowledge that this PhD thesis was partially supported by the European Commission under grant agreement numbers FP7-248258-First-MM and FP7-260026-TAPAS.

Contents

1 Introduction

Starting with the industrial revolution in the 18th century, humans began handing over tedious and physically demanding labor to machines. Through this ongoing process, we continuously improve human well-being and increase living standards. In today's economy, automation makes a wide range of products and amenities affordable by reducing the amount of human work needed. Positively phrased, automation increases the productivity of human work, i.e., it increases the output each hour of human work yields.

Traditional industrial automation requires a long and costly setup phase for machinery and its programming and is therefore only viable for rigid manufacturing processes with high throughput. A classical example are production lines for the chassis of automobiles. These production lines are only economic when producing a large number of identical vehicles. The goal of flexible automation, in contrast, is to become economically worthwhile also for smaller batch sizes and changing processes. Therefore, flexible automation requires low setup and adaption costs. Then, it can enable scenarios like switching a chassis production line from producing limousines to trucks on a daily basis according to market demands. Automation then also becomes applicable to the production of mini-series or heavily changing or customized products.

Flexible industrial automation is therefore widely regarded as a key technology in the context of today's worldwide challenges. The foreign markets opening with globalization go along with increased logistic efforts, world-wide competition and the need to rapidly adapt to changing customer demands. Here, more flexible automation can provide advanced factory logistics at competitive cost and help to increase the efficiency and adaptability and thereby the competitiveness of production processes (Koren, 2010). In today's aging societies and the thereby aging work force (Zaeh and Prasch, 2007), automation increases the amount of high value added jobs and reduces the amount of people required to perform physically demanding labor. Flexible automation reduces the amount of buffer zones and single-purpose areas needed in factories. Typically, a factory expends a significant percentage of its energy footprint for heating and light. Since the energy required for this scales with building size, more efficient use of space in factories directly translates into savings in energy costs and carbon footprint.

State-of-the-art systems in industrial automation, logistics and manipulation are typically very inflexible. Where automated vehicles replaced conveyor belts, they rely on additional infrastructure like beacons, guidance wires, reflective markers, visual landmarks or even guidance rails to follow their routes on the shop floor (Wurman et al.,

2008; Albrecht, 2011; Frog AGV Systems, 2015). The necessary infrastructure yields high costs for deployment, integration and maintenance, as well as for changing routes. Thereby it entails substantial investments for process adjustments. Systems for the industrial automation of manipulation tasks like for example welding or riveting are even more restricted. They typically consist of fenced-in manipulators that are bolted to the shop floor. Here, a change in a production process entails costly and time consuming rearrangement and reprogramming of machinery by experts.

For today's industrial automation, changeover times and costs are too high of an economic burden to flexibly adapt and switch between production processes. For truly flexible factories, industry needs autonomous vehicles that can reliably deliver materials and goods with high precision to easily adaptable target locations. Furthermore, one could combine such vehicles with robotic arms to create mobile manipulators. In addition to materials and goods, such vehicles could then also deliver manipulation capabilities, becoming multi-purpose machinery. This would enable true multi-purpose areas in factories that can quickly reconfigure and switch to different production processes, e.g., from limousine chassis to truck chassis production. This would enable timely reaction to ever-changing markets and customer demands.

1.1 Challenges in Industrial Robot Navigation

This thesis focuses on mobile robot navigation, a cornerstone technology for flexible industrial automation that enables vehicles to deliver materials as well as manipulation capabilities. While approaches to move a mobile robot to a goal location have existed for some time (Nilsson, 1984; Quinlan and Khatib, 1993; Fox et al., 1997), several challenges and requirements prevent the direct widespread deployment of flexible navigation to applications in the industrial context so far.

For successful application in industrial automation, autonomously navigating vehicles need to operate with high accuracy. Their behavior needs to be predictable and closely match planned motions that account for environment constraints like obstacle distances and limits on velocities. This is a requirement for the safety of the vehicles and their environment. Additionally, anticipatory behavior like smooth deceleration when approaching people or narrow passages also increases the acceptance and perceived safety by human personnel sharing the workspace of autonomous vehicles. Traditionally, robot navigation systems plan paths that are of purely geometric nature and employ a controller that reactively determines velocities to approximately follow such paths. The inaccuracies inherent to this form of path execution prevent accurate prediction of the robot's behavior and can lead to substantial deviations from the planned path, e.g., by cutting corners or overshooting curves (Stachniss and Burgard, 2002). For safety and for the comfort of human personnel, however, autonomous vehicles need to closely follow

plans that account for constraints imposed by their environment. At the same time, the motion of autonomous vehicles should be smooth to reduce wear and tear inflicted on the hardware. Furthermore, the planned behavior should ensure fast travel time and its computation needs to be efficient enough to allow frequent updates to react to changes in the environment.

In addition to accurate execution of planned motions to the goal, high positioning accuracy at the target location is a fundamental prerequisite for industrial automation. This capability enables autonomously navigating vehicles to link together individual processes to process chains, typically through some form of load exchange. To achieve high positioning accuracy at the junctions of process chains, current systems require overhead in form of additional sensor infrastructure and behaviors to ensure successful docking at a work station.

While fully autonomous navigation to target locations provides the highest flexibility in changing environments, there is also a demand for more constrained flavors of automated navigation in real-world applications. Decision makers and shop floor personnel are accustomed to the repetitive and low variance behavior of automated vehicles that strictly follow predefined routes. A challenge is to provide this capability of existing automated vehicles with albeit a heavily reduced infrastructure footprint (no guidance rails or wires, markers, additional sensors). This constitutes an attractive starting point for a transition towards more flexibility while maintaining backwards compatibility with existing processes and expectations.

There are also scenarios in which following predefined routes is the more economic choice since it can be too costly or impractical to model the required constraints for the behavior of fully autonomous vehicles. Consider for example an environment with dedicated driving space for vehicles, indicated by lane markings on the factory floor. Here, the corresponding rule set to constrain autonomous motion would either need to be manually encoded into the internal representation of the environment or one would need to employ additional sensors to perceive the lane markings. Generally, if scenarios feature constraints or rule sets that are not readily available in encoded form or not perceivable without additional dedicated sensors, one has to trade off between the benefits and costs of full autonomy and predefined routes.

To increase flexibility in automation where predefined routes are necessary or desired for the reasons discussed above, we need to achieve sufficiently high accuracy without additional infrastructure like guidance rails, wires, or markers and the corresponding sensors. Then, costs for deployment and reconfiguration of such systems decrease substantially. To fully exploit this increase in flexibility, instruction and reconfiguration of such systems needs to be fast and easy. Ideally, non-expert shop floor workers should be able to instruct these systems with a simple demonstration.

Finally, to convince decision makers to deploy autonomous vehicles in industrial automation, we require methods to assess the performance, robustness and reliability of

complete navigation systems. While it is possible to evaluate individual components like mapping, localization or path planning with recorded sensor data, the overall performance of a navigation system depends also on the interplay of its components and requires interaction between the robot and its environment. We need evaluations that provide realistic estimates of error bounds and failure modes through the use of suitable ground truth systems and appropriate evaluation scenarios. Furthermore, a methodology for meaningful comparisons between different navigation systems operating on different robots is key for informed decisions about the deployment of specific systems.

1.2 Contributions

In this thesis, we present contributions to the field of mobile robot navigation that address the challenges and questions discussed above to pave the way towards a widespread deployment of flexible industrial navigation in real-world applications. In the following, we outline the thesis and its main contributions.

Accurate and efficient navigation We first present an accurate navigation system for omnidirectional robots in Chapter 2. While we rely on established techniques for mapping and localization, our contribution focuses on the components of a navigation system that plan and execute motions. To this end, we use quintic splines in our system to represent smooth, curvature continuous paths. We augment the paths with velocity profiles that account for safety and platform constraints. By planning for the dynamics of the platform in our velocity profiles, we achieve highly accurate execution of our trajectories with an error feedback controller. We employ an optimization scheme to adapt the trajectory shape during planning to yield fast travel time. The efficiency of our approach allows to frequently update the trajectories while the robot is moving to react to changes in dynamic environments. Our extensive experimental evaluations demonstrate the robustness and accuracy of our system.

High accuracy localization and docking To address the challenge of high accuracy docking at target locations to successfully link individual processes in automation, we present a system for high accuracy localization in Chapter 3. We only rely on the mandatory safety scanners for mobile platforms and do not require any additional infrastructure. Our approach builds on the localization component of our navigation system. It refines the localization estimate with a scan matcher that compares the current safety scanner readings against reference data previously recorded at the target locations. The method is robust against moderate changes in the environment. For platforms with multiple safety scanners we can exploit redundancy to increase accuracy and robustness through fusing the estimates of individual scanners and performing consistency checks between them. In

our experiments, we combine this system with our navigation system and evaluate it with a highly precise motion capture studio, showing localization and positioning errors of a few millimeters at target locations.

Predefined routes through user demonstrations In the context of more constrained motion for automated vehicles, the path model employed in our proposed navigation system is also suitable to represent and execute predefined trajectories. While one possibility to generate such virtual rails is a graphical user interface that allows to draw paths on a map of the environment, we present a more intuitive instruction method in Chapter 4. We apply the programming by demonstration paradigm and propose a method to fit reference paths to user demonstrations. Our method applies non-linear least-squares optimization to accurately fit an instance of our path model to the reference. As we show in our experiments, we thereby require substantially fewer free parameters than standard approaches to achieve similar residual errors. In Chapter 4, we also show how to employ the optimization scheme from Chapter 2 to post-process and smooth user demonstrations and how to use it to robustly follow virtual rails. Through the combination of our navigation system with reference paths, different levels of autonomy become possible in the spectrum between virtual rails and full autonomy, e.g., to allow autonomous evasion of obstacles on a virtual rail within given bounds.

Teach-and-repeat navigation without a pre-built map The approaches discussed above all rely on a previously built, globally consistent metric map of the environment. For virtual rail application scenarios, in which the robot is to strictly follow a reference path, we present in Chapter 5 an approach that removes the time consuming mapping phase. We propose a teach-and-repeat framework that allows intuitive instruction of the robot through a single user demonstration and relies only on the mandatory safety scanners and wheel encoders of the platform. This approach extends ideas from the high accuracy localization from Chapter 3 towards a time-varying reference for a moving robot. Our real-world experiments show that our system achieves mean errors of a few millimeters when repeating user demonstrations. Non-expert users can use our one-button framework to quickly and intuitively instruct vehicles for navigation tasks. As an extension of the position-based system from Chapter 3, the approach is also applicable to capture and repeat more complex docking maneuvers.

Benchmark protocol for indoor navigation In Chapter 6, we present an evaluation methodology for complete navigation systems in changing environments. While this benchmark protocol allows to assess the reliability and robustness of individual navigation systems, it also establishes comparability between different systems operating on different robots and at possibly different experimentation sites. In addition to definitions for bench-

mark environments and their dynamics, we introduce a reference system to normalize performance across different experimentation sites and robots. In our experiments, we compare our navigation system with a reference system and the navigation system of a different research group.

Taken together, our contributions form an ensemble that in its entirety enables application scenarios for flexible robot navigation in the spectrum between full autonomy and virtual rails. We realize safe and highly accurate motion to the goal without additional infrastructure and we keep setup and adaption costs to a minimum for the task at hand. In addition to flexible navigation for a multitude of application scenarios our contributions also include means to assess the performance and robustness of our system and alternatives in a relevant manner. Thereby, the contribution of this thesis exceeds the sum of its individual parts through a portfolio of methods that as a whole pave the way for application of flexible mobile robot navigation in real-world industrial settings.

1.3 Publications

Parts of the research presented in this thesis have undergone international peer review. In the following, we list the corresponding publications in chronological order.

- C. Sprunk, B. Lau, P. Pfaff, and W. Burgard. Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

- C. Sprunk, B. Lau, and W. Burgard. Improved non-linear spline fitting for teaching trajectories to mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

- J. Röwekämper, C. Sprunk, G. D. 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 Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

- 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.

- C. Sprunk, J. Röwekämper, G. Parent, L. Spinello, G. Tipaldi, W. Burgard, and M. Jalobeanu. An experimental protocol for benchmarking robotic indoor navigation. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2014.

- F. Endres, C. Sprunk, R. Kümmerle, 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.

- M. Mazuran, C. Sprunk, W. Burgard, and G. D. Tipaldi. LexTOR: Lexicographic teach optimize and repeat based on user preferences. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

The following publications of the author of this thesis present work related to robot navigation. They are, however, outside of the main focus of this thesis and therefore not covered in detail. The works concern the application of splines in a different navigation context and configuration space representations that allow efficient collision checks and generation of initial paths in changing environments and for arbitrarily shaped robots and their payloads.

- B. Lau, C. Sprunk, and W. Burgard. Improved updating of Euclidean distance maps and Voronoi diagrams. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

- B. Lau, C. Sprunk, and W. Burgard. Incremental updates of configuration space representations for non-circular mobile robots with 2D, 2.5D, or 3D obstacle models. In *Proc. of the European Conf. on Mobile Robotics (ECMR)*, 2011.

- M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

- B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61 (10):1116–1130, 2013.

- M. Đakulović, C. Sprunk, L. Spinello, I. Petrovic, and W. Burgard. Efficient navigation for anyshape holonomic mobile robots in dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

- M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard. Online generation of homotopically distinct navigation paths. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

- C. Sprunk, B. Lau, and W. Burgard. The dynamicEDT3D package for updatable Euclidean distance transforms in 3D. Open source software released as part of the octomap package, `https://github.com/OctoMap/octomap`, 2015. Online, accessed 2015-08-18.

1.4 Collaborations

This thesis covers work that involved collaborations with other researchers, which we will outline in the following. As supervisor of this thesis Wolfram Burgard has contributed to all its parts through scientific discussions. The navigation system presented in Chapter 2 is an extension of the author's diploma thesis on trajectory generation that was co-supervised by Boris Lau and Patrick Pfaff. The chapter takes up the basic ideas for the employed path model, velocity profiles and optimization approach, which the author substantially extends in this thesis. The extensions include novel constraints, a different approach to generating initial paths suitable for non-circular robots, an evaluation of different initial path planning approaches, an evaluation of different variants of trajectory optimization, comparison with a different method of trajectory generation, experiments for individual components of the controller used for trajectory execution and an extensive evaluation of the techniques combined into a complete navigation system.

The work on the system for accurate localization at target locations presented in Chapter 3 is a collaboration with Jörg Röwekämper and was supported by discussions with Gian Diego Tipaldi, Cyrill Stachniss and Patrick Pfaff. Jörg Röwekämper realized the localization module for experiments with the omniRob platform while the author of this thesis focused on the integration with the navigation system and the motion capture studio. The author also contributed the covariance intersection for multiple laser scanners and the use of reference data across vehicles with different calibration, as well as a realization of the approach for the experiments with the larger-scale omniMove platform.

The approach for fitting trajectories to user demonstrations that we present in Chapter 4 was developed in collaboration with Boris Lau, who contributed with discussions and the realization of the baseline approach for spline fitting.

For the teach-and-repeat framework presented in Chapter 5 of this thesis, the author discussed the approach with Gian Diego Tipaldi and Andrea Cherubini, who contributed expertise in control theory.

The benchmark for robot navigation that we present in Chapter 6 is a collaboration with Jörg Röwekämper, Gershon Parent, Luciano Spinello, Gian Diego Tipaldi and Mihai Jalobeanu. Gershon Parent and Mihai Jalobenau contributed the marker-based ground truth system from Microsoft Research and conducted the experiments in Redmond with the Microsoft Research prototype navigation system and the reference system. With equal contributions, the author of this thesis and Jörg Röwekämper instantiated the protocol for the experiments in Freiburg and conducted the benchmark runs with the omniRob system as well as with the reference system. The author of this thesis adapted the complete navigation system for use with the benchmark ground truth software, iterated on the configuration of the reference system with Gershon Parent and conducted the evaluation of the benchmark runs in Freiburg. All collaborators were involved in discussions about the nature, suitability and instantiation of navigation challenges.

2 Accurate and Efficient Navigation in Industrial Environments

Flexible logistics are widely regarded as a key technology to increase adaptability and cost efficiency of today's factories. For example, fully autonomous transport vehicles aim to gradually replace conveyor belts, guided vehicles, and manual labor. In this context, especially omnidirectional mobile robots are appealing thanks to their advanced maneuvering capabilities. In industrial applications, however, accuracy as well as safety and efficiency are key requirements for successful navigation systems. In this chapter, we present an accurate navigation system for omnidirectional robots. We design our system for accurate motion execution by devising smooth, curvature continuous trajectories, by planning appropriate velocities and by accounting for platform and safety constraints. In this way, it exploits the maneuvering capabilities of omnidirectional robots and optimizes trajectories with respect to time of travel. We present extensive experimental evaluations in simulation and in changing real-world environments to demonstrate the accuracy and robustness of our system.

Automated transportation is a cornerstone functionality for logistics in today's highly automated factories. Starting from conveyor belts, industry is gradually moving to automated guided vehicles (AGVs) that provide higher logistic flexibility and a reduced infrastructure footprint. However, many of such AGVs still require additional infrastructure to travel along their predefined routes on the shop floor. This includes optical markers or guidance wires mounted on the floor, walls, or the ceiling. With industry pushing automation from mass products towards more flexible, changing production processes, the high setup costs for reconfiguring AGV routes tend to make automation of logistics with these platforms impractical. To make automation worthwhile in such changing production environments that are typically shared with human workers, mobile robots have to autonomously, accurately and safely find their route to the designated goal.

Omnidirectional platforms can perform complex and efficient maneuvers in confined spaces due to their ability to drive in any direction. Therefore, we explicitly consider the capabilities of omnidirectional robots. In this chapter, we present an all-around

Figure 2.1: The KUKA Moiros robot using our system for accurate omnidirectional navigation during a demonstration at the Hannover Messe 2013 fair. Our system accounts for platform and safety constraints and reliably navigated for several days in this inspection task for wind turbine blades.

system for accurate autonomous navigation that we tailor to leverage the characteristics of omnidirectional robots, from path planning to trajectory execution. The core factors in the targeted use case of industrial shop-floor navigation are accuracy, safety, and efficiency.

We address accuracy of robot motion by planning trajectories with velocities, accounting for platform constraints such as acceleration and maximum wheel velocities. Thereby, our system can accurately follow these planned trajectories. The trajectories are smooth and curvature continuous, this is a key factor for accurate execution and to reduce wear and tear on mobile robots that work on an industrial factory floor for extended periods of time. We address safety of robot motion by considering safety constraints for the trajectories. In particular, we impose obstacle-dependent speed limits. The closer the robot comes to obstacles, the slower it has to drive. Driving slower enables higher accuracy in trajectory execution near obstacles. Furthermore, this behavior increases acceptance of the robot by human workers through an increased perceived safety.

In large robotic platforms, like the one shown in Figure 2.1, even low rotational velocities generate large tangential velocities of the robot body. In our system we take into account the footprint of the robot body and constrain, during the motion planning, the maximum velocity achievable by any point on the contour.

In addition to accuracy and safety, another objective of our system is efficiency of robot motion. Our system therefore performs optimization of travel time while satisfying all of the above constraints. For increased efficiency, we also explicitly take into account the ability of omnidirectional robots to independently translate and rotate. It is also important

that autonomous robots that share the factory floor with human workers can cope with changing environments and unmapped obstacles. Therefore, our system regularly updates trajectories while the robot is moving. It smoothly joins the updated trajectory with the current trajectory and performs the necessary planning within a given short time deadline.

The system presented in this chapter has been experimentally evaluated and quantified in all its key components showing high accuracy with efficient computation. We present also a real-world experiment consisting of almost 3 km distance traveled in 2 hours. Additionally, the system had direct impact in the industry and is currently embedded on the industrial mobile omnidirectional platform KUKA omniRob, a smaller-scale version of the robot shown in Figure 2.1.

The remainder of this chapter is organized as follows. We first present an overview of the proposed system in Section 2.1. Then, we present the approaches that we rely on for mapping and localization in Section 2.2. In Section 2.3 we describe our optimization-based approach to trajectory generation. The system sends the resulting trajectories to an error feedback controller for execution, which we present in Section 2.4. Section 2.5 contains the experimental evaluation of our system and Section 2.6 discusses related works before we conclude the chapter with Section 2.7.

2.1 Overview

Our navigation system includes a mapping and localization module, a trajectory generation module and a trajectory execution module. For operation, the robot relies on a map of the environment. The robot uses this map for localization and robustly updates it during operation (Section 2.2). We outline the different steps and components used for computing and executing trajectories in Figure 2.2.

In a first step, our system employs a geometric path planner to find a collision free path from the current robot configuration to the goal (Section 2.3.1). After expressing the initial path with our spline-based path model (Section 2.3.2), we compute a velocity profile for the geometric path that respects vehicle and safety constraints to generate a feasible time-parameterized trajectory suitable for accurate execution (Section 2.3.3). The velocity profile also provides the time of travel of the current trajectory. An optimization procedure incorporates this into its cost function and iteratively improves the shape of the path (Section 2.3.4). Once the current trajectory cannot be further improved by the optimization or there is no more planning time left, an error feedback controller receives the optimized trajectory for execution (Section 2.4).

We decouple the generation of trajectories and their execution such that the feedback controller is executed at a higher frequency (e.g., 35 Hz) while we generate updated trajectories at a lower frequency (e.g., 1.5 Hz), see the dashed line in Figure 2.2. This way, trajectory execution in the odometry frame of the robot can achieve high accuracy through

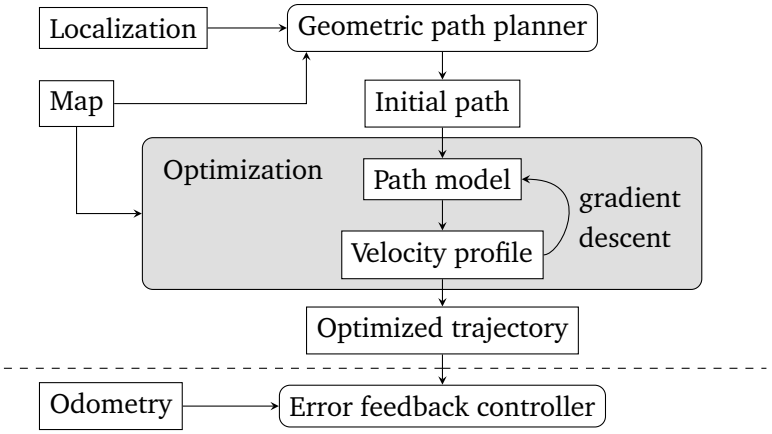


Figure 2.2: Overview of our system for accurate navigation of omnidirectional robots. The dashed line marks the separation between trajectory generation and execution. The feedback loop for trajectory execution runs at a higher frequency than the (re-)planning of trajectories.

a high-frequency feedback loop. Regular re-planning of trajectories in the global frame accounts for accumulating drift in the odometry frame, localization noise and changes in the environment.

When our system plans an updated trajectory for the already moving robot, it first decides on the time frame within which the new trajectory should be ready. Then, using localization and the currently executed trajectory, it predicts the configuration of the robot at that point in time, including position, orientation, velocity, and curvature. To ensure a smooth transition to the new trajectory, we use this future configuration of the robot as starting configuration of the robot in the new trajectory. Therefore, it is important that our path model supports setting the curvature at the start of the path, see Section 2.3.2.

2.2 Mapping and Localization

The environment representation of our system consists in a 2D grid map. We employ a graph-based formulation of the simultaneous localization and mapping (SLAM) problem to build a grid map from sensor data as proposed by Lu and Milios (1997), Dellaert and Kaess (2006) and Grisetti et al. (2010). In this formulation, the poses of the robot correspond to nodes in a graph and edges represent constraints between poses. These constraints originate either from wheel encoders or one extracts them from other sensor measurements. In our case, we compute incremental motion constraints from laser range readings with a scan matcher, see for example Censi (2008). We detect constraints that correspond to re-visited places in the environment (loop closures) by employing FLIRT features as proposed by Tipaldi et al. (2013, 2014). We optimize the resulting graph with

the g2o framework by Kümmerle et al. (2011).

For localization during operation, our system uses a Monte Carlo localization (MCL) approach as proposed by Dellaert et al. (1999b). MCL uses a set of samples called particles to represent the belief of the system about its state. Whenever the robot moves, the approach propagates the particles according to the motion model of the robot. Whenever the robot takes a measurement of its environment, it updates the particle weights according to the likelihood of the measurement for the respective particle. In particular, our particle filter employs an odometry motion model and a beam-endpoint model, as defined by Thrun et al. (2005). The performance of MCL depends on the number of particles used to represent the belief. For efficiency and robustness, we apply the KLD sampling suggested by Fox (2003). This approach performs online adaptation of the number of employed particles to save computational resources while limiting the error introduced through the sample-based representation.

To account for dynamic changes in the environment, we continually update the map meanwhile the robot moves. We use a robust and effective strategy that always combines the initial grid map with the latest range measurements. Based on the current pose of the robot, we set the grid cells that correspond to beam end points of the range measurements as occupied. When new range measurements are available, we revert the changes that originated from the previous measurements. We propagate these incremental grid map updates to other map representations used for efficient path planning and collision checking: the distance map and the discretized Voronoi diagram of the map. The distance map represents the distance to the closest obstacle in each cell, we use this information for collision checking. The Voronoi diagram provides a skeletonization of the environment that contains all points that are equidistant to at least two obstacles. We employ the Voronoi diagram to plan an initial path.

2.3 Trajectory Generation

This module is the backbone of the robot navigation system. It takes care of computing the trajectory shape and the velocity profile for the robot. The resulting trajectories are ready for execution by robot control. The trajectory generation module computes trajectories for omnidirectional robots and accounts for constraints to enable accurate and safe motion while optimizing for time of travel.

2.3.1 Path initialization

As shown in the overview in Figure 2.2, we initialize the trajectory generation system with a path computed by a geometric path planner. For this step, our system is independent of the choice of the path planner.

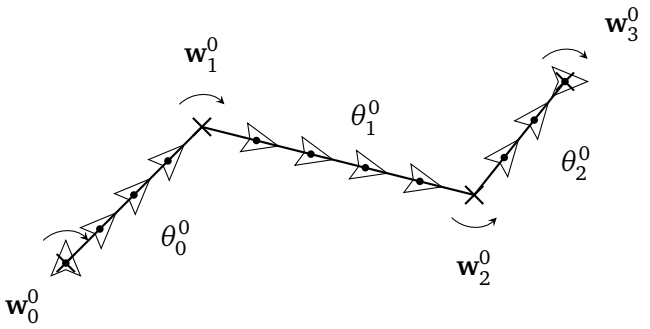


Figure 2.3: Our system uses an initial path generated by a geometric path planner as initialization. The paths consist of translations with constant orientations θ_i^0 alternating with turns on the spot at the waypoints $\mathbf{w}_i^0$ of the initial path. The darts indicate the orientation of the robot.

The system accepts an initial path as a sequence of $M+1$ waypoints $\mathbf{w}_i^0$, $i \in \{0,\dots,M\}$, connected by M straight line segments, see Figure 2.3 for a visual explanation. We expect the path to be traversable without collisions when moving with constant orientation θ_i^0 between $\mathbf{w}_i^0$ and $\mathbf{w}_{i+1}^0$ for all segments. In this initial path, the robot changes its orientation by turning on the spot at the waypoints.

In practice, for path initialization, we use an efficient path planner that employs a discretized Voronoi diagram of the environment. In essence, the Voronoi diagram is a graph that contains all points in the environment that are equidistant to at least two obstacles. The motivation behind our approach is to efficiently generate a conservative initialization for the path. Paths based on the Voronoi diagram are conservative in the sense that they emphasize distance to obstacles. Using such paths as initialization, our optimization-based trajectory generation can then decrease obstacle distance for more efficient paths. If however the planning time for trajectory optimization ends before the optimization converges to a balance between efficiency and safety, the resulting trajectory will be biased towards safety.

In a first step, our path planner connects start and goal to the discretized Voronoi diagram and searches the graph for a path that guarantees obstacle clearance for the incircle of the robot contour. Since the path at this point consists of a concatenation of grid cells, the planner converts it to straight line segments based on the Douglas-Peucker algorithm (Hershberger and Snoeyink, 1992). In the next stage, the planner checks for collisions when moving the robot along this path by following the orientation of its line segments and turning on the spot at their join points. Aligning the robot with the orientation of the line segments is the most conservative variant of augmenting the path with orientations, however one cannot guarantee the absence of collisions. In cases where the robot collides with the environment, an A* planner replaces the colliding segment of the path. This local A* planner accounts for a configuration space consisting of positions

collides_circ(circle C, distance map D**)**
  $(c_x, c_y) \leftarrow$ center(C)
  return $\big(D(c_x, c_y) \leq \text{radius}(C)\big)$

collides_rect(rectangle R, distance map D**)**
  $(c_x, c_y) \leftarrow$ center(R)
  if $D(c_x, c_y) \leq r_i$ **then**
    return *true*
  else if $D(c_x, c_y) > r_o$ **then**
    return *false*
  compute rectangles R_1, R_2
  return $\big(\text{collides\_rect}(R_1, D) \vee \text{collides\_rect}(R_2, D)\big)$

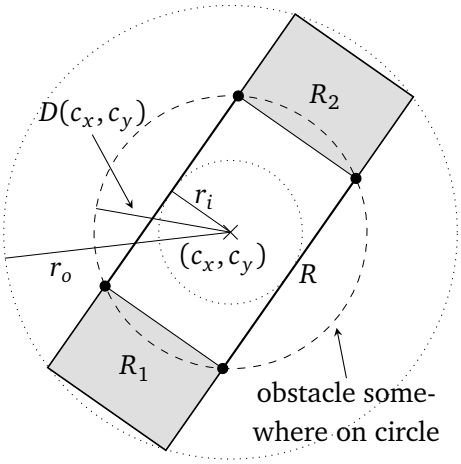


Figure 2.4: Efficient collision checks for circular and rectangular elements. The drawing shows a rectangular element R with inner and outer diameter r_i and r_o. Only if the obstacle distance $D(c_x, c_y)$ at the center of R is between r_i and r_o, the result depends on recursive collision checks of the subparts R_1 and R_2 of R. Exceeding recursion can be pruned assuming collision.

with orientations. Since the A* planner returns a sequence of grid cells, we again apply the Douglas-Peucker algorithm to convert the path into a sequence of orientation-augmented line segments. Here, in addition to the approximation error we also rely on collision checks when deciding whether or not to merge line segments in the Douglas-Peucker algorithm.

To retain completeness with this planning approach one can disconnect the corresponding edge of the discretized Voronoi diagram should the A* planner fail to resolve a collision for the respective segment and query the Voronoi diagram for an alternative path. Our path planner is similar in spirit to the one proposed by Foskey et al. (2001), who bridge segments with a sampling-based planner in 3D workspaces.

Collision checking

In our system, not only the path initialization step but also other components of the trajectory generation make heavy use of collision checking. Therefore, we propose an efficient method for collision checking, shown in the pseudo code and the illustration in Figure 2.4. The idea is to leverage an approximation of the shape of the robot and its payload as a set of circles and rectangles. These elements can be efficiently checked against a dynamically updated distance map $D(x, y)$ of the environment. A distance map contains in each cell (x, y) the distance $D(x, y)$ to the closest obstacle and can be efficiently updated when the environment changes, see for example Lau et al. (2013) who maintain Euclidean distance transforms for changing environments.

Circular elements with radius r and center (c_x, c_y) collide with an obstacle when $r > D(c_x, c_y)$. For rectangular elements centered at (c_x, c_y) we perform collision checking with a recursive procedure, see Figure 2.4: If the obstacle distance $D(c_x, c_y)$ (dashed circle) is larger than the circumcircle of the rectangle (radius r_o), the rectangle is free of collisions. If $D(c_x, c_y)$ is smaller than the incircle of the rectangle (radius r_i), the rectangle collides. In the remaining case, i.e., $D(c_x, c_y)$ is in-between the radius of the incircle and the circumcircle, we recursively check two rectangular areas for collisions (R_1, R_2, shaded). We determine their dimensions and center points with basic trigonometry. To prune the recursive procedure for obstacles close to the contour of the robot one can assume collision at a certain level of recursion.

In case of more complex robot shapes, it can be beneficial to employ incrementally updatable configuration space representations that trade increased memory consumption for faster collision checks. After an initial overhead, Lau et al. (2013) achieve substantial speedups through updatable convolutions of the robot shape with the environment map for discrete orientations.

2.3.2 Compact path model

In this section, we present the model we use to represent paths of the robot in the environment. In practice, a path $\mathbf{s}(u) \in \mathbb{R}^2 \times [0, 2\pi) = SE(2)$ is a progression of robot poses $\langle x, y, \theta \rangle$ in global coordinates as a function of an internal parameter $u \in [0, M] \subset \mathbb{R}$. To avoid confusion: The path $\mathbf{s}(u)$ determines where the robot drives in the environment but it does not determine *when* the robot arrives at a certain place and with which velocity. This is determined by velocity profiles, which we introduce in Section 2.3.3.

The goal of our path representation is to model a path in the environment with a small set of parameters. With our representation, changes in these parameters result in modifications of the path and allow fine adjustments of its shape. Furthermore, we enforce continuity in curvature for accurate and smooth execution.

We base our path model on quintic Bézier splines. In the Bézier formulation, each segment $\hat{\mathbf{s}}_i$ of a spline is a polynomial of degree five and it is defined for its parameter $u_i \in [0, 1]$ by six control points $\mathbf{p}_0, \ldots, \mathbf{p}_5$:

$$\begin{aligned}\hat{\mathbf{s}}_i(u_i) = (1-u_i)^5\mathbf{p}_0 + 5(1-u_i)^4 u_i \mathbf{p}_1 + 10(1-u_i)^3 u_i^2 \mathbf{p}_2 \\ + 10(1-u_i)^2 u_i^3 \mathbf{p}_3 + 5(1-u_i) u_i^4 \mathbf{p}_4 + u_i^5 \mathbf{p}_5 \,.\end{aligned} \tag{2.1}$$

The factors for each control point are also known as the Bernstein polynomials of degree five. As observable from Eq. (2.1), a segment interpolates between its control points, starting at $\hat{\mathbf{s}}_i(0) = \mathbf{p}_0$ and ending at $\hat{\mathbf{s}}_i(1) = \mathbf{p}_5$, see also Figure 2.5. The spline segment is not passing through its intermediate control points. These directly and independently determine the first and second derivative of the segment at the start and end of the

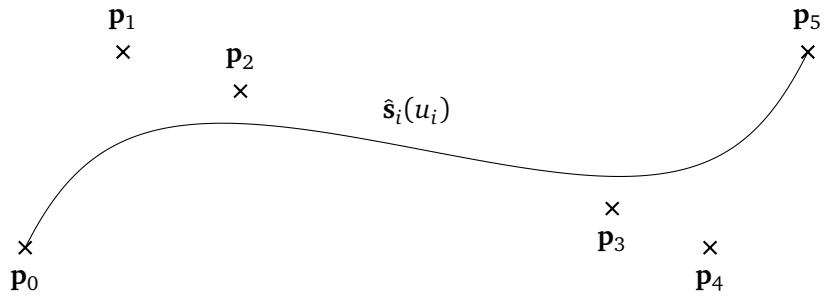


Figure 2.5: A segment $\hat{\mathbf{s}}_i$ of a quintic Bézier spline as used in our path model. The figure shows a 2D segment along with its control points $\mathbf{p}_0, \ldots, \mathbf{p}_5$. A Bézier segment passes through its first and last control point. Eq. (2.2) shows how the inner control points influence the first and second derivative at start and end.

segment:

$$\begin{aligned} \hat{\mathbf{s}}_i'(0) &= 5(\mathbf{p}_1 - \mathbf{p}_0), \quad \hat{\mathbf{s}}_i''(0) = 20(\mathbf{p}_0 - 2\mathbf{p}_1 + \mathbf{p}_2), \\ \hat{\mathbf{s}}_i'(1) &= 5(\mathbf{p}_5 - \mathbf{p}_4), \quad \hat{\mathbf{s}}_i''(1) = 20(\mathbf{p}_3 - 2\mathbf{p}_4 + \mathbf{p}_5). \end{aligned} \tag{2.2}$$

We ensure continuity at the join points between individual path segments by requiring control points of neighboring segments to satisfy $\hat{\mathbf{s}}_i(1) = \hat{\mathbf{s}}_{i+1}(0)$. Similarly, we enforce constraints for the first and second derivative of the path and thereby achieve curvature continuity along the complete path. When concatenating the path $\mathbf{s}(u)$ from individual segments $\hat{\mathbf{s}}_i(u_i)$, we map appropriately from the internal parameter u to the respective u_i. Through modification of $\mathbf{p}_1$ and $\mathbf{p}_2$ of the first segment, we have control over the first and second derivative at the start of the path. Thereby, we can control the curvature at the start of the path: a key property of our path model when planning updated trajectories.

Considering the continuity constraints introduced above, our path model has as free parameters: the position and first and second derivative at its start, end, and at the join points of path segments, which we call waypoints $\mathbf{w}_i$ as in the initial path. In practice, we reduce the number of model parameters with some heuristics. The first heuristic determines the direction of the first derivative at an inner waypoint $\mathbf{w}_i$ to be perpendicular to the angular bisector of the angle formed with the adjacent waypoints $\mathbf{w}_{i-1}, \mathbf{w}_{i+1}$, see the arrows in Figure 2.6. The magnitude of the first derivative is a free parameter and represented through a scalar elongation factor e_i in the path model. As shown in Figure 2.6, the elongation factor influences the radius of the curve performed at waypoint $\mathbf{w}_i$. The second heuristic determines the second derivative at inner waypoints. To achieve smooth curves we mimic the behavior of cubic splines which minimize the integral of the second derivative's absolute value on a segment. Cubic splines, however, connect with discontinuities in second derivative and hence in curvature at waypoints. To achieve a similar behavior with quintic splines, we set the second derivatives of the spline segments at a waypoint as a weighted average of the second derivatives that adjacent

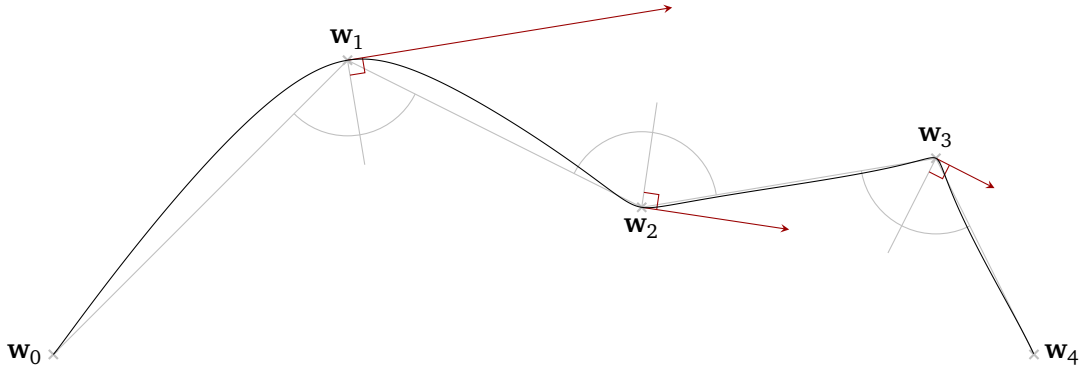


Figure 2.6: Our path model sets the direction of the first derivative at an inner waypoint (arrows) to be orthogonal to the bisector of the angle formed by the adjacent waypoints. The magnitude of the first derivative (arrow length) is a free parameter and influences the wideness of curves.

cubic spline segments would yield, if created at identical positions and with the same first derivatives at start and end of the segment. Figure 2.7 visualizes the effect of the heuristic, it depicts the quintic spline resulting from our path model as well as the cubic spline used to compute the second derivative at the join point of the segments. We determine the second derivative at an inner waypoint by a weighted average:

$$\mathbf{s}''(i) = \frac{\|\mathbf{d}_i\|}{\|\mathbf{d}_{i-1}\|+\|\mathbf{d}_i\|} \lim_{u \to i^-} \mathbf{s}''_{\text{cubic}}(u) + \frac{\|\mathbf{d}_{i-1}\|}{\|\mathbf{d}_{i-1}\|+\|\mathbf{d}_i\|} \lim_{u \to i^+} \mathbf{s}''_{\text{cubic}}(u), \tag{2.3}$$

where $\mathbf{d}_i = \mathbf{w}_{i+1} - \mathbf{w}_i$, and $\mathbf{s}_{\text{cubic}}$ is a cubic spline constructed with matching waypoints and first derivatives.

Our path representation explicitly models the position in the environment. The orientation of the robot is implicitly encoded by the first order derivative of the path. This model is suitable for differential drive and synchro-drive robots where the first derivative $\mathbf{s}'(u)$ of the path uniquely determines the orientation of the robot as $\theta(u) = \text{atan2}(\mathbf{s}'_y(u), \mathbf{s}'_x(u))$. However, this model does not fully leverage the capabilities of omnidirectional robots that can rotate independently of their translational motion. Thus, we extend the path model to explicitly represent the orientation of the robot as an independent component.

Modeling independent rotation and translation

To explicitly represent the orientation of the robot, we add a corresponding dimension to the control points of the quintic Bézier segments, such that $\mathbf{p}_0, \ldots, \mathbf{p}_5 \in \mathbb{R}^3$ in Eq. (2.1). At first glance, augmentation of the waypoints with a dedicated dimension would suffice to represent orientation independently of translation. However, the robot would continuously rotate between two waypoints as the connecting spline segment interpolates between the

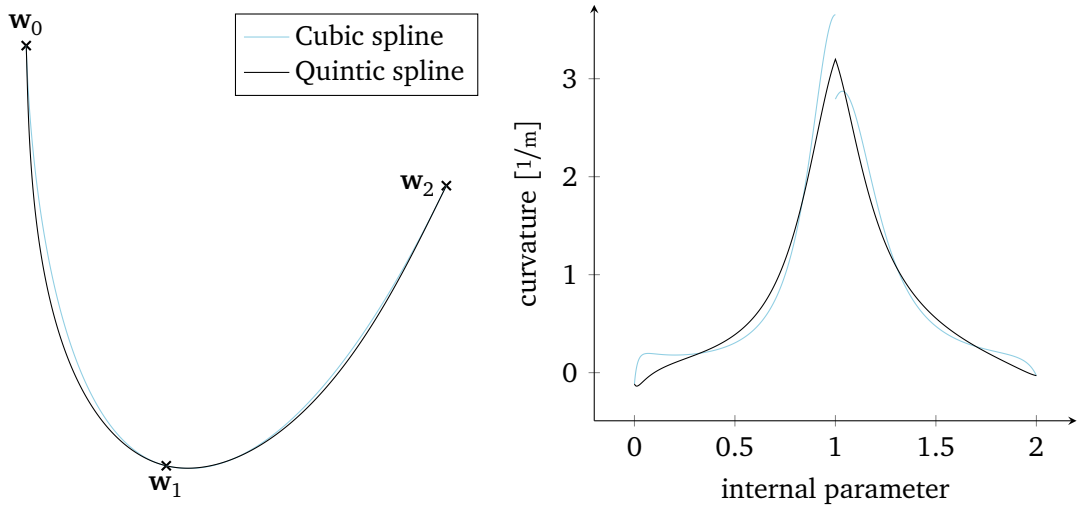


Figure 2.7: Our path model employs quintic splines and determines their second derivative at inner waypoints with a heuristic: We choose a weighted average of the discontinuous second derivative of a cubic spline constructed with matching waypoints and first derivatives, see Eq. (2.3). This way, we maintain curvature continuity while mimicking cubic splines.

orientations of adjacent waypoints. This behavior may hinder driving through narrow passages that require a fixed orientation.

Instead of inserting additional waypoints with identical orientation around narrow passages, our representation tackles this problem by a more flexible concept. Recall that the representation of the initial path provided by a geometric planner (see Section 2.3.1) intends for the robot to travel with constant orientation on a segment and to perform rotations on the spot only at the waypoints. To "distribute" the rotation from a waypoint to the adjacent segments, we introduce rotational-control-points r_i^{s} and r_i^{e} on the incoming and outgoing segments connected to each waypoint $\mathbf{w}_i$. These points mark the beginning (r_i^{s}) and the end (r_i^{e}) of the robot rotation from θ_{i-1} associated with the incoming segment of $\mathbf{w}_i$ to θ_i related to the outgoing segment. The idea is visually exemplified in Figure 2.8: In the configuration on the left r_i^{s} and r_i^{e} coincide with $\mathbf{w}_i$, resulting in a rotation on the spot. On the right, r_i^{s} and r_i^{e} are distant from $\mathbf{w}_i$ (green arrows), this causes a smooth rotation during translation. When r_i^{e} of waypoint $\mathbf{w}_i$ coincides with r_{i+1}^{s} of $\mathbf{w}_{i+1}$, this results in an uninterrupted rotation along the segment.

Technically, we realize rotational-control-points by subdividing the spline segments in a way that leaves the positional part unaffected and affects only orientation to achieve the desired behavior. For this, we evaluate the spline segment with its derivatives at the subdivision point and use these values as end and start parameters at the new join point. Due to the change in parameterization, we need to scale the derivatives as follows. Suppose that we subdivide spline segment i, $i \in \{0, \dots, M-1\}$ at internal parameter

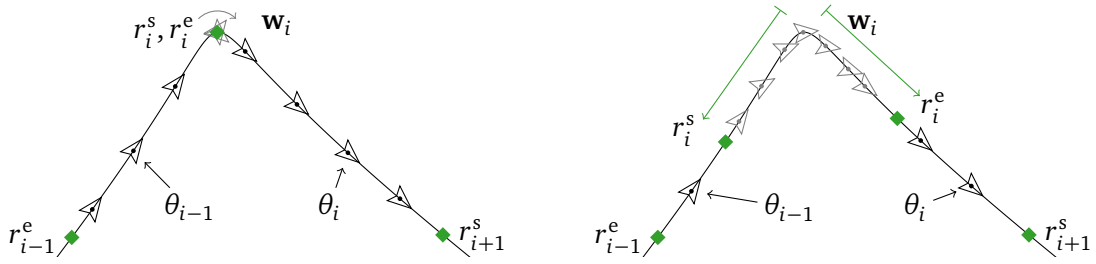


Figure 2.8: Influencing simultaneous rotation and translation with rotational-control-points (green diamonds): r_i^s and r_i^e coinciding with waypoint $\mathbf{w}_i$ yields a turn on the spot (left). If r_i^s and r_i^e move away from $\mathbf{w}_i$ rotation takes place simultaneously with translation (right). The darts indicate robot motion with constant (black) and changing (gray) orientation.

$u_d \in (i, i+1)$. The new sub-segments span the entire domain of the internal parameter: $[i, i+1]$ and $[i+1, i+2]$ instead of $[i, u_d]$ and $[u_d, i+1]$. To retain the shape of the spline segments we have to scale the k-th derivative of the first sub-segment by $(u_d - i)^k$ and the k-th derivative of the second sub-segment by $(i+1-u_d)^k$.

When subdividing segments at rotational-control-points, we set the orientation component of the first and second derivative to zero, as these points mark a transition between a segment with constant orientation and a segment with varying orientation. At the waypoints, we set the first derivative to point into the direction of rotation, $T_{i,\theta} = \frac{1}{2} e_\theta \left(\theta_i - \theta_{i-1}\right)$, where e_θ is a parameter of the path model for each waypoint. The second order derivative of the orientation component follows the same heuristic as its translational counterpart at every waypoint.

The employed spline segments interpolate between the orientations at rotational-control-points and the waypoints. To prevent unintended turns into the wrong direction, we add appropriate multiples of 2π to the orientation values at these points. For example, we replace a turn $(\pi - \epsilon) \to (-\pi + \epsilon)$ with the turn $(\pi - \epsilon) \to (\pi + \epsilon)$ for $0 < \epsilon < \pi$ to model the small turn of 2ϵ.

Influencing the amount of rotation along the path

The rotational-control-points r_i^s and r_i^e introduced above determine *where* on the path the robot rotates. However, the initial values θ_i^0 for the constant orientation θ_i between waypoints given by the geometric path planner can be conservative. Adjusting the θ_i to reduce the total amount of rotation can reduce travel time or energy consumption. In the example in Figure 2.9, the orientation behavior given by the initial path requires the robot to drive forward on every segment of the path (top). The bottom part of the figure shows an orientation behavior that minimizes the total amount of rotation. Here, we introduce a parameter for our path model to smoothly change between these two extremal orientation

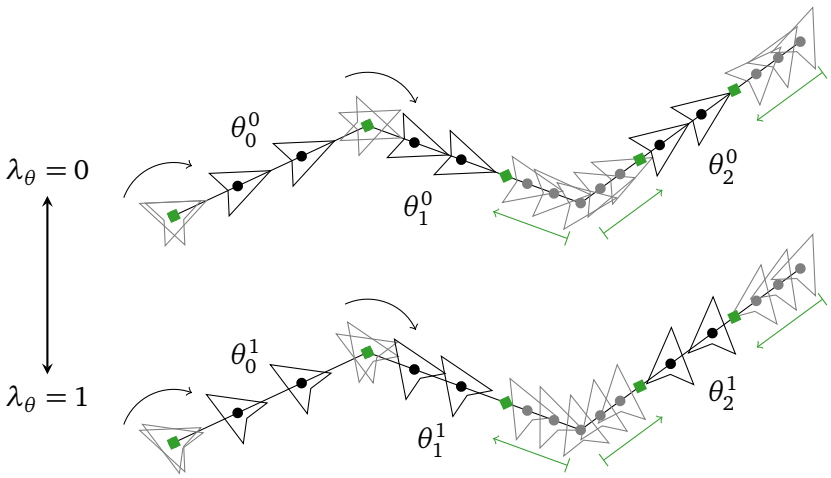


Figure 2.9: Paths with different rotation behaviors. *Top:* initial paths use $\lambda_\theta = 0$ to orient the robot as specified by the waypoint planner. *Bottom:* minimized change of orientation with $\lambda_\theta = 1$. The optimization can adjust λ_θ to interpolate between these two extremes. For reference, the figure also shows the location of rotation control points (green diamonds).

behaviors.

Usually, the orientation of the robot is predetermined at start and goal. At some intermediate segments it can also be fixed due to a narrow passage requiring a particular orientation. We therefore split the path at such segments and consider the rotational behavior of the resulting sub-paths independently. For a sub-path with waypoints $\mathbf{w}_i$, $i \in \{0, \dots, M\}$, we compute the orientations θ_i^1 that correspond to a minimal change of orientation. They interpolate the given start and end orientation θ_{start} and θ_{end} of the sub-path, as shown in Figure 2.9 (bottom):

$$\theta_i^1 = \theta_{\text{start}} + f_i \left(\theta_{\text{end}} - \theta_{\text{start}}\right) . \tag{2.4}$$

Here, $f_i \in [0,1]$ stands for the fraction of path length up to the middle of the i-th segment:

$$f_i = \frac{\left(\sum_{k=1}^{i} \|\mathbf{w}_k - \mathbf{w}_{k-1}\|\right) + \frac{1}{2}\|\mathbf{w}_{i+1} - \mathbf{w}_i\|}{\sum_{k=1}^{M} \|\mathbf{w}_k - \mathbf{w}_{k-1}\|} . \tag{2.5}$$

For every sub-path we introduce a parameter $\lambda_\theta \in [0,1]$ that blends the rotational behavior between the initial (θ_i^0) and minimized (θ_i^1) values according to

$$\theta_i = (1 - \lambda_\theta)\theta_i^0 + \lambda_\theta \theta_i^1 . \tag{2.6}$$

For $\lambda_\theta = 0$, we use the initial θ_i^0 orientations in the corresponding sub-path as shown in Figure 2.9 (top). Choosing $\lambda_\theta = 1$ uses the θ_i^1 and thus achieves a minimal change

of orientations as depicted in Figure 2.9 (bottom), which however might not be free of collisions. The optimization starts with all $\lambda_\theta = 0$ and continuously changes these parameters to obtain more efficient paths that are still collision-free. Note that this parameter does not affect whether rotations occur on the spot or while translating. In Figure 2.9 (top vs. bottom), compare the black darts (constant orientation) and gray darts (changing orientation): Their position along the path is identical, only their orientation differs. We set the orientation component $\mathbf{w}_{i,\theta}$ of a waypoint $\mathbf{w}_i$ to the average value of the adjacent segment orientations, $\mathbf{w}_{i,\theta} = \frac{1}{2}\left(\theta_{i-1} + \theta_i\right)$.

Modeling the initial path

As described in Section 2.3.1 and shown in Figure 2.2 we initialize our system with a path generated by a geometric path planner that consists of straight-line motions alternating with turns on the spot. To pass this initial path to the optimization, we transform it into an instantiation of our compact path model.

The geometric path planner provides a path to the goal that is free of collisions. To ensure a feasible starting point for the optimization we need to set the parameters of our path model in a way that closely approximates the initial path. We set the waypoints $\mathbf{w}_i$ of our path model to the points provided by the path planner, $\mathbf{w}_i = \mathbf{w}_i^0$. By selecting low values for e_i we achieve a close approximation of a straight-line path with sharp curves. One can control the approximation error by subdividing longer segments with additional waypoints. To correctly model the orientation behavior, we set the rotational-control-points to coincide with their waypoints for turning on the spot (see Section 2.3.2) and set all $\lambda_\theta = 0$ to enforce the segment orientations given by the path planner (see Section 2.3.2). We set the initial scaling factor for the orientational component of the first derivative at waypoints to $e_\theta = 1$ for the initial path. For an example of an initial path represented by our model, see the top row of Figure 2.10.

Starting from the initial path, the optimization moves inner waypoints to adapt the path to the environment, widens curves at waypoints by adapting e_i, distributes rotation along the path by moving r_i^{s} and r_i^{e} away from $\mathbf{w}_i$ and changes the amount of rotation performed along the path by adapting λ_θ. The resulting changes in path shape influence how fast the robot can travel along the path to the goal. In the next section, we introduce velocity profiles that determine the actual velocities of the robot along the path and thereby also control the time of travel to the goal.

2.3.3 Velocity profiles

As introduced in the previous section, a path $\mathbf{s}(u)$ defines a progression of poses for the robot from start to goal. Together with a mapping $u(t)$ from time t to the internal path parameter u, the path becomes a trajectory that defines the configurations of the robot

over time t. Thus $u(t)$, which is strictly monotonically increasing, defines a velocity profile and the time t_{travel} required to execute the trajectory. For a path $\mathbf{s}(u)$ defined by the parameters described in Section 2.3.2, we seek to compute the mapping $u(t)$ that corresponds to the fastest velocity profile that respects robot and safety constraints.

The speed at which the robot traverses a path $\mathbf{s}(u)$ is determined by the mapping $u(t)$, such that $\mathscr{T}(t) := \mathbf{s}(u(t))$ determines the pose and velocities of the robot over time t. A trajectory $\mathscr{T}(t)$ defines velocities by its first derivative with respect to time t,

$$\dot{\mathscr{T}}(t) = \mathbf{s}'(u(t))\,\dot{u}(t)\,, \tag{2.7}$$

where the dot and the prime denote derivatives with respect to t and u, respectively.

We finely discretize $\mathbf{s}$ into a set of points, e.g., whenever the robot would travel more than 0.02 m or rotate more than 0.02 rad. For these points we compute the maximum $\dot{u}(t)$ that satisfies a set of constraints.

Maximum velocity

We can limit the translational and rotational velocities of the platform to $v_{\max}, \omega_{\max}$. This results in the following constraint for $\dot{u}(t)$:

$$\dot{u}(t) \leq \frac{v_{\max}}{\|\mathbf{s}'_{xy}(u(t))\|}, \quad \dot{u}(t) \leq \frac{\omega_{\max}}{|\mathbf{s}'_{\theta}(u(t))|}\,. \tag{2.8}$$

Obstacle-dependent velocity limit

For safety of robot motion, we impose an obstacle-dependent speed limit on the robot. After determining the distance of the robot contour to the closest obstacle, we limit its velocity to an upper bound that allows coming to a complete halt before collision with an obstacle. The computation assumes constant acceleration and allows to parameterize a reaction time and a maximum deceleration for these braking maneuvers.

Maximum velocity of robot contour

When driving with larger robots and in workspaces that are shared with humans it can be beneficial to not only limit the velocity of the robot at its center of motion as introduced above but to also account for the effect of rotational velocity at the contour of the robot: We limit the maximum translational speed of *any* point of the robot with respect to the environment.

If the center of motion of the robot moves with velocity $\mathbf{v} = (v_x, v_y, v_\theta)^\top$, a point $(p_x, p_y)^\top$ on the robot expressed relative to the robot's center of motion moves with velocity

$$\mathbf{v}_p = (v_x - v_\theta p_y, v_y + v_\theta p_x, v_\theta)^\top\,. \tag{2.9}$$

To enforce a maximum translational velocity of a point p on the robot, we require

$$\|\mathbf{v}_p\| = \sqrt{(v_x - v_\theta p_y)^2 + (v_y + v_\theta p_x)^2} \leq v_{\max}\,. \tag{2.10}$$

We obtain the following constraint by substituting $\mathbf{v} = (v_x, v_y, v_\theta)^\top = \mathbf{s}'(u(t))\,\dot{u}(t)$ into Eq. (2.10):

$$\dot{u}(t) \leq \frac{v_{\max}}{\sqrt{(\mathbf{s}'_x - \mathbf{s}'_\theta p_y)^2 + (\mathbf{s}'_y + \mathbf{s}'_\theta p_x)^2}}\,, \tag{2.11}$$

where for readability $\mathbf{s}'$ is short for $\mathbf{s}'(u(t))$. For every discretized path point, the point of the robot contour that maximizes the denominator of Eq. (2.11) gives the strongest constraint on $\dot{u}(t)$. For a rectangular robot with width w and length l, we can exploit the symmetry to formulate a single constraint:

$$\dot{u}(t) \leq \frac{v_{\max}}{\sqrt{(|\mathbf{s}'_x| + |\mathbf{s}'_\theta| l/2)^2 + (|\mathbf{s}'_y| + |\mathbf{s}'_\theta| w/2)^2}}\,. \tag{2.12}$$

Wheel turn rate

In addition to limiting the resulting velocity of the robot, it can also be necessary to account for a maximum wheel turn rate to model friction or motor saturation effects that would prevent the accurate execution of feasible trajectories. A good example are differential drive robots that have limited wheel turn rates: The maximum achievable translational velocity depends on the radius of the driven curve.

Let $\psi_i(u)$ be the turn rate of wheel i computed from $\mathbf{s}'(u)$. To respect a maximum turn rate $\psi_{\max}$, we need to constrain $\dot{u}(t)$ by

$$\dot{u}(t) \leq \frac{\psi_{\max}}{\max\limits_{i \in \text{wheels}} \{\psi_i(u(t))\}}\,. \tag{2.13}$$

For an omnidirectional robot with omniWheels, one can compute the turn rates of the wheels $\psi_i(u)$ from the orientation $\mathbf{s}_\theta(u(t))$ and the path derivative $\mathbf{s}'(u(t))$. After rotating $\mathbf{s}'(u(t))$ by $-\mathbf{s}_\theta(u(t))$ into the robot frame, one can retrieve the turn rates by application of the equations of motion, see Eq. (2.16) together with Figure 2.11 and the explanation in Section 2.4 for the ones of the robot used in our experiments.

Centripetal acceleration

To prevent skidding and to protect sensitive payload, we can enforce a maximal centripetal acceleration a_c by

$$\dot{u}(t) \leq \sqrt{\frac{a_\text{c}}{\|\mathbf{s}'_{xy}(u(t))\| \cdot |\mathbf{s}'_\theta(u(t))|}}\,. \tag{2.14}$$

We enforce all of the above constraints at each discretized path point, i.e., we set the maximum $\dot{u}(t)$ that satisfies all constraints. This implicitly defines the maximum translational and rotational velocity at each support point (this is called the velocity limit curve in the literature). In a second step, we further decrease $\dot{u}(t)$ at the support points to also respect acceleration constraints.

Acceleration

We assume constantly accelerated motion of the robot between the discretized path points. To ensure safe transportation of sensitive payload, we limit the range of allowed acceleration and deceleration in the intervals between the path points. To determine the minimum and maximum velocities achievable by limited acceleration, we compute the arc length and orientation distance of the respective intervals by numerical integration.

We reduce the values for $\dot{u}(t)$ in two passes originating from beginning and end of the trajectory to ensure that they meet acceleration constraints between the supports points. The procedure is very similar to the time-scaling algorithm proposed by Bobrow et al. (1985) and Shin and McKay (1985). Given $\dot{u}(t)$, we can now compute the mapping from time to the internal parameter $u(t)$.

2.3.4 Optimization

We employ optimization to improve the initial trajectory with respect to a user defined cost function, e.g., time of travel. As shown in Figure 2.2, we initialize the optimization with the initial path (see, Section 2.3.2) and then iterates between two steps. First, we compute a velocity profile for the path described by the current parameters. Then, we use the inferred time of travel to compute the cost of the current trajectory. The change in cost then determines how to change the path parameters for the next iteration via magnitude-free gradient descent.

The optimization influences the shape of the trajectory through modification of the parameters of the compact path model, see Section 2.3.2. In particular, the effect of the optimization is the widening of the sharp turns of the initial path to a point at which the increased length of the path balances with the higher admissible velocities in curves with larger radius. The optimization also balances path length and distance to obstacles that impose velocity limits on the robot. Here, the optimization changes the location of inner waypoints in a coordinate system that is oriented along the gradient of the distance-transformed map. This means that the two degrees of freedom for moving the waypoint correspond to getting closer or farther away from the nearest obstacle and to an orthogonal movement with respect to it. Finally, the optimization also gradually changes the orientation behavior of the initial path to a simultaneous translation and rotation. Additionally, it reduces the overall robot rotation in the path, as long as this reduces the

Algorithm 1 Trajectory optimization with respect to the user-defined cost function c.

1: $\mathscr{T}_{\text{best}} \leftarrow$ initial trajectory
2: $\mathscr{P} \leftarrow$ parameters of initial trajectory
3: $\Delta_p \leftarrow \Delta_p^0 \quad \forall p \in \mathscr{P}$ // reset step size for all parameters
4: **repeat**
5: $\quad \Delta_c \leftarrow 0$ // track cost improvement
6: $\quad$ **for all** $p \in \mathscr{P}$ **do**
7: $\quad\quad \mathscr{T}_{\text{curr}} \leftarrow \mathscr{T}_{\text{best}}$
8: $\quad\quad \Delta_p \leftarrow \Delta_p^0$ // always reset step size
9: $\quad\quad i \leftarrow 0$ // track number of iterations
10: $\quad\quad$ **repeat**
11: $\quad\quad\quad i \leftarrow i + 1$
12: $\quad\quad\quad p \leftarrow p + \Delta_p$
13: $\quad\quad\quad \mathscr{T}_{\text{mod}} \leftarrow \text{modifyPathParameter}(\mathscr{T}_{\text{curr}}, p)$
14: $\quad\quad\quad \Delta_p \leftarrow \begin{cases} 1.2\Delta_p & c(\mathscr{T}_{\text{mod}}) < c(\mathscr{T}_{\text{curr}}), \\ -0.5\Delta_p & \text{else} \end{cases}$
15: $\quad\quad\quad \mathscr{T}_{\text{curr}} \leftarrow \mathscr{T}_{\text{mod}}$
16: $\quad\quad\quad$ **if** $c(\mathscr{T}_{\text{mod}}) < c(\mathscr{T}_{\text{best}})$ **then**
17: $\quad\quad\quad\quad \Delta_c \leftarrow \max\{\Delta_c, c(\mathscr{T}_{\text{best}}) - c(\mathscr{T}_{\text{mod}})\}$
18: $\quad\quad\quad\quad \mathscr{T}_{\text{best}} \leftarrow \mathscr{T}_{\text{mod}}$
19: $\quad\quad\quad\quad$ **break** // continue with the next parameter
20: $\quad\quad$ **until** $|\Delta_p| < \epsilon_p \vee i > i_{\max} \vee$ planning time is up
21: $\quad\quad$ **if** $|\Delta_p| < \epsilon_p \vee i > i_{\max}$ **then**
22: $\quad\quad\quad \Delta_p \leftarrow \Delta_p^0$ // variant: reset step size when stuck only
23: **until** $\Delta_c < \epsilon_c \vee$ planning time is up
24: **return** $\mathscr{T}_{\text{best}}$

travel time.

We propose an optimization procedure that is based on the update rule of RPROP (Riedmiller and Braun, 1993), which is a derivative-free optimization algorithm known for its robust convergence. Algorithm 1 shows the proposed procedure as pseudo-code.

In lines 1–3 we set as starting point the trajectory generated from the initial path and reset the step size Δ_p for all optimization parameters p. As long as planning time is left and non-negligible improvements occur for the cost of the trajectory (lines 4, 5, 23) we iterate through the list of tunable path model parameters. First, we modify the currently selected parameter p according to its current associated step size Δ_p and then update the trajectory accordingly (lines 12, 13). Then, we adapt the step size for the current parameter depending on whether or not the modified trajectory has a lower cost than the previous one (line 14). We do this according to the update rule of RPROP: If the change

of parameter p resulted in an improvement, we increase the step size by a factor of 1.2, otherwise, we halve the step size and invert the step direction. If at any point in this procedure we improve over the currently best trajectory, we record the cost improvement for the abortion criterion (line 17) and save the corresponding parameters (line 18). To avoid local minima, we also continue the optimization with the next parameter in the list as soon as we made an improvement (line 19). In the experiments, we compare versions with and without this continuing. We also abort the optimization of an individual parameter once its step size falls below a threshold or the number of iterations exceeds a threshold (line 20). In the experiments, we also compare several strategies regarding resetting the parameter step size Δ_p: always (default, see line 8), never (remove line 8), or resetting it only when the optimization is stuck (remove line 8, add lines 21, 22).

Figure 2.10 shows the progress of the optimization for an exemplary trajectory. To achieve faster travel times the optimization increases the magnitude of the tangents to widen the curves. It furthermore balances the path length and the distance to obstacles and changes the overall orientation behavior along the path.

During computation of the velocity profile we also check for collisions of the robot with the environment and assign infinite cost to colliding trajectories. At each iteration our method returns feasible trajectories with decreasing cost. As obstacles that cause collisions repel the optimization via infinite cost, the optimization cannot "jump" over obstacles and is therefore confined to the route of the initial path through the obstacles. To consider also homotopically different solutions one can seed parallel optimizations with homotopically different initial paths as proposed by Kuderer et al. (2014).

When planning a trajectory for a robot that is already in motion, the velocities of the robot at the start of the trajectory are non-zero. In this context, it can occur that the shape of the initial trajectory prevents a transition with continuous velocities, e.g., due to a sharp curve near its beginning that enforces a low robot velocity. In such situations, we change the cost function of the optimization to minimize the amount of velocity discontinuity between desired and maximum achievable velocities at the start of the new trajectory. In this way, the optimization adapts the shape of the trajectory to allow a smooth fit. As soon as it found parameters that match the start velocities, the optimization switches back to the original cost function for the remaining planning time. As the optimization can typically fix the mentioned discontinuities by increasing the magnitude of the first derivative of the start waypoint, we do not skip to the next parameter upon improvement in this case.

2.4 Trajectory Execution

As shown in the lower part of Figure 2.2, we send the final optimized trajectory to an error feedback controller for execution. Before execution, we transform the trajectory from

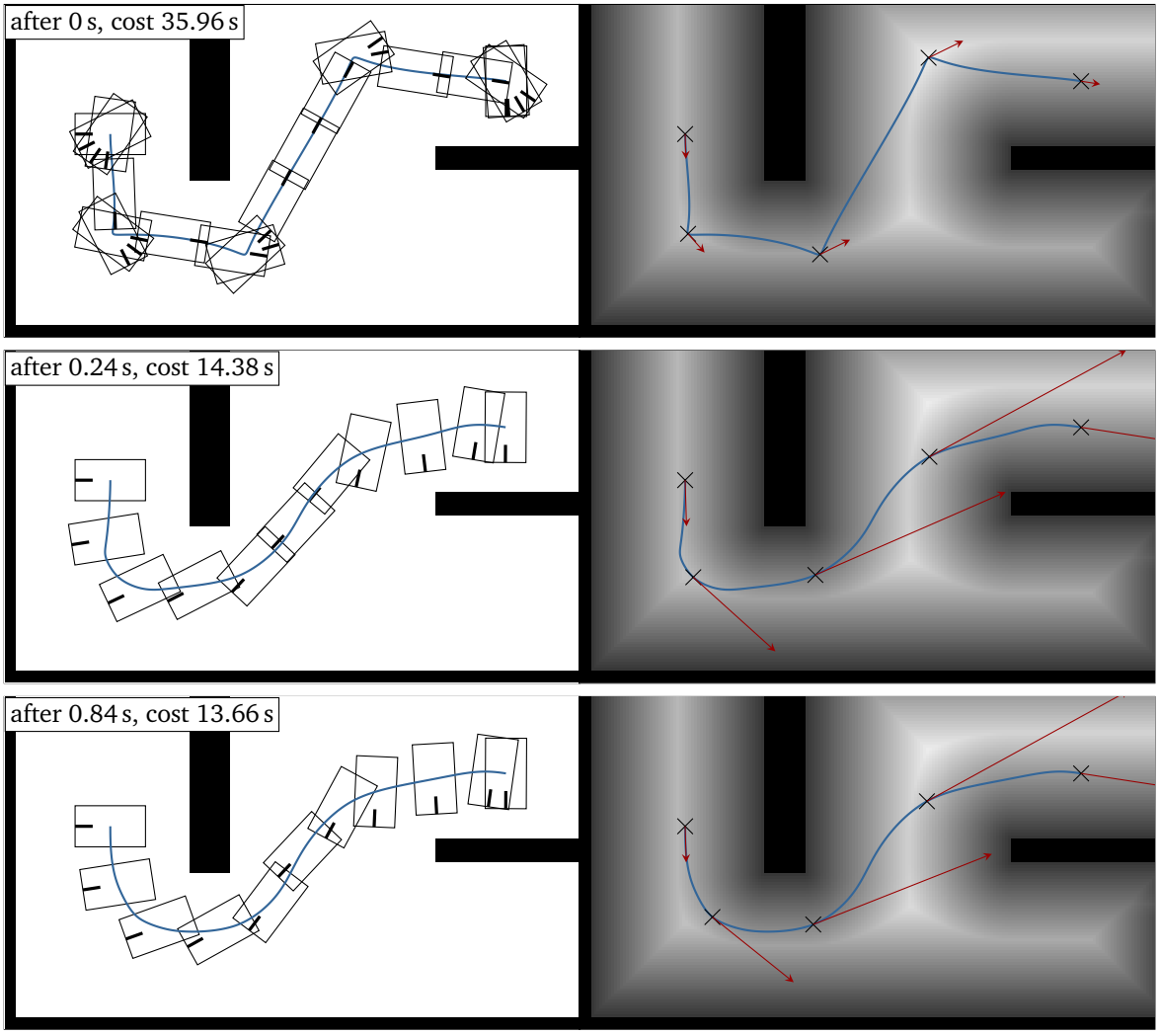


Figure 2.10: Progress of the trajectory optimization over time. The optimization balances path length, distance to obstacles, orientation and trajectory shape for faster traversal. The left part of the figure shows the current trajectory with overlays of the robot contour in the grid map of the environment. The right part shows the trajectory with waypoints $\mathbf{w}_i$ (crosses) and the first derivatives of the path (arrows) on a distance transform of the environment. The figure shows obstacles in black and darker cells are closer to obstacles.

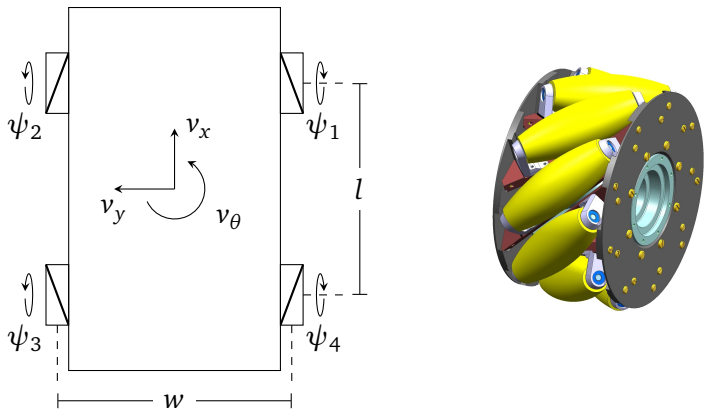


Figure 2.11: Schematic drawing of the robot used in our experiments. The robot is driven by four Mecanum wheels like the one shown on the right. Their passive rollers are oriented at $\pm\frac{\pi}{4}$, diagonal lines show the respective orientation on the floor in the drawing on the left.

global map coordinates into the odometry frame of the robot. The error feedback controller then executes the trajectory solely based on odometry information, decoupled from global trajectory planning and global localization. Thereby, the error feedback controller can run at a higher frequency than the trajectory planning. We account for accumulating odometry drift and changes in the environment by planning updated trajectories while the robot executes the trajectory.

The inputs for the controller are the planned position $\mathcal{T}(t)$ and velocities $\dot{\mathcal{T}}(t)$ for the current time t as determined by the trajectory $\mathcal{T}$ as well as the current position $\mathbf{x}(t)$ in the odometry frame as measured by the wheel encoders of the robot. The output of the controller is a velocity $\mathbf{v} = (v_x, v_y, v_\theta)^\top$ that we send as a command to the robot. The controller computes this command velocity as follows:

$$\mathbf{v} = \dot{\mathcal{T}}(t + \Delta t_{\text{del}}) + \text{diag}(g_x, g_y, g_\theta)(\mathcal{T}(t) - \mathbf{x}(t)) \,, \tag{2.15}$$

where g_x, g_y and g_θ are translational and rotational gains. The control law consists of two summands, the first summand is the feedforward part and the second summand is the error feedback. The feedforward part consists of the velocities as planned in the trajectory. We retrieve them for a point in time that is shifted by Δt_{del} into the future. This accounts for the time delay that occurs before the robot is actually executing the command.

To execute the commanded velocity, the robot needs to transform it into wheel turn rates. One can retrieve them through the equations of motion for the particular robot used. For the omnidirectional robot used in our experiments, the equations of motions

are (Muir, 1988, Eq. (6.2.13))

$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & 1 & 0.5\,(w+l) \\ 1 & -1 & -0.5\,(w+l) \\ 1 & 1 & -0.5\,(w+l) \\ 1 & -1 & 0.5\,(w+l) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_\theta \end{pmatrix}, \tag{2.16}$$

where ψ_i are the wheel turn rates, r is the wheel radius and w and l are track and wheelbase, respectively. Figure 2.11 shows a schematic drawing that introduces the used quantities. The above equation is also used when computing constraints regarding the maximum turn rate of wheels, see Section 2.3.3.

2.5 Experiments

We evaluate the core components responsible for robot motion in our navigation system: the trajectory generation module and the trajectory execution module. We start by evaluating the influence of the initial path planner on the proposed method. Then, we analyze our trajectory optimization by comparing the results of several variants on a set of navigation tasks. On these same set of tasks we also compare the trajectories of our system with the ones generated by an approach based on the RRT* path planning algorithm. We present real-world experiments for evaluating the trajectory execution module and examine the influence of different constraint settings on the system behavior. Then, we present an evaluation of the system's capability to follow the planned trajectories in a changing environment. At the end of this section, we report on real-world robotic applications of our navigation system.

We conducted our experiments with the KUKA omniRob platform shown in Figure 2.12. The omnidirectional platform has dimensions of ca. $0.7\,\mathrm{m} \times 1.2\,\mathrm{m}$ and perceives its environment with two SICK safety laser range finders mounted in a way that yields a 360 degree view. We performed all computations on a consumer grade notebook that we mounted on the platform.

When performing trajectory planning in real-world changing environments, it is not economical to spend computational resources on optimizing trajectories all the way to the goal pose for longer navigation tasks. The later parts of a longer trajectory are outside the sensors' field of view and the environment is likely to change until the robot arrives there, requiring to update the trajectory, discarding the initial results. We also require regular updating of planned trajectories to cope with localization errors and accumulating odometry drift. In our experiments we found a good trade-off by using the first four waypoints of the initial path for trajectory planning and triggering replanning of updated trajectories every 0.6 s, which is the time allotted for path planning and trajectory optimization.

Figure 2.12: The omnidirectional mobile manipulator omniRob by KUKA that we used in our experiments.

2.5.1 Influence of the choice of geometric path planner

Our system uses an initial path generated by a geometric global path planner to generate an initial trajectory as initialization for the optimization. Naturally, the choice of planning algorithm can have an influence on the final result.

2D path planners ignore orientation and therefore require the circumcircle of the robot to be obstacle-free along the path, see for example Figure 2.13 (A). Waypoint planners that account for the orientation and shape of the robot during forward motion are able to find paths through passages as shown in Figure 2.13 (B). Planners operating in the full holonomic configuration space of the robot can also find paths through passages that require lateral or diagonal movements as in Figure 2.13 (C). Our approach can generate admissible spatio-temporal trajectories for all these situations, given a suitable waypoint planner.

To assess the robustness of our system against the choice of waypoint planner, we compare the optimization results for the CARMEN 2D value iteration planner (CARMEN, 2006), an A*-based configuration space planner, and our hybrid approach that uses Voronoi diagrams where possible and resorts to configuration space planning when necessary (see Section 2.3.1). We selected six start/goal poses on a map of a factory floor as shown in Figure 2.14 and let a simulated robot perform travel tasks for all of the 30 start-goal combinations.

Table 2.1 shows the average times needed for planning (t_{plan}), optimization (t_{opt}), and

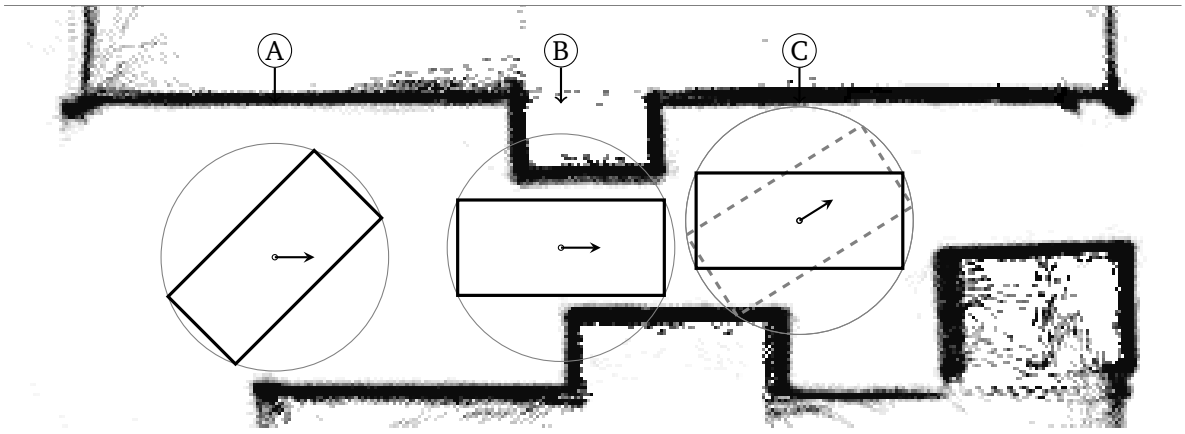


Figure 2.13: Omnidirectional platforms are versatile in narrow spaces. They can rotate freely in open spaces (A), move with constant orientation in narrow passages (B), and execute maneuvers (C) where nonholonomic trajectories would lead to collisions (dashed). For readability, the figure contains the circumcircle of the rectangular robot.

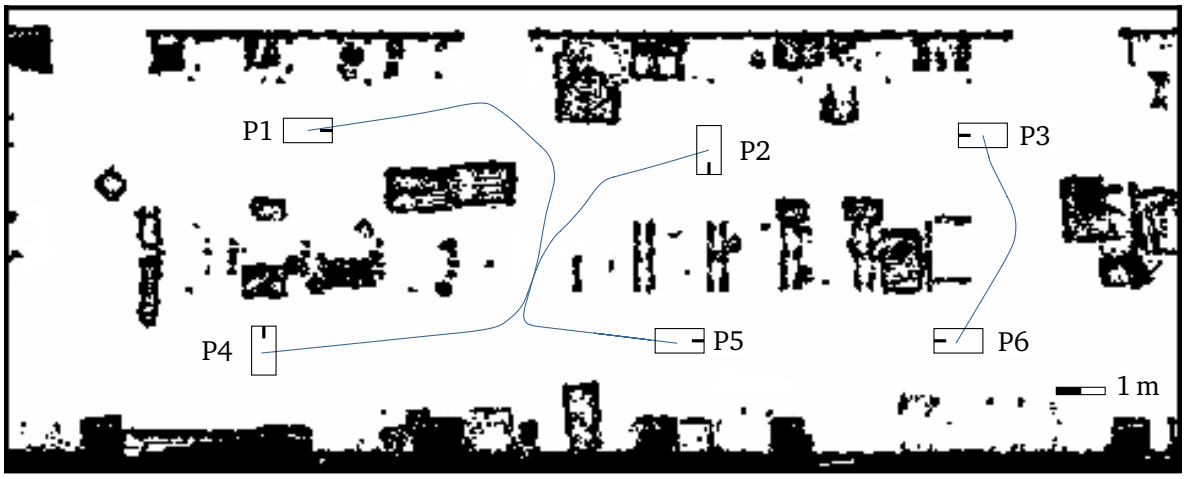


Figure 2.14: Map with start/goal poses used in our evaluation of different path planners in combination with our trajectory generation method. The figure also contains selected example trajectories: P1→P5, P2→P4 and P3→P6.

Table 2.1: Numerical results of global path planner comparison.

Planner	t_{plan}	t_{opt}	$t_{\text{travel-I}}$	$t_{\text{travel-O}}$
2D value iteration	0.022 s	0.55 s	82.37 s	38.11 s
Voronoi	0.005 s	0.46 s	36.28 s	28.46 s
Configuration space	4.864 s	0.43 s	31.49 s	27.19 s

Figure 2.15: The grid map used for our trajectory optimization experiments, obstacles are shown in black. The map also shows the robot contour at the nine start/goal poses that we use to generate 72 navigation tasks.

execution of the initial and optimized trajectories ($t_{\text{travel-I}}$ and $t_{\text{travel-O}}$, respectively). As expected, the configuration space planner requires substantially more time to find an initial path compared with the others, which prevents its use in most practical applications. On the other hand, it generates waypoints that fully exploit the omnidirectional capabilities of the robot which leads to trajectories with the shortest initial and optimized travel times. The hybrid planner uses Voronoi paths which results in substantially different waypoints. Nevertheless, the optimization results are comparable, but obtained in a much shorter planning time. In contrast, the 2D value iteration plans with the circumcircle of the robot. This can cause detours for narrow passages and the waypoints of this planner have to maintain a larger distance to obstacles in general, which also causes longer initial paths. However, our optimization can compensate this drawback to a large degree by adjusting the trajectory shape, which causes a substantial decrease in execution time as shown in Table 2.1.

In summary, our method is robust to the choice of initial planner and the Voronoi-based planner offers fast computation of initial paths that account for the orientation of the robot.

2.5.2 Trajectory optimization

In this experiment, we evaluate the convergence of the trajectory optimization and analyze the effect of different choices in the optimization algorithm. Figure 2.15 shows the grid map together with nine start/goal poses that we used to generate the 72 navigation tasks for this experiment. For each start/goal combination we computed an initial trajectory and optimized it for 1.5 s using the method proposed in Section 2.3.4. To aggregate data over the different tasks, we normalize the trajectory cost during optimization by dividing it by the initial cost before optimization. The plot in Figure 2.16 shows the average over all 72 tasks for different variants of the optimization algorithm. In general, there are bigger gains in the beginning of the optimization and the convergence rate lowers over time until it reaches a gain of 30% to 35%. The standard deviation lies within 12% for all data points and is not shown in the plot for readability. The high variance is caused by the different potential for optimization in the navigation tasks: For some start/goal combinations the initial trajectory is already close to the optimal solution, e.g., for the tasks that correspond to driving a straight line in the middle of a corridor.

In Figure 2.16 we compare the performance of our algorithm against RPROP (Riedmiller and Braun, 1993) and against variants of our method in Algorithm 1 with different behavior regarding switching to the next parameter upon an improvement (line 19) and resetting the step length for parameters (lines 8, 21, 22). In contrast to our algorithm, RPROP works on all parameters simultaneously and needs an extra step to compute the partial derivative for each parameter. After a slightly better convergence in the first few steps, it converges at a slightly lower rate to an about 5% worse improvement over the initial trajectories than the proposed method. Since one iteration also takes longer for RPROP than for our algorithm, it creates feasible solutions at a lower frequency (but with bigger improvements between them). This is a disadvantage when trying to completely exploit an allotted time window for optimization, in the plot we observe that it overshot the planning time by 0.1 s before creating the next feasible solution. The variant of the proposed method that performs no reset of the parameter step length Δ_p (lines 8, 21, 22 not present) converges to a similar improvement as RPROP, 5% worse than the variants that reset the step lengths. After 0.1 s it shows for a short time a better convergence than RPROP and the variants that reset the step length. Then, this advantage fades and after about 1 s it aborts optimization as it could not find any further improvements. The variants that reset the step size in every round (line 8 present) and only when stuck (lines 21, 22 present, line 8 not present) show comparable performance with a slight advantage for the variant that only resets when stuck. As the plot shows, convergence behavior greatly benefits from switching to the next parameter upon improvement (line 19): The variant without this converges to a comparable improvement, but at a slower rate.

In summary, our optimization procedure results in high improvements on the cost function in the first few iterations. Furthermore, we observe a beneficial contribution from

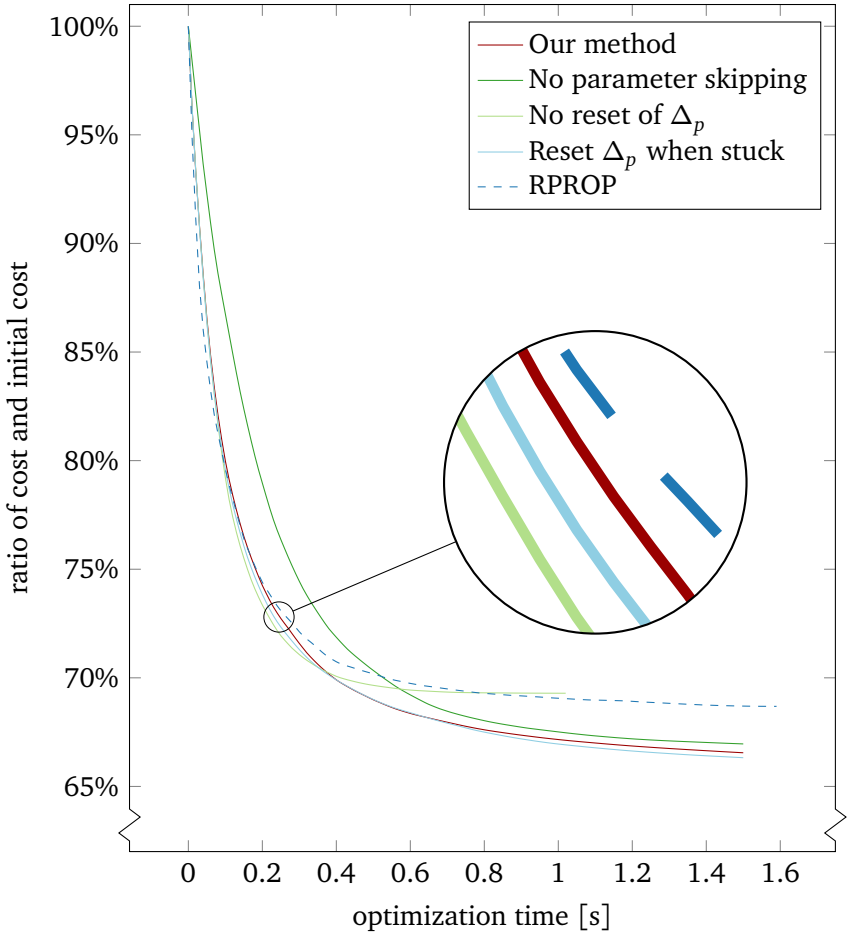


Figure 2.16: The progress of trajectory optimization over time, averaged over the 72 navigation tasks defined in Figure 2.15. The plot compares different variants of our proposed method from Algorithm 1 (see text) and RPROP. To average over different navigation tasks with different potential for optimization, the plot shows the ratio between current trajectory cost and cost of the initial trajectory over time.

continuing with the next parameter when finding an improvement and from resetting the parameter step size in our optimization algorithm. We also observe an improvement over direct application of the RPROP algorithm.

2.5.3 Comparison of trajectory generation

The goal of our proposed method is to provide smooth, curvature continuous and time-optimized robot motion that accounts for kinematic and dynamic constraints and limits the speed of the robot in the vicinity of obstacles. In this experiment, we compare our method with trajectory generation based on the widely spread motion planning framework

Open Motion Planning Library (OMPL) by Şucan et al. (2012).

In contrast to our specialized method, the planners provided in OMPL are very flexible and also applicable to high-dimensional search spaces and robotic arms. To generate smooth, curvature continuous trajectories with these planners one needs to resort to sampling in acceleration control space. However, the planners available for these control spaces do not optimize an objective, i.e., they stop after finding a path to the goal. Optimizing planners are available for the geometric domain, for example the RRT* planner proposed by Karaman and Frazzoli (2011) that converges to shortest paths consisting of linear connections in the workspace, SE(2) in our case. To generate feasible trajectories from the resulting paths, we compute a velocity profile as for our approach (see Section 2.3.3). Due to the curvature discontinuities at the join points of the linear segments, this profile decelerates the robot to a brief stop at these points.

For this experiment we also resort to the 72 navigation tasks given by the start/goal poses in Figure 2.15. We gave both methods 1.5 s of total planning time. Figure 2.17 shows a selection of the resulting trajectories for both approaches. As the figure shows, the path generated by RRT* is non-smooth at the join points of linear segments and it keeps little distance to obstacles (as RRT* optimizes for path length). In comparison, the trajectories generated by our method are smooth and curvature continuous and keep a safe distance to nearby obstacles.

The statistics shown in Figure 2.18 provide a quantitative evaluation of the differences between the two approaches. The plots show the distribution of ratios and differences of path length and travel time between the trajectories generated by RRT* and the proposed method for the 72 navigation tasks. For the most part, the paths computed by RRT* are shorter than the trajectories returned by our method Figure 2.18 (top). This is a direct consequence of the different objectives of the two methods: While RRT* optimizes for path length, our method aims to reduce travel time while respecting a set of kinematic and dynamic constraints as well as obstacle imposed speed limits. Together with the stops at join points of linear segments this explains the higher travel time for the trajectories generated from the RRT* paths: Figure 2.18 (bottom) shows that the proposed method consistently generates trajectories that are 50% faster than the ones returned by computing a velocity profile for the paths generated by RRT*. One-tailed paired t-tests on the differences (Figure 2.18 (right)) show that the paths returned by our method are significantly longer than the RRT*-based paths ($P = 2 \cdot 10^{-8}$) but that at the same time our trajectories feature a significantly shorter travel time ($P = 2.2 \cdot 10^{-19}$).

In summary, the comparison with the RRT*-based trajectory generation highlights the importance of accounting for smooth paths with velocity profiles and distance to obstacles. Our solution accounts for these aspects while optimizing travel time.

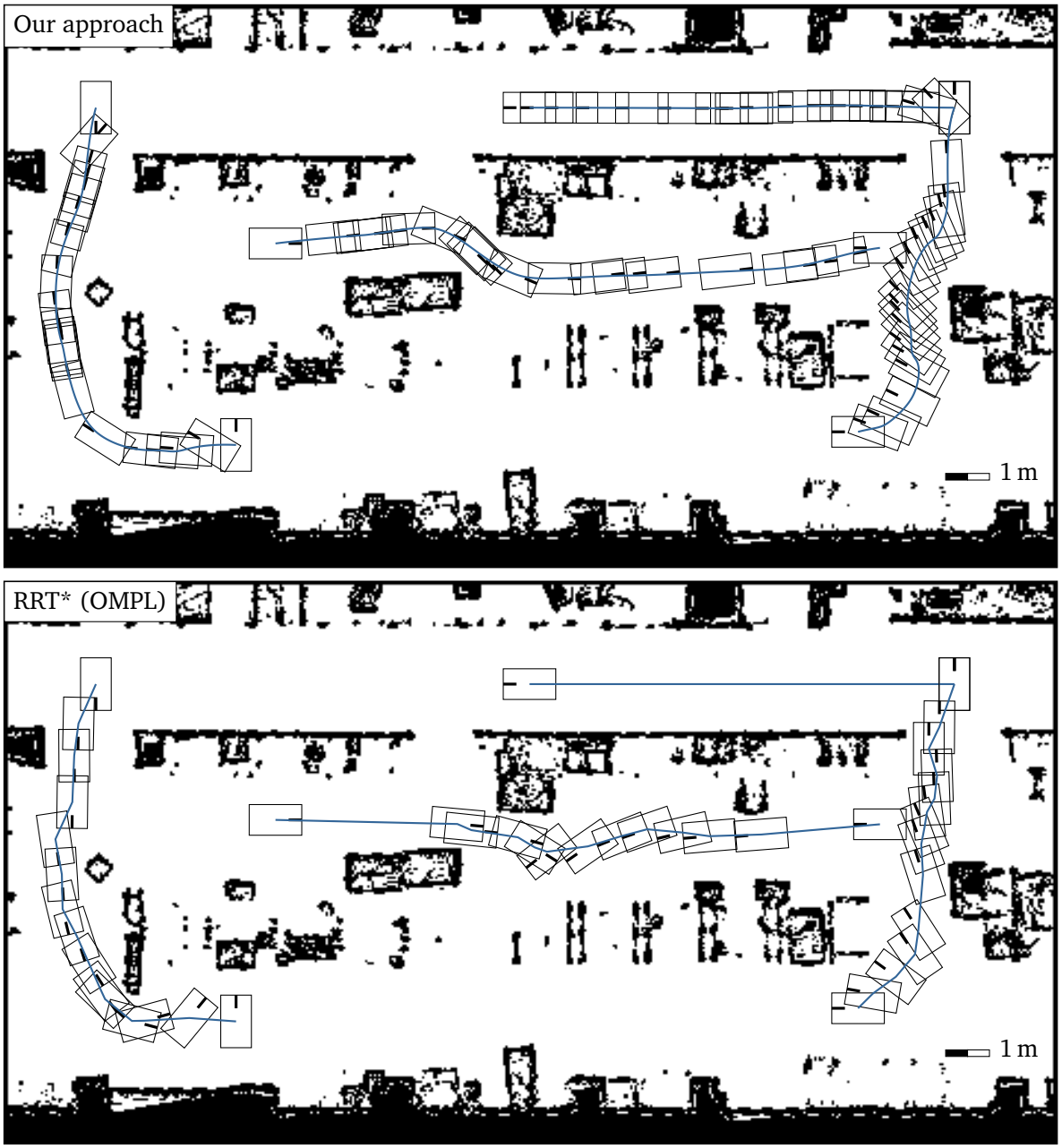


Figure 2.17: Comparison of trajectory generation by our approach (top) and a procedure based on the implementation of RRT* (Karaman and Frazzoli, 2011) in OMPL (Şucan et al., 2012) (bottom). The figure shows the results for a selection of the 72 tasks defined by the start/goal poses shown in Figure 2.15. The robot center is shown as blue solid line with an overlay of the robot contour along the trajectory.

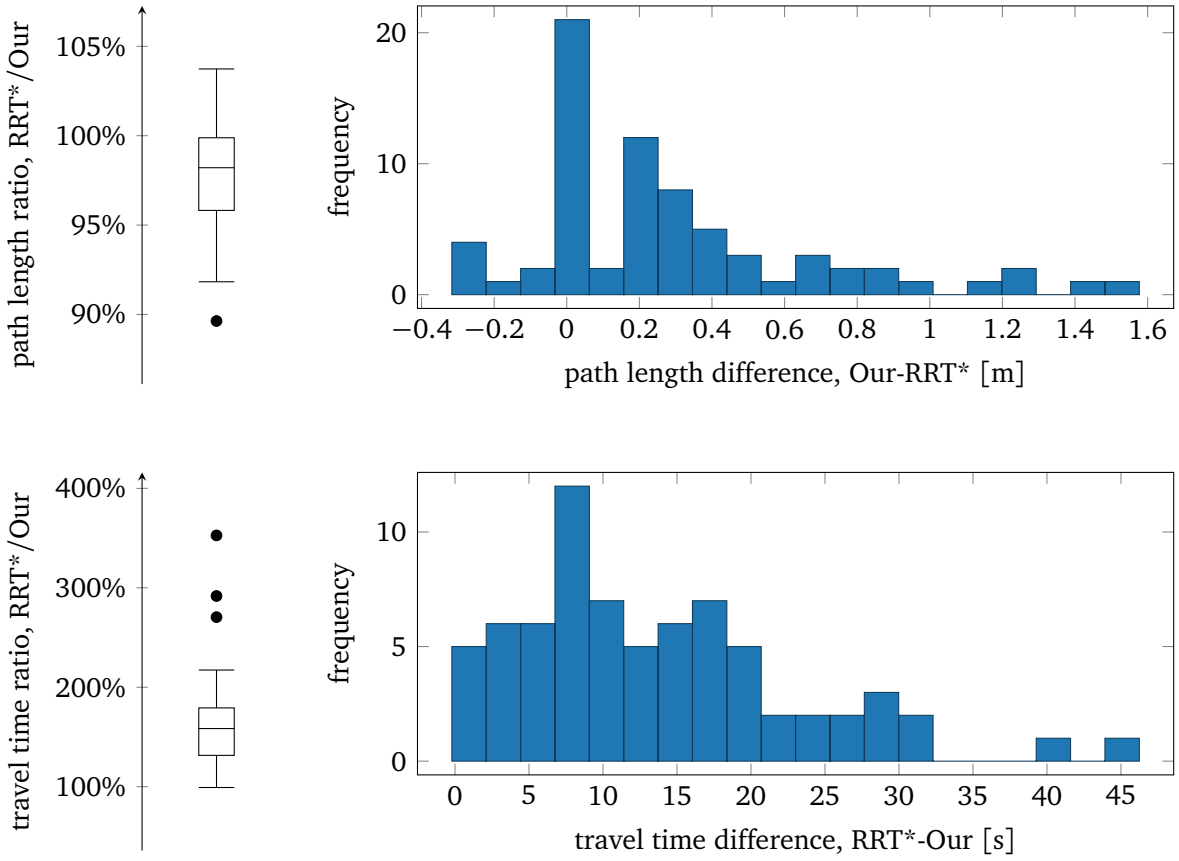


Figure 2.18: Comparison of trajectory generation by our approach and a procedure based on the implementation of RRT* (Karaman and Frazzoli, 2011) in OMPL (Şucan et al., 2012). The plots visualize the differences in path length (top) and travel time (bottom) of RRT* and our approach for the 72 navigation tasks in Figure 2.15. *Left:* The boxplots show ratios of path length and travel time, indicating lower and upper quartile (box) along with the median (line). The whiskers indicate data within 1.5 times the interquartile range (IQR) of lower and upper quartile, respectively. Dots mark data outside this range. *Right:* The plots show histograms of the differences in path length and travel time. See Figure 2.17 for examples for the generated trajectories.

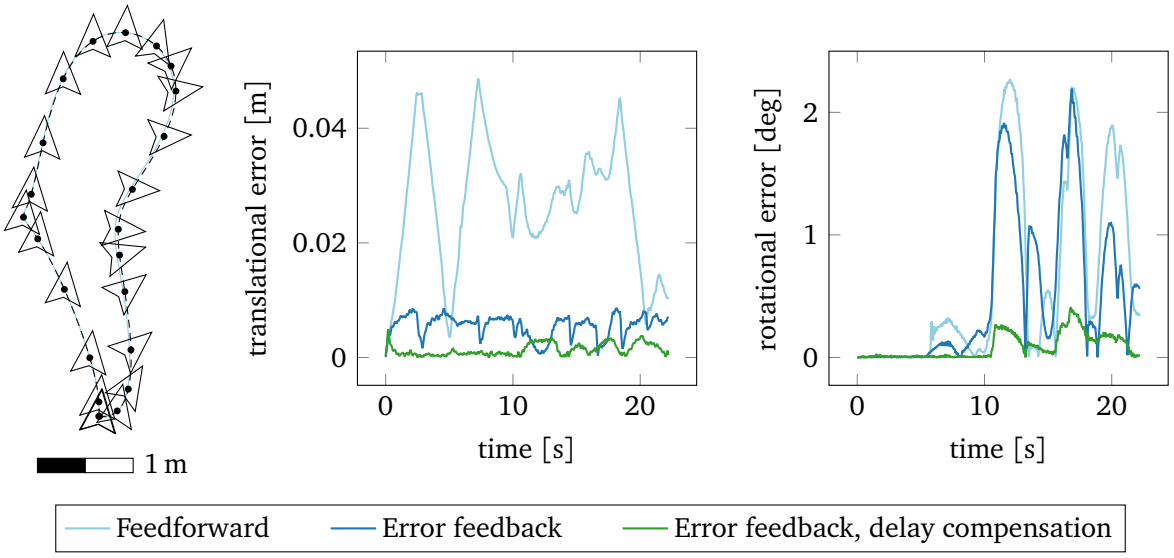


Figure 2.19: Execution of a reference trajectory by a real robot. The left part shows the reference trajectory (dashed), indicating the orientation of the robot with darts drawn every second. The left part also shows a feedforward execution (solid). The middle and right plot show the translational tracking error $\|\mathcal{T}(t) - \mathbf{x}(t)\|$ and the rotational tracking error $|\mathcal{T}_\theta(t) - x_\theta(t)|$ over time, respectively. The plots show the errors for feedforward execution , error feedbackand for error feedback with delay compensation.

2.5.4 Open-loop and closed-loop trajectory execution

In this experiment, we evaluate the suitability of our trajectory representation on the basis of tracking errors for executing a reference trajectory on a real robot. We perform this experiment with and without error feedback and delay compensation in the controller (see Section 2.4, Eq. (2.15)). More specifically, we once set $\Delta t_{\text{del}} = 0\,\text{s}$, $g_x = 0$, $g_y = 0$, $g_\theta = 0$ (feedforward), once $\Delta t_{\text{del}} = 0\,\text{s}$, $g_x = 3$, $g_y = 3$, $g_\theta = 0.3$ (error feedback) and once $\Delta t_{\text{del}} = 0.06\,\text{s}$, $g_x = 3$, $g_y = 3$, $g_\theta = 0.3$ (error feedback and delay compensation).

Figure 2.19 (left) shows the reference trajectory which specifies a loop in an area of $10\,\text{m}^2$ with roughly 22 s of travel time and exploits the holonomic capabilities of our platform. The figure indicates the orientation of the robot with darts and shows the driven path for a feedforward execution as solid line. Figure 2.19 also shows plots of the translational (middle) and rotational (right) deviation of the actual robot pose from the planned pose for every point in time, as measured by the odometry of the robot.

When executing the reference trajectory with feedforward commands only, the system relies on the appropriate modeling of the trajectory and its velocity profile. The average errors for this execution are 0.027 m with a standard deviation of 0.01 m and 0.6 deg with a standard deviation of 0.7 deg and never exceed 0.05 m and 3 deg. We attribute

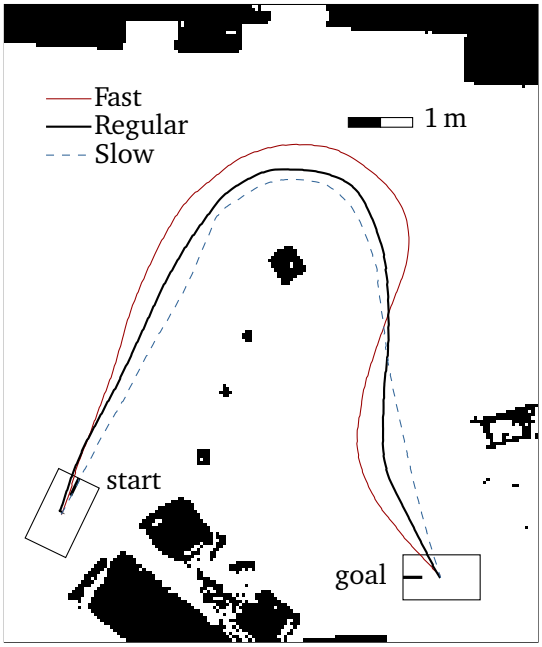


Figure 2.20: Setup for an experiment to resemble a medium range transportation task. The figure shows the path of the robot in three exemplary executions with different limits for speed and acceleration in the computation of velocity profiles. With higher admissible speeds the system chooses longer paths that it can traverse faster while staying within obstacle imposed speed limits and constraints on centripetal acceleration.

these errors to execution inaccuracies and approximation errors introduced by sampling the trajectory with discrete commands sent to the robot. The run with error feedback shows how the controller can compensate these errors. The run with delay compensation illustrates the contribution of this measure to tracking performance. Accounting for delay with the parameter Δt_{del} reduces the average tracking error from 0.0056 m and 0.52 deg to 0.0014 m and 0.09 deg. This effect was even stronger in an earlier version of the omniRob, where we had to account for roughly double the amount of delay in the system.

In summary, the low tracking errors achieved show that the compact path model and velocity profiles used by our approach generate trajectories that the robot can accurately follow.

2.5.5 Influence of trajectory constraints

This experiment shows how our system adapts to different values for the constraints in velocity profile generation, see Section 2.3.3. We show the experimental setup that is inspired by a medium range transportation task in Figure 2.20. The figure shows

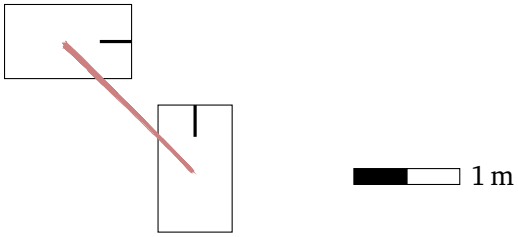


Figure 2.21: Experimental setup to resemble a repetitive pick&place task. The figure shows the start/goal poses and an overlay of the 19 trajectories the robot drove in two minutes.

the driven paths for three exemplary executions of the task with different settings for maximum allowed velocity and acceleration. Lower speed limits lead the system to choose shorter paths that are closer to obstacles. A reduction of the speed limits leads to a reduction of the area in which obstacle speed limits are the dominating constraint and in turn a reduction in path length leads to shorter travel times. For higher speed limits, the optimization chooses longer paths that feature higher clearance from obstacles and wider curves. The higher possible velocities at greater distance from obstacles and in wider curves make up for the increased length of the path.

2.5.6 Trajectory execution accuracy

In these experiments we quantify the performance of our system in executing the planned trajectories. We evaluate the system on the transportation task introduced above (see Figure 2.20) as well as on the task shown in Figure 2.21: Here, the robot has to repeatedly travel between two close-by positions while changing orientation. We designed this task to resemble a repetitive pick&place application for a mobile manipulator. For both tasks we conducted experiments with regular velocity limits and with higher speed limits. We executed the transportation task ten times, yielding an average travel time of 35.4 s with a standard deviation of 0.2 s. Five executions of the variant with raised velocity limits led to a reduced average travel time of 28.4 s again with a standard deviation of 0.2 s. We executed the pick&place task for two minutes yielding 19 very similar trajectories that Figure 2.21 shows as overlay. In a run with higher admissible velocities and accelerations, the robot achieved 30 iterations in the same time interval.

We also evaluated the system in a task that consisted of navigation in a dynamically changing environment. Figure 2.22 shows a grid map of the environment together with the eight goal configurations between which the robot traveled repeatedly for two hours and 2.9 km. While the robot was traveling, the system had to cope with people blocking its path and changing arrangements of objects in the environment. We collected the data for this experiment when evaluating the navigation benchmark that we present in detail

Figure 2.22: Our experiment for navigation in a dynamic environment. The robot looped for 2.9 km and two hours between eight goal configurations while object rearrangements and moving people changed the environment. The figure shows roughly half of the driven paths, deviations from the typically chosen route are due to dynamic obstacles blocking the usual path of the robot.

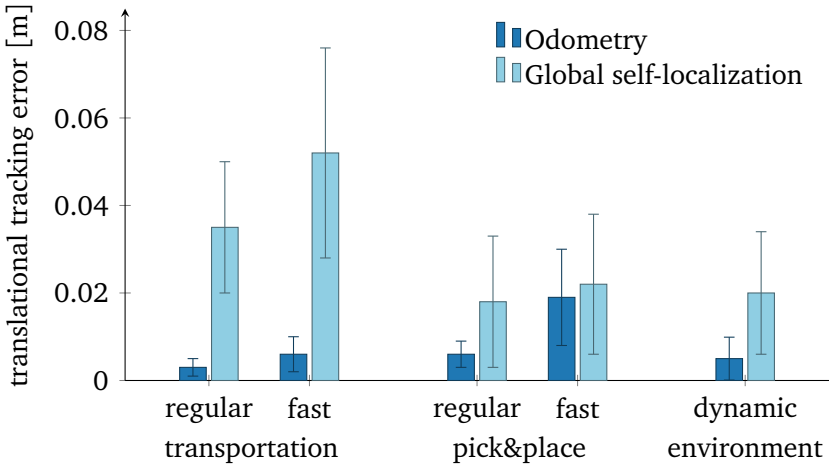


Figure 2.23: Mean translational tracking error and standard deviation for the transportation task shown in Figure 2.20, the pick&place task shown in Figure 2.21, and the navigation in a dynamic environment shown in Figure 2.22. The plot shows the error computed in the odometry frame of the robot as well as with respect to global self-localization.

Figure 2.24: The omniRob navigating with the proposed trajectory generation approach in an industrial mobile manipulation demonstration at the KUKA booth at the AUTOMATICA 2010 fair.

in Chapter 6 of this thesis.

Figure 2.23 shows the translational tracking errors for all tasks. The plot shows the error computed within the odometry frame of the robot as well as with respect to the global self-localization based on a grid map and the two laser range finders of the platform (see Section 2.2). The errors are the deviation of the robot from its planned pose, averaged over time. When we measure the deviation based on the odometry of the robot, the average errors are below 0.02 m. The average errors according to the global self-localization are below the grid map resolution of 0.05 m.

In summary, our approach shows high accuracy when executing the generated trajectories over extended periods of time and in changing environments.

2.5.7 Real-world deployed applications

We have extensively tested and deployed the navigation system proposed in this chapter. In one application scenario the omniRob had to perform random navigation tasks between three goal locations and never failed to reach the goal in a 3.5 hour trial. This trial was a combined evaluation with a system for high accuracy localization and docking that we present in the next chapter of this thesis. The proposed navigation system was also employed in several mobile manipulation demonstrations in the industrial context. The system has been shown on the omniRob at the AUTOMATICA Trade Fair for Automation and Mechatronics in 2010 and 2014 (see Figure 2.24) and at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in 2011. The KUKA Moiros platform, a scaled-up version of the omniRob, used the navigation system in its demonstration at the Hannover Messe 2013, where it won the Robotics Award (see Figure 2.25). Additionally, the navigation system has been deployed on the KUKA youBot, a scaled-down version of the omniRob, and demonstrated at the International Conference on Robotics and Automation (ICRA) in 2014.

Figure 2.25: The KUKA Moiros robot using our omnidirectional navigation system during a live demonstration at the Hannover Messe 2013 fair.

In addition to the omniRob, the Moiros and the youBot platform, we successfully tested the navigation system on the KUKA omniMove, a large-scale omnidirectional platform and the Willow Garage PR2 robot. Currently, KUKA Roboter GmbH is shipping a version of the proposed system as prototypical navigation software for the omniRob mobile manipulator.

2.6 Related Work

In this chapter, we presented a navigation system for omnidirectional robots. We first discuss related work from the areas navigation systems, industrial applications and omnidirectional robots. Then, we discuss works that are related to our approach to trajectory representation and generation. This covers the areas of spline-based trajectory representations, planning with velocities, and deforming and optimizing paths and trajectories.

Navigation systems A successful navigation system for mobile robots from academia is the open source framework CARMEN (2006) initiated by Montemerlo et al. (2003). It localizes the robot in a grid map of the environment with the help of range sensors and MCL as proposed by Thrun et al. (2001). CARMEN generates velocity commands for the robot by either following a global gradient field computed on the grid map or through waypoint following algorithms applied to a sequence of cells on the grid map.

The ROS navigation stack for the PR2 is a more recent system for which Marder-Eppstein et al. (2010) demonstrated robust performance. The system considers robot motion in the plane but performs collision checks in 3D. It generates paths by an A* search on a costmap of the environment and uses the Dynamic Window Approach by Fox et al. (1997) to generate velocity commands that follow the path to the goal. The system employs the adaptive Monte Carlo localization by Fox (2003) which we also use in the system proposed in this chapter.

Kümmerle et al. (2014) present a navigation system for a differential drive robot in pedestrian zones that also performs traversability analysis of the environment. The system employs MCL on a grid map that it generates with a graph-based formulation of the SLAM problem also employed in the proposed system. Since their application scenario is of larger scale, Kümmerle et al. propose to use tiled maps in their system. They realize travel

to a goal configuration with a three-tiered approach. On the highest level a topology planner operates on the map tiles while on the middle level a planner based on Dijkstra's algorithm generates waypoints within the map tiles. These waypoints serve as subgoals for a local planner that operates on a state lattice that includes the velocity of the robot in the plan (Rufli et al., 2009).

The DARPA Grand and Urban Challenge sparked a considerable amount of research for navigation systems for autonomous cars (Thrun et al., 2006; Likhachev and Ferguson, 2009; Ziegler et al., 2008; Ziegler and Stiller, 2009; Werling and Gröll, 2008; Maček et al., 2009). The systems developed for these challenges are tuned specifically to car-like vehicles driving within a given road network or maneuvering on a parking lot. Thrun et al. (2006) present the system that won the Grand Challenge. They propose a separate velocity and steering controller to keep the robot at a lateral offset from the previously smoothed road description. The velocity controller slows the car down for lateral changes or when hitting bumps. Also the system for the Grand Challenge proposed by Ziegler et al. (2008) and Werling and Gröll (2008) relies on separate controllers for velocity and heading to track a geometric path. Likhachev and Ferguson (2009) present the system that won the Urban Challenge. The system generates curvature discontinuous geometric paths with Anytime Dynamic A* on a multi-resolution lattice with 32 discrete orientations. It tracks the path with a local planner and reduces the maximum speed for higher curvature sections to cope with the curvature discontinuities in practice.

While all of the above approaches determine velocities with controllers during execution, the systems for the Urban Challenge presented by Ziegler and Stiller (2009) and Maček et al. (2009) plan for velocities. Maček et al. (2009) rely on partial motion planning that grows a tree of suitable velocity commands within a horizon and Ziegler and Stiller (2009) use quintic splines to represent lane changing maneuvers in a spatio-temporal state lattice. Similar to Ziegler and Stiller we plan for velocities on paths based on quintic splines, however we consider an omnidirectional robot that is not restricted to lane changes.

Industrial navigation In the industrial context there are a variety of navigation systems for mobile robots available on the market today. The well-known approach by KIVA Systems consists of a fleet of mobile robots that fetch shelves to human workers to let them perform the necessary pick operations in warehouse automation. The robots navigate on a grid with the help of optical markers on the warehouse floor (Wurman et al., 2008; Guizzo, 2008). Frog AGV Systems (2015) offers robots that rely on a grid pattern of magnets embedded in the shop floor for navigation. There are also systems that do not rely on external infrastructure for navigation. The TransCar system sold by swisslog (2015) is capable of laser-guided navigation on predefined routes in the context of hospital logistics. The BlueBotics ANT navigation system described by Tomatis (2011) localizes itself with a Kalman Filter by matching previously mapped features that appear in laser scans of the environment. The system follows paths computed on a graph representation

of the environment with the help of the Dynamic Window Approach by Fox et al. (1997). The ARNL navigation stack by adept mobilerobots (2015) also uses this approach and computes the followed paths on a grid map built with range sensors. Our system, instead, computes smooth trajectories that plan for velocities. Therefore, our system can execute planned motions with high accuracy. We tailored our system to exploit the capabilities of omnidirectional robots and presented an all-around robot navigation suite that includes mapping, localization, trajectory generation, and robot control.

Omnidirectional robots Previous work on omnidirectional robots mostly covers fundamental control topics like position and velocity control (Rojas and Förster, 2006; Watanabe, 1998) and trajectory tracking (Liu et al., 2008). Several authors proposed approaches to determine short time-optimal trajectories without considering obstacles (Balkcom et al., 2006; Kalmár-Nagy et al., 2004; Purwin and D'Andrea, 2006). These approaches are suitable for generation of motion primitives or for smooth ad-hoc waypoint following. In our system, we resort to a family of paths that we deform according to a cost function that also depends on the obstacles in the environment.

The work by Hornung et al. (2012) for the omnidirectional PR2 robot employs Anytime Repairing A* search on a discrete set of motion primitives to generate smooth paths. As for most of the navigation systems discussed so far, a trajectory rollout controller executes these paths and follows their geometry but determines velocities ad-hoc during path execution.

Spline-based path representations Several authors represent parametric paths using splines. Hwang et al. (2003) and Mandel and Frese (2007) use cubic Bézier splines, which induce curvature discontinuities at the join points of segments. For all types of drives this implies instantaneous changes of the steering angle, infinite accelerations, or infinite jerk, with the consequence of higher tracking errors or non-smooth movements. To achieve curvature-continuous trajectories, Gulati and Kuipers (2008) and Shiller and Gwo (1991) proposed to use cubic B-splines. However, since local changes of these paths affect up to six neighboring spline segments, this representation prevents replacing or attaching a new segment to an existing trajectory, e.g., for reacting to dynamic obstacles. To overcome this problem, we apply *quintic* Bézier splines which allow us to update existing trajectories without curvature discontinuities. Quintic polynomials are also employed by Takahashi et al. (1989) to generate paths with minimized jerk for AGVs and by Piazzi et al. (2002) for curvature continuous, vision-based control of a robotic car. Splines are also used for mobile robots without wheels, e.g., Kolter and Ng (2009) propose cubic spline optimization to generate stepping trajectories for a quadruped robot. There are also authors who propose spline representations for different quantities than the robot pose directly. For example, Gulati (2011) proposes a spline-based, scaled arc

length parameterization of speed and orientation of the robot. While this has benefits for computing certain cost functions, the pose of the robot is only available through numerical integration for such representations. In our approach, we directly represent position and orientation of the robot with splines that are parameterized over an internal parameter.

Planning velocities for trajectories Many approaches to robot motion plan paths through traversable space without considering velocities at all, e.g., using algorithms like A*, D*, probabilistic roadmaps, or rapidly exploring random trees (RRTs). To execute these geometric paths, motor velocities are then typically determined during execution by controllers or reactive systems like the Dynamic Window proposed by Fox et al. (1997), see also the discussion of navigation systems above. These systems typically smooth the planned path and thereby deliberately deviate from it during execution. Some approaches more closely interweave planning and execution, e.g., the vector field histograms by Borenstein and Koren (1991), the nearness diagrams by Minguez and Montano (2004) and the potential field method by Khatib (1986). The planning horizon of these methods is however rather short and they therefore need to account for local minima.

For the above approaches, the exact behavior of the system is difficult to predict and the amount of smoothing and deviation from the planned path depends on carefully tuned parameters. Furthermore, reactive approaches that do not account for velocities and acceleration limits can run into overshooting problems when following geometric paths, as shown by Stachniss and Burgard (2002) for the example of a sharp turn when entering a room from a corridor. Stachniss and Burgard propose applying A* in a five-dimensional search space including discretized translational and rotational velocities to address the problem. They guide their search by a corridor around a two-dimensional path and compute a path consisting of circular arcs that yield curvature discontinuities at their join points.

The field of kinodynamic motion planning includes velocities into the planning problem. The aim is to compute minimal-time trajectories that avoid collisions and respect constraints on velocities and accelerations that can depend on obstacles, e.g., a speed limit depending on obstacle distance (Donald et al., 1993). The resulting trajectories map time to configuration, velocities and accelerations of the robot. If the constraints of the robot are specified correctly, one can typically execute such trajectories with high accuracy.

LaValle and Kuffner Jr. (2001) employ RRTs to find trajectories connecting start and goal configuration in high-dimensional state spaces including velocities, e.g., for simulated hovercrafts or space crafts. With RRT* proposed by Karaman and Frazzoli (2011) there exists an algorithm that is able to compute optimal paths in the limit. However, it requires an optimal solution to a two point boundary value problem between two elements of the state space. While straight lines realize this for geometric problems with path length as cost function, problem domains involving velocities often have no known solution to optimally connect two configurations. Karaman et al. (2011) present an application to

a robotic fork lift, employing minimal arc length curves as optimal connection between configurations. Their approach continuously improves the path and plans back to it in case of deviations during the execution. Webb and van den Berg (2013) present a variant of RRT* that employs a cost function that combines squared control effort and time. In this way, they can compute optimal connections between configurations for simulated robots with linear dynamics. However, they cannot constrain the velocity of the robot in their approach. In our system, we iteratively modify the trajectory shape in response to changes in a cost function that incorporates the time of travel for the robot. While we cannot guarantee reaching a global optimum with our optimization, we are however able to incorporate the cost function and constraints relevant for practical applications. Furthermore, our optimization scheme is able to substantially improve on the initialization within the short planning time necessary for real-world applications in changing environments.

Deforming and optimizing paths Instead of concatenating paths from a discrete action set, a number of approaches employ parametric path representations to inherently plan smooth paths. By modifying the parameters of such representations, one can deform these paths to account for obstacles (Quinlan and Khatib, 1993; Brock and Khatib, 2002; Lamiraux et al., 2004; Connors and Elkaim, 2007; Fraichard and Delsart, 2009; Yang and Brock, 2010). The elastic bands by Quinlan and Khatib (1993) adjust such paths to evade dynamic obstacles using circular approximations of the free space around the robot. Brock and Khatib (2002) adapted this idea to more complex robots with multiple degrees of freedom. Yang and Brock (2010) extended the approach to elastic roadmaps that they deform to account for the motion of obstacles in the environment. Connors and Elkaim (2007) proposed to model possibly colliding trajectories using splines and to establish collision-freeness by iteratively moving control points off the obstacles. However, their approach cannot guarantee to find a collision-free solution. Lamiraux et al. (2004) also deform paths to evade obstacles, but specifically consider nonholonomic constraints. Fraichard and Delsart (2009) also deform trajectories to restore collision-freeness after dynamic obstacles invalidated them.

There are also approaches that deform trajectories by optimization (Ratliff et al., 2009; Kalakrishnan et al., 2011; Schulman et al., 2013; Byravan et al., 2014). Ratliff et al. (2009) propose CHOMP, a gradient descent approach to compute collision free trajectories of a predetermined duration. They optimize with respect to the workspace integral of an obstacle distance dependent cost function. Byravan et al. (2014) present T-CHOMP, an approach that extends this concept to time-dependent cost functions and also optimizes the timing of trajectories. Kalakrishnan et al. (2011) propose a stochastic, gradient-free optimization to compute collision free trajectories for a fixed duration. Their cost function is a combination of smoothness, a minimum distance to obstacles, torque limits and constraints on the end effector pose. Schulman et al. (2013) propose to find collision free

paths with sequential convex optimization, repeatedly increasing penalties for violated constraints. In contrast to the above approaches our system starts optimization from a feasible initialization and uses the time of travel as cost function to balance curve radii, closeness to obstacles and arc length of the path while respecting constraints on velocity and acceleration.

In summary, we present a complete navigation system that focuses on accuracy, efficiency and safety and explicitly accounts for the maneuvering capabilities of omnidirectional platforms.

2.7 Conclusion

In this chapter, we presented a system for accurate, efficient and safe navigation in industrial environments. By devising smooth, curvature continuous trajectories we achieve accurate execution with an error feedback controller. Our system optimizes trajectories for travel time while taking into account platform and safety constraints and leveraging the high maneuverability of omnidirectional robots. Our extensive experiments in simulation and the real world demonstrate the robustness and accuracy of our system and showcase its applicability to the domain of industrial factory floor logistics. There is a direct impact in the industry as the proposed navigation system has been integrated on the industrial mobile omnidirectional platform KUKA omniRob together with a system for accurate localization and docking at target locations which we present in the next chapter.

As shown in the experiments, our navigation system follows its planned trajectories with an accuracy that is higher than the resolution of the grid map employed for localization of the robot. However, when arriving at a destination in flexible industrial automation scenarios, this accuracy might not be enough. Connecting a navigation task to the next step in a process chain often requires docking of the vehicle with accuracy in the order of millimeters. Traditionally, automation systems achieve such accuracy with additional mechanical docking infrastructure or sensors. Since our goal is to enable truly flexible and economic industrial automation, we introduce in the next chapter a high accuracy localization and docking system for taught locations that only relies on the safety laser scanners of a platform.

3 High Accuracy Localization and Docking at Target Locations

The navigation system presented in the previous chapter enables robots to autonomously reach a goal location. In the context of automation, such a goal location typically constitutes a junction with the next step in a process chain. These junctions often require very high localization accuracy, for example to exchange load or to perform manipulation actions. Traditionally, automation solutions employ additional infrastructure and sensors to achieve the required accuracy at the cost of reduced flexibility. In this chapter, we present a highly accurate localization system that only relies on the mandatory safety laser scanners of mobile platforms. We combine Monte Carlo localization on a grid map with a scan matcher that operates directly on the sensor data to achieve highly accurate localization and docking at previously taught target locations. We furthermore present means to increase accuracy and robustness in application scenarios that feature multiple safety laser scanners on the mobile robot. Our evaluations show that localization and docking accuracies of a few millimeters are possible when only relying on laser scanners.

In the previous chapter, we presented a navigation system that enables robots to autonomously drive to a goal location. While we designed the system for accurate execution of trajectories, it relies on global localization in a discretized grid map of the environment, which limits the achievable accuracy at a goal location. In the context of automation, however, the goal location of a navigation task typically constitutes a junction in a larger process chain. To reliably connect two processes, the robot often needs to accurately dock at specified locations to exchange load or to perform manipulation actions. Figure 3.1 shows an example situation in which the robot drove to a work station to deliver and pick up parts. So far, the required positioning accuracy is typically established through mechanical docking infrastructure or by dedicated markers and sensors at the target location and on the vehicle. This additional infrastructure yields direct costs for the involved hardware and its initial configuration and setup. In addition, the costs for reconfiguration of these systems substantially reduce their flexibility and impose an economical burden on changing processes.

Figure 3.1: An autonomous platform drives to a work station and docks to it, relying only on its safety laser scanners. The positioning accuracy is high enough to carry out subsequent manipulation actions.

To make accurate localization and docking more flexible, we consider in this chapter a system that only relies on the mandatory safety laser scanners of mobile platforms. In the situation in Figure 3.1 the robot relies only on its laser scanners to dock at the work station. Since the robot is able to position itself with high accuracy, it can directly continue with a manipulation action. The instruction of the system consists in manually driving the robot to the target location and recording reference data from the laser scanners. This procedure is intuitive and can be quickly conducted by non-experts, which further reduces the costs and increases the flexibility of the system. When returning to the target location later, the system is then able to give an accurate estimate of the platform offset from the previously taught pose. The platform can then actively correct this offset by repositioning itself or account for it by adapting subsequent manipulation actions.

The system that we present in this chapter combines existing techniques for global localization and scan matching and adds measures to increase robustness in applications that feature multiple safety scanners per robot. As a key contribution we evaluate how accurate a system relying only on safety laser scanners can get to provide a decision-making basis for application designers in automation. We perform our extensive evaluation with a motion capture studio that achieves sub-millimeter precision. Using this ground truth system, we are able to show in our experiments that our system can achieve a localization and positioning accuracy of a few millimeters at previously taught target locations.

The remainder of this chapter is organized as follows. We first present our system in Section 3.1 before we report on its evaluation in Section 3.2. Then, we discuss related works in Section 3.3 and conclude the chapter in Section 3.4.

3.1 High Accuracy Localization

Our system for accurate localization builds on global localization in a grid map of the environment. As introduced in Section 2.2, we build such a map using a graph-based formalization of the simultaneous localization and mapping problem. During the instruction phase as well as during autonomous operation, we localize the robot in the grid map with a variant of Monte Carlo localization (MCL) that adapts the number of particles according to KLD sampling, see Section 2.2.

To teach a new or updated target location to the system, the user manually drives the robot to the desired location and triggers the recording of the reference configuration. This reference configuration consists of the current global localization according to MCL and of the current readings of the laser range finder. To estimate the offset of the platform from the taught target location later during autonomous operation, the system first computes an initial guess by comparing the current MCL pose estimate with the one stored for the reference location. A scan matcher refines this offset estimate to an accurate relative localization by comparing the current laser readings with the stored reference scan. Based on the accurate localization with respect to the target location, the system can now use a position controller to perform a correction maneuver if it needs to dock at the specified position. In the case of manipulation actions, the succeeding process could also simply adapt to the known displacement if it is sufficiently small.

In the following, we introduce the employed scan matcher and present means to further increase robustness and accuracy of the system in real-world applications.

3.1.1 Scan matching

Figure 3.2 visualizes the basic principle of a scan matcher. The left part shows robot locations and laser scans sensing a nearby structure when recording the reference location (blue) and when determining the offset to this reference location (red). The middle part shows how both situations look from the sensor point of view, the structure of the environment appears rotated and translated between the reference scan and the measurement. With the help of an initial guess, in our case provided by MCL, the scan matcher finds a transformation $\mathbf{l}$ between the two sensor poses that aligns the sensed structure in the scans, see the right part of Figure 3.2. Based on this sensor offset $\mathbf{l}$ and on knowledge of the mounting location $\mathbf{c}$ of the laser scanner on the robot (extrinsic calibration), we can then infer the offset $\mathbf{m}$ of the robot pose with respect to the target location:

$$\mathbf{m} = \mathbf{c} \oplus \mathbf{l} \ominus \mathbf{c}, \tag{3.1}$$

where $\mathbf{m}, \mathbf{c}, \mathbf{l} \in SE(2)$ are transformations in the plane and $\oplus$ and $\ominus$ are the compounding operator and its inverse, respectively (Smith et al., 1990).

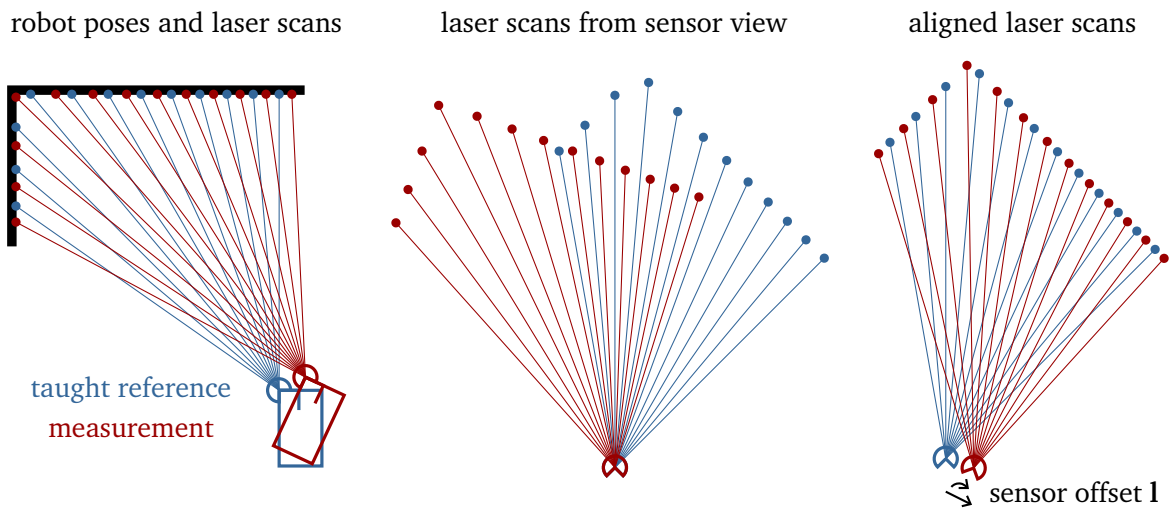


Figure 3.2: We use a scan matcher to localize the robot relative to a taught reference pose at which we recorded a reference scan (left). By aligning the current measurement (red) with the reference (blue) the scan matcher can infer the offset $\mathbf{l}$ between the position of the sensor when taking the reference scan and the current measurement (right). From this offset we then infer the relative position of the robot with respect to the taught reference location.

In our system we use the scan matcher proposed by Censi (2008). It builds on the iterative closest points algorithm (ICP) that iteratively establishes correspondences between points on the reference scan and the current measurement and applies a transformation that minimizes the displacement for these correspondences. The scan matcher proposed by Censi employs a point-to-line metric to determine the displacement between corresponding points. Figure 3.2 depicts a motivation for this metric: From different poses, the laser scanner detects different points on the structure. Minimizing an accumulated point-to-point error instead of the point-to-line metric would introduce an error in the sensor offset estimate, as the scan matcher would slide the measurement scan along the longer line segment in the environment structure in the right part of Figure 3.2 to reduce the error introduced by the non-coinciding end points of the laser beams. To achieve robustness against moderate dynamics in the environment, the scan matcher also performs outlier rejection with a trimmed version of the ICP algorithm as proposed by Chetverikov et al. (2002). This variant neglects the correspondences with the highest error when computing the transformation between the scans.

The ICP algorithm can converge to a local minimum when provided with a far-off initial guess, especially when the orientation component of the initialization is erroneous. In our system, the combination with MCL localization provides good initial guesses, in particular the orientation estimate of MCL is typically consistent.

3.1.2 Robustness and accuracy for real-world applications

In our system we employ several means to reduce the influence of sensor noise and to increase the robustness of our approach. Measurements of laser range finders are subject to noise. Since the robot is standing still while teaching a reference location and typically also while estimating its current offset to the reference location, we can aggregate multiple laser scanner readings into an averaged scan. By averaging the measurements for individual beams and rejecting beams with a variance above a threshold we are able to remove noise and spurious readings from both the reference and the measurement scan.

If the robot is equipped with multiple safety laser scanners we can use this as redundancy to detect failures in the scan matching process. If the reference location offset estimates differ too much between different sensors, we can infer an error in the scan matching process, possibly caused by changes in the environment, erroneous sensor readings or a degenerated initial guess. Instead of acting based on the erroneous scan matching result, the system can then decide to either report failure or to slightly reposition itself for a further attempt at determining the offset from the target location. If there is consensus between the offset estimates of the individual sensors the system can base its actions on the average of the reported offsets.

For docking at a target location, the offset computed by our system acts as input to a position controller that performs a compensating motion. For robust docking, we alternate offset estimation and repositioning of the platform until the offset from the target location falls below a threshold. First, this increases the robustness of the overall system against execution errors of the controller. Second, this improves the quality of the final offset estimate as the estimates of the scan matcher get better the closer the current sensor view matches the one of the recorded reference scan.

Depending on the structure of the environment captured in the laser scans, a scan matcher can be more certain in its estimate along a specific direction in the environment than along others. Censi (2007) proposes a way to compute a covariance $\mathbf{L}$ for a scan matcher result $\mathbf{l}$ that captures this uncertainty of the estimate. If the robot is equipped with multiple safety scanners $i \in [1, \ldots, n]$, we can employ this covariance $\mathbf{L}_i$ for each scan matcher result $\mathbf{l}_i$ of the n safety scanners to accordingly aggregate the individual estimates. In addition to transforming the individual sensor offsets $\mathbf{l}_i$ into robot pose offsets $\mathbf{m}_i$ via Eq. (3.1), we also need to transform the associated covariance matrices $\mathbf{L}_i$ to covariances $\mathbf{M}_i$ in the robot frame before fusing the estimates:

$$\mathbf{M}_i = \mathbf{R}(\theta_i)^\top \mathbf{L}_i \, \mathbf{R}(\theta_i), \tag{3.2}$$

where $\mathbf{R}(\theta)$ is the rotation matrix for θ in homogeneous coordinates and θ_i is the rotational part of the extrinsic sensor calibration $\mathbf{c}_i$. This is an optimistic approximation since it neglects the additional uncertainty in the position of the robot that results from the rotational uncertainty of the sensor pose. However, in scan matching the orientation

uncertainty of the sensor pose is typically very small.

Finally, we use covariance intersection according to Niehsen (2002) to combine the series of estimates $\mathbf{m}_i$ of the robot offset from the target location with uncertainties $\mathbf{M}_i$ to the fused offset $\mathbf{m}$ with uncertainty $\mathbf{M}$:

$$w_i = \frac{\operatorname{tr}(\mathbf{M}_i)^{-1}}{\sum_{i=1}^{n} \operatorname{tr}(\mathbf{M}_i)^{-1}}, \tag{3.3}$$

$$\mathbf{M} = \left(\sum_{i=1}^{n} w_i \mathbf{M}_i^{-1} \right)^{-1}, \tag{3.4}$$

$$\mathbf{m} = \mathbf{M} \left(\sum_{i=1}^{n} w_i \mathbf{M}_i^{-1} \mathbf{m}_i \right), \tag{3.5}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix.

A prerequisite for an adequate performance of the system is an accurate extrinsic calibration $\mathbf{c}_i$ of the involved laser scanners. To determine this calibration we employ the method proposed by Kümmerle et al. (2012), who add the extrinsic calibration of sensors as additional nodes in a graph-based formulation of the simultaneous localization and mapping problem. The method can calibrate the sensor mounting pose by moving the robot through the environment and can therefore also monitor and adapt the calibration online during long-term operation. With accurate calibration of sensor locations and the usual horizontal mounting of safety scanners it is possible that different vehicles of the same type can use the reference locations taught to only one specific vehicle of the fleet. Here, we need to adapt Eq. (3.1) to account for the calibration $\mathbf{c}^{\text{ref}}$ of the sensor on the vehicle used to teach the location and the calibration $\mathbf{c}^{\text{meas}}$ on the vehicle that determines its offset from the target location:

$$\mathbf{m} = \mathbf{c}^{\text{ref}} \oplus \mathbf{l} \ominus \mathbf{c}^{\text{meas}}. \tag{3.6}$$

3.2 Experimental Evaluation

In this section, we evaluate our system for accurate localization with respect to taught target locations. We furthermore use the determined offsets to the target location to perform a compensating motion with the robot. We first introduce our evaluation methodology, that we base on a motion capture studio. Then, we present real-world experiments that combine navigation and docking with the omnidirectional robot omniRob that we already used in the experiments of the previous chapter. We also present experiments on a larger-scale platform that is equipped with four safety laser scanners.

3.2.1 Evaluation with a motion capture studio

To evaluate or system we use a Motion Analysis motion capture studio with nine Raptor-E cameras (mocap) as ground truth. The mocap triangulates the position of reflective markers with high precision. With the help of multiple markers that are rigidly attached to the robot the system can precisely determine the 6D pose of the robot. In an initial calibration of the mocap, markers attached to the floor allow to align the coordinate system such that we can perform all computations in the space of translations and rotations of the robot on the floor plane.

In our experiments, the robot stands still when we assess its pose with the mocap. To eliminate a slight jitter in the estimates we aggregate multiple measurements of the mocap that provides robot pose estimates with up to 300 Hz. In several tests, in which we covered and uncovered the markers on the robot to simulate returning to the exact pose for the mocap, the difference in pose estimates was in the order of 0.1 mm.

After initial calibration the mocap relies on its cameras remaining fixed in the environment. Depending on the camera mounting even temperature changes can cause slight movements of the cameras with respect to each other, jeopardizing the precision of the mocap. To counter this issue we calibrate the mocap right before starting our experiments and analyze after the experiments whether the assessed errors increased over time. This was not the case for any of our experiments, which eliminates decalibration as a bias for our evaluation.

To evaluate our approach we record the pose reported by the mocap along with the MCL pose and the reference scan when teaching target locations. Then, whenever our system has determined an offset to a target location during evaluation, we can compare this offset with the offset between the current pose of the mocap and the one recorded during location teaching. This difference between the offset estimates by our approach and the mocap constitutes the *localization error* that evaluates the performance of our localization system. In addition, we can also assess the *positioning error* whenever the robot fulfilled a docking task of returning to the target location as closely as possible. The positioning error consists in the offset between current mocap pose and the one recorded during teaching of the target location.

3.2.2 Experiments with the omniRob platform

In this experiment, we evaluated our approach to accurate localization and docking with the omnidirectional omniRob that we also used in the navigation experiments of the previous chapter. It is equipped with two SICK S300 safety laser scanners with a 270 degree field of view and 541 beams each. Figure 3.3 shows the experiment environment together with the grid map that we built for MCL as described in Section 2.2. We defined three target locations L_1, L_2, L_3 in the environment and taught them to the

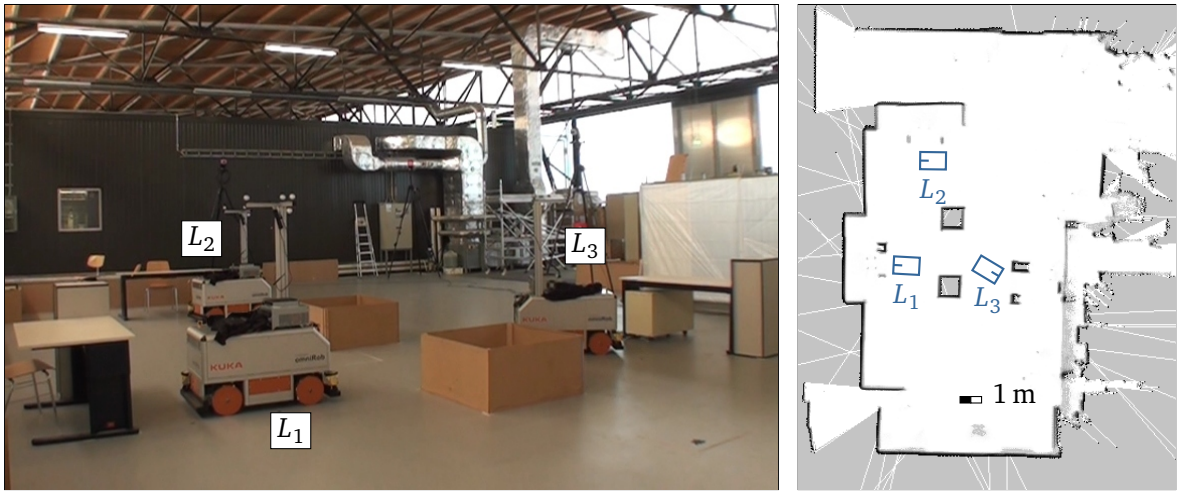


Figure 3.3: Setup for our experiments with the omnidirectional robot omniRob, shown on the left at the three taught target locations L_1, L_2, L_3. The right part of the figure shows the grid map used for MCL with darker cells representing obstacles. The task of the robot was to repeatedly drive to the target locations and position itself at them as accurately as possible.

localization system, also recording the mocap pose as described above. Then, the robot had to repeatedly drive to a randomly chosen target location with the navigation system presented in the previous chapter and to subsequently use the offset estimation from our approach to position itself accurately at the target location. We iterate offset estimation and repositioning twice for each docking action before we estimate a final offset from the target location and assess the localization and positioning error as described above.

We first conducted this experiment in a static environment, lasting for approximately 3.5 hours in which the omniRob performed 390 combined navigation and docking actions. The combined navigation system was always able to reach the goal and to position itself at the target location. Figure 3.4 presents the resulting positioning and localization errors. The figure shows the maximum values as crosses along with mean and standard deviation at all target locations. The average positioning error, i.e., the offset between platform pose and target location was 0.005 m and 0.15 deg. The average localization error was 0.003 m and 0.06 deg. An important observation is that also the maximum errors are small with 0.012 m and 0.48 deg for positioning and with 0.01 m and 0.27 deg for localization with respect to the target location. For comparison, the localization error for MCL on a grid map with 0.05 m resolution typically had an average of 0.03 m and 0.18 deg and a maximum of 0.055 m and 0.6 deg. The results in Figure 3.4 also show that the positioning error is consistently higher than the localization error. The reason lies in limitations of the employed positioning controller when compensating for very small pose offsets.

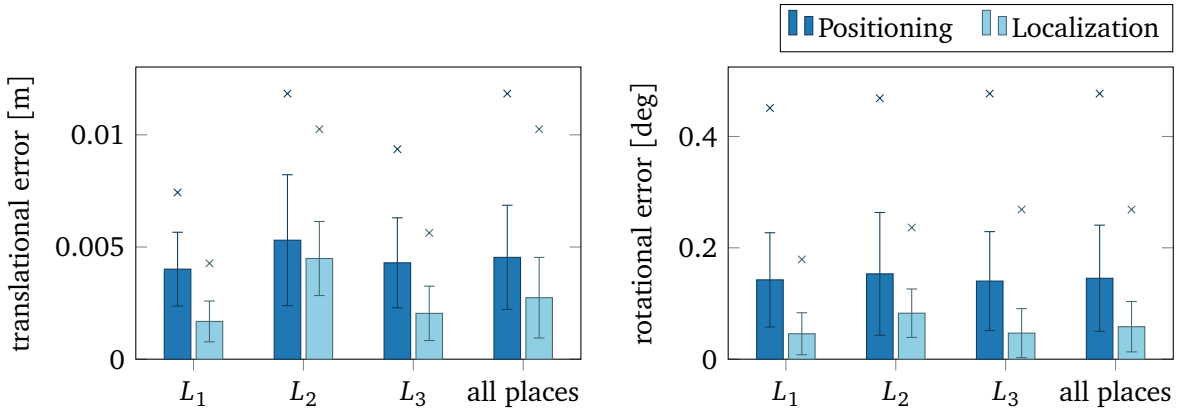


Figure 3.4: Positioning and localization errors for 390 docking actions performed by the omniRob in the static experiment shown in Figure 3.3. The plots show mean error and standard deviation along with the maximum error encountered (crosses).

Effects of environment dynamics

In a further experiment, we introduced dynamics to test the robustness of our approach against changes in the environment. We realized three levels of dynamics. The first level featured three people walking through the environment. At the second level up to eight people populated the environment and positioned themselves closer to the robot. At the third level of dynamics we additionally used wooden boards to obstruct the sensors of the robot and to create ambiguous structures. Figure 3.5 shows several pictures of this experiment.

The navigation system successfully negotiated the environment dynamics at all three levels and always reached the target location. Our accurate localization at the target location always succeeded for levels one and two and only reported an error for a few actions at the third level of environment dynamics. Here, the disturbances caused by people and wooden boards were strong enough to yield too different offset estimates from the two safety scanners. Consequently, our system reported an error instead of averaging the estimates and performing a repositioning action. Figure 3.6 shows the results of the experiment. Positioning and localization errors increase compared with the static environment, although not drastically. Over all level of dynamics, the average positioning error is at most 0.008 m and 0.175 deg and the maximum positioning error over all levels is 0.017 m and 0.527 deg. The localization error is lower, the highest average error for any level is 0.005 m and 0.098 deg and the maximum localization error over all levels of dynamics is 0.012 m and 0.258 deg.

Figure 3.5: Experiment with the omniRob in a dynamic environment in which people and wooden boards changed the appearance of the environment. Figure 3.6 shows the results for different levels of environment dynamics.

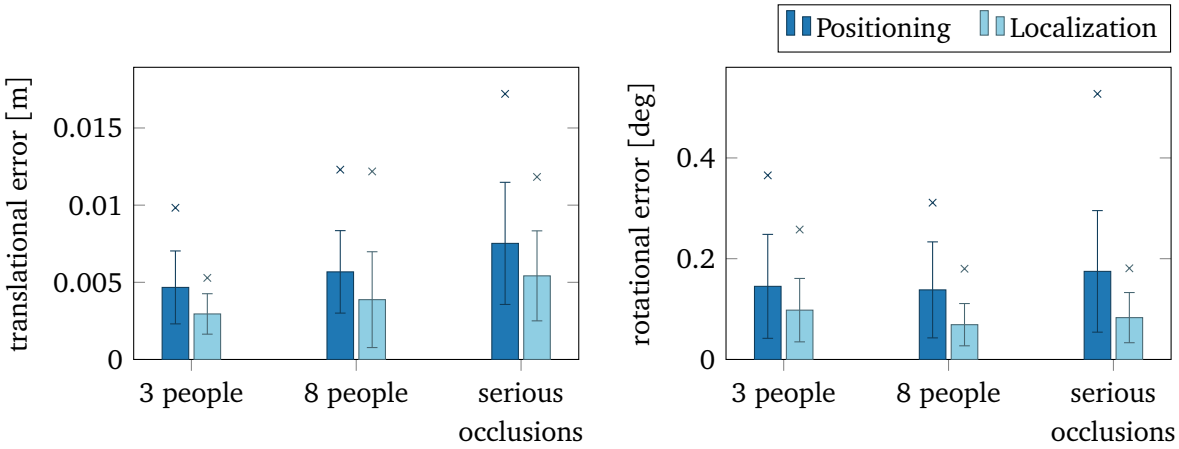


Figure 3.6: Positioning and localization errors for our experiment in a dynamic environment with the omniRob. The system performed 39 (3 people), 16 (8 people) and 16 (serious occlusions) positioning actions for the respective levels of environment dynamics.

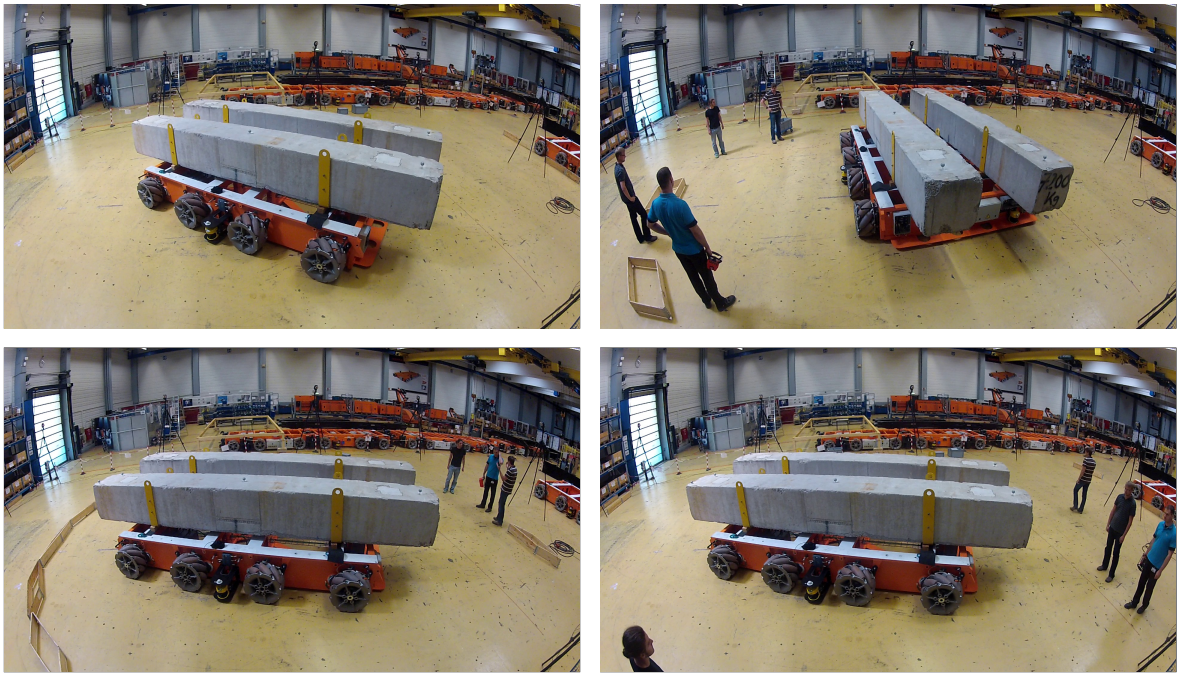


Figure 3.7: Localization experiments in a changing environment with the large-scale KUKA omniMove platform, which is equipped with four safety laser scanners. See Figure 3.8 for the results of this experiment.

3.2.3 Experiments with the omniMove platform

In this experiment, we evaluated our approach on a larger-scale KUKA omniMove platform that is equipped with four SICK S3000 safety laser scanners with a 190 degree field of view and 761 beams. Figure 3.7 gives an impression of the employed 11 ton omniMove vehicle. In this experiment, we fuse the offset estimates from the four safety scanners with the covariance intersection method introduced in Section 3.1.2. As the positioning controller of this platform was still in a development stage we do not evaluate the positioning error for this platform and restrict ourselves to the localization error only.

We conducted trials in the factory floor environment shown in Figure 3.7. As before, we once kept the environment static and introduced dynamics through walking people and wooden boards in a second trial, see the top left versus the remaining panels of Figure 3.7. In the static experiment the platform performed 124 positioning actions distributed over four target locations and in the dynamic experiment it performed 38 positioning actions at two target locations. The system was always able to drive to the target location and never reported an error when estimating its offset from the target location. Figure 3.8 shows the results of the experiments. The localization error is low for both, the static and the dynamic environment, and never exceeds 0.005 m and 0.052 deg with an average of 0.001 m and 0.007 deg.

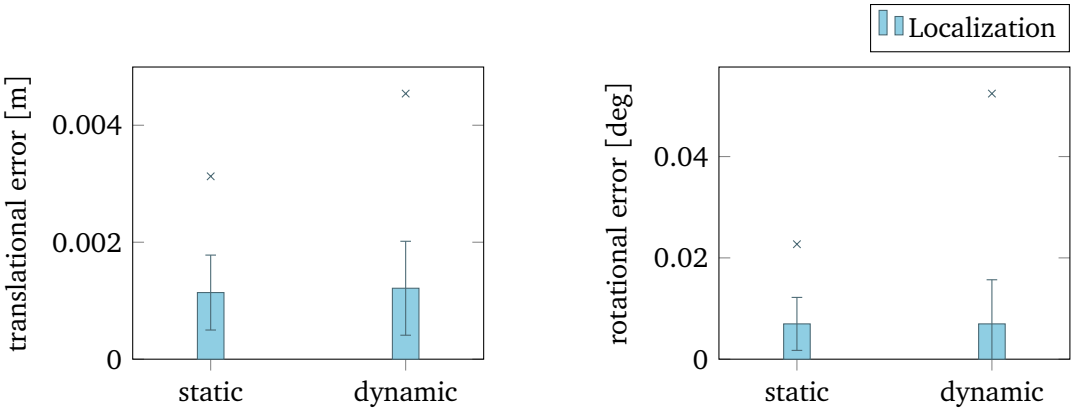


Figure 3.8: Localization error for the experiments with the large-scale KUKA omniMove platform in a static and a dynamic environment, see Figure 3.7. The robot performed 124 actions in the static and 38 actions in the dynamic environment.

In summary, the experiments with the omniRob and the omniMove platform show that our approach is capable of accurate localization and docking at target locations with errors in the range of a few millimeters. This capability also transfers to situations in which the environment changes moderately between teaching the target location and when localizing with respect to it.

3.3 Related Work

Localizing a mobile robot with respect to a given map is a fundamental task in robotics and has therefore received a lot of attention by researchers, see for example the approaches proposed by Leonard and Durrant-Whyte (1991); Fox et al. (1999); Dellaert et al. (1999b,a); Gross et al. (2002, 2003); Elinas and Little (2005); Andreasson et al. (2005); Bennewitz et al. (2006). For robust localization of a mobile robot probabilistic approaches explicitly account for the involved uncertainties and probability distributions. Successfully applied techniques in this domain are the extended Kalman filter (EKF) (Leonard and Durrant-Whyte, 1991), the histogram filter (Fox et al., 1999) and MCL, which we use as initialization for our scan matching approach (Dellaert et al., 1999b,a; Gross et al., 2002, 2003; Elinas and Little, 2005; Andreasson et al., 2005; Bennewitz et al., 2006). The literature also refers to MCL techniques as particle filters or condensation.

Researchers have employed different sensors for robot localization including wireless receivers, cameras and laser range finders. Wireless receivers that estimate the strength of radio signals are used in the approaches of Ferris et al. (2005); Duvallet and Tews (2008); Cho and Kim (2010); Quigley et al. (2010). Based on the known positions of signal emitters and chirp-spread-spectrum ranging, Cho and Kim (2010) propose an EKF-based

method to localize a mobile robot. The approaches of Ferris et al. (2005) and Duvallet and Tews (2008) rely on WiFi signals to localize the robot and both employ Gaussian processes to probabilistically model the signal strength of wireless networks. In addition to WiFi signals, the localization approach proposed by Quigley et al. (2010) relies on consumer-grade inertial measurement units and cameras. While the achieved sub-meter accuracy suffices for contexts like smartphone localization, it is typically not enough for industrial applications.

A large body of work proposes cameras to localize mobile robots. Dellaert et al. (1999a) first proposed to use MCL for a vision-based approach, a concept then followed by many researchers. A group of approaches uses MCL with panoramic images for mobile robot localization, e.g., Gross et al. (2002, 2003); Menegatti et al. (2004); Andreasson et al. (2005). Andreasson et al. (2005) report an accuracy of around 1 m while Gross et al. (2002, 2003) achieve accuracies between 0.4 m and 0.8 m. Menegatti et al. (2004) report an error below the distance of reference images, which was 0.2 m in their experiments. There are also approaches that rely on perspective cameras and build a database of landmarks using visual features. In this context, the scale-invariant feature transform (SIFT) has retrieved considerable attention. Se et al. (2002) first used this feature for localization of a robot, however without tracking its pose over time. Elinas and Little (2005) employed stereo cameras and SIFT features for MCL and report a localization error of 0.2 m. Bennewitz et al. (2006) rely on SIFT features in images from only a monocular camera in their MCL-based localization for which they report an average error of 0.39 m.

For our approach, we employ laser range finders which are popular in robotics since their readings need little processing as they directly measure distances to objects in the environment. In the industrial context another advantage is that often regulations prescribe the use safety laser scanners for mobile platforms anyway, therefore other sensors would only cause additional costs. For MCL-based localization on grid maps Dellaert et al. (1999b) and Thrun et al. (2001) report errors between 0.05 m and 0.2 m with SICK laser range finders. Gutmann et al. (1998) compared grid map-based MCL localization and scan matching-based localization with Kalman filters. In their work they reach the conclusion that while scan matching techniques are more accurate, they tend to fail in situations with higher noise and uncertainty, whereas MCL based on a grid map performs more robustly in such situations. In our approach, we rely on the robust MCL and refine its estimate by scan matching near the target locations. Furthermore, since the work of Gutmann et al., that was already conducted in 1998, mocap systems have become available which provide us with a more precise ground truth for the evaluation of localization techniques.

While the laser-based MCL approaches discussed above rely on occupancy grid maps, Saarinen et al. (2013) propose to represent the environment and laser measurements with a normal distribution transform (NDT). Biber and Strasser (2003) suggested to use the NDT for laser scan matching and Magnusson et al. (2007) present an extension to

3D data. With NDT, a set of normal distributions model laser scan measurements and the environment in a piecewise continuous fashion. With this representation, Saarinen et al. propose NDT-MCL, which reduces the localization error compared with grid map MCL. They show this in an evaluation in which they switch the localization input of an industrial automated vehicle from a commercial, infrastructure-based localization system (Hyyppa, 1989) to their method and to grid map MCL. Valencia et al. (2014) present an extension of NDT-MCL that accounts for dynamics in the environment by representing it at two distinct time scales. While our method achieves very high accuracy by working with raw sensor data at the target locations, NDT-MCL similarly to grid map MCL relies on a discretized representation of the environment. On the other hand, NTD-MCL is applicable in the whole mapped environment while our method, as presented in this chapter, is only tailored towards highly accurate localization at specific goal locations.

In industrial systems, localization is typically realized with sensors that rely on external infrastructure such as reflective or optical markers, guidance wires or magnets (Wurman et al., 2008; swisslog, 2015; Frog AGV Systems, 2015). In this chapter, we presented a solution that achieves localization accuracy suitable for docking maneuvers based on safety laser scanners only. In addition to presenting means for increasing accuracy and robustness in real-world applications, we employ a highly precise mocap system as ground truth to conduct an experimental evaluation of our approach, which we integrated with our navigation system.

3.4 Conclusion

In this chapter, we presented a method to localize a mobile robot with respect to a target location with high accuracy. This is a key technique to dock mobile robots and join processes that involve navigation. We base our method on MCL localization and apply scan matching against reference laser scans that were previously recorded at the target locations. Our method relies only on safety laser scanners as sensors and is able to achieve an accuracy of a few millimeters through the use of non-discretized sensor data at the target locations. The reported accuracies form a dependable information basis for decision makers and application designers in automation since we evaluated our method with real-world experiments using two different mobile platforms in a highly precise motion capture studio.

In our evaluation we combined our method for accurate localization and docking at target locations with the navigation system presented in the previous chapter. With this combination, we can now connect navigation tasks to processes that require accurate positioning of the platform, e.g., for charging, load exchange or manipulation actions.

When transitioning from conveyor belts and guided systems in automation to more flexible and versatile autonomous systems, full autonomy of such platforms is sometimes

too drastic of a change. In certain scenarios safety considerations, regulations, as well as expectations and concerns of human workers sharing the shop floor with autonomous platforms make behaviors that follow a predefined route the right choice. Therefore, we present in the next chapter a method to intuitively instruct route following navigation tasks to mobile platforms that only rely on their safety laser scanners as sensors. We will revisit scan matching techniques in Chapter 5, in which we not only teach target poses as presented in the current chapter, but instead teach and accurately repeat docking maneuvers and longer trajectories.

4 Fitting Trajectories to User Demonstrations

The previous chapters proposed a system for fully autonomous navigation. In the industrial context, there are also application scenarios in which it is more appropriate to strictly follow a predefined path and establish predictability of system behavior for the human observer for safety reasons. In this chapter, we present an approach to fit trajectories to user-demonstrated reference paths to increase the flexibility and usability of navigation systems that follow such virtual rails. Given a recorded reference trajectory, we apply non-linear least-squares optimization to accurately fit a parametric curve to the reference. We employ the path model proposed in Chapter 2 as representation and thereby require substantially fewer free parameters than standard approaches to achieve similar residual errors. Our experiments on real-world data demonstrate the advantages of our method in comparison with standard approaches. We also show how one can use the optimization method from Chapter 2 to refine the shape of the demonstrated trajectory and how to adapt the replanning approach from Chapter 2 for robust execution of the virtual rails generated by the approach in this chapter.

With the combination of Chapter 2 and Chapter 3 we have presented a system that is able to navigate a robot to a goal pose with full autonomy. The system optimizes its behavior to reduce the travel time to the goal. However, full autonomy is not always required, possible or demanded in navigation tasks in industrial scenarios. Regulations, safety considerations, or simply habits of the involved people can make a system the right choice that will follow a predefined path through the environment with high accuracy. Among the benefits of such a system are the high predictability of its behavior for humans and the possibility to implicitly encode knowledge into the predefined path that is inaccessible to the sensors of the robot, such as lane markings for example.

Due to the above-mentioned benefits, the majority of mobile transportation platforms in industry today are automated guided vehicles (AGVs) that carry loads on predefined paths, typically coupled to their route by magnetic or optical strips and markers (Mäkelä and von Numers, 2001; Leea and Yang, 2012; Fujimoto et al., 2001; Wurman et al., 2008;

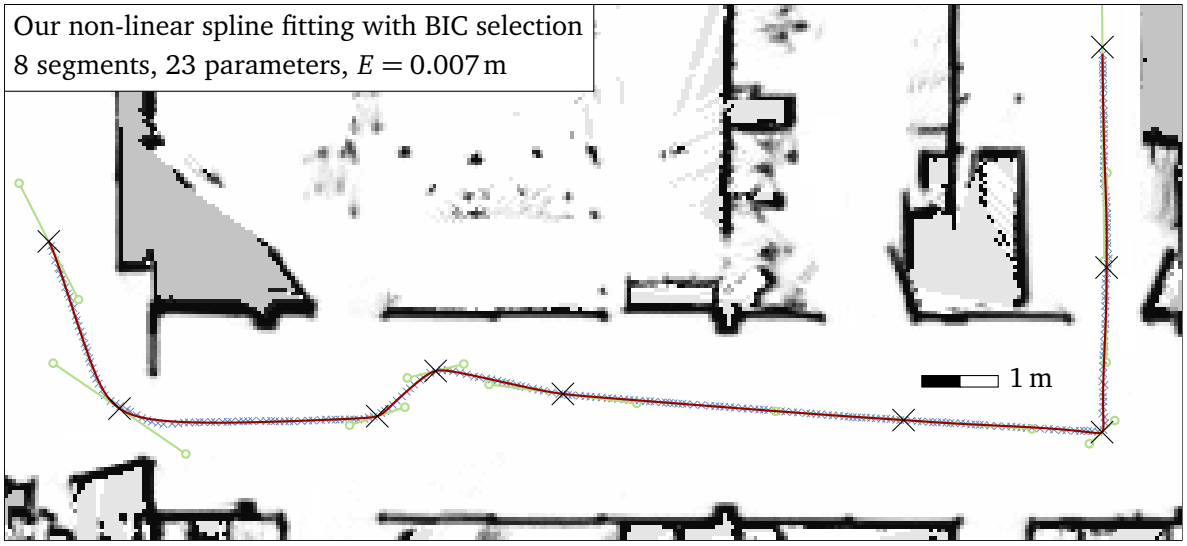


Figure 4.1: Fitting our path model (red) to a reference trajectory (blue crosses). Our non-linear optimization places control points in curve apices and adjusts their position (black crosses) and tangents (green markers). Thus, we achieve more accurate fits with less parameters compared with standard approaches.

Guizzo, 2008; Frog AGV Systems, 2015). Lately, several authors proposed approaches in the context of where to place and how to select artificial markers to localize a robot along its route, see for example the works by Sala et al. (2006), Jourdan and Roy (2008), Vitus and Tomlin (2010), Meyer-Delius et al. (2011) and Beinhofer et al. (2011). Systems that rely on external infrastructure to guide them along their route are not very flexible and changing the route involves a substantial amount of effort and costs. Therefore, their use is only economical for sufficiently big batch sizes, i.e., the production task has to remain unchanged for a long enough time such that the setup costs of the AGV system can pay off. With industry pushing towards the automation of smaller batch sizes to retain competitiveness, the lack of flexibility of these systems becomes a critical point.

Mobile robots are often equipped with laser range finders for safety reasons. Exploiting these sensors for localization in a global reference frame opens the possibility to use these systems without external infrastructure, as is the case for the fully autonomous navigation system proposed in Chapter 2 and Chapter 3. It is possible to instantiate the path model employed by this system by drawing in a graphical user interface, similar to applications in computer aided design (CAD). In this way, the user can define virtual rails, which the mobile robot can follow without external infrastructure.

To increase the usability and to make the use of these virtual rails more intuitive, we propose a method that allows to fit virtual rails to user demonstrations, see Figure 4.1 for an example. This way, even non-experienced users can flexibly reinstruct the system by driving the robot along the new route and thereby quickly react to changes in the

production process. Since we rely on the path model presented in Chapter 2 to fit a continuous representation to the user demonstration, we can also optimize the resulting reference trajectory with the approach of that chapter if desired. Furthermore, we present an adaption of the replanning approach from Chapter 2 that allows the robust execution of a reference trajectory. By appropriate orchestration of the system proposed in the previous chapters and the approach proposed in this chapter, one can cover the spectrum between strict following of a virtual rail and fully autonomous navigation.

In particular, this chapter presents a novel approach to trajectory teaching for mobile robots by non-linear fitting of a specific path model. A key challenge is to achieve accurate path fits with a small number of parameters in the model. A low number of parameters achieves robustness to noise and furthermore reduces the complexity of a potential subsequent optimization of the fitted path. Our approach substantially reduces the number of parameters required to achieve the same fitting accuracy as standard spline fitting approaches. We employ the Bayesian Information Criterion to trade off model complexity and fit error and refine our spline fits with non-linear optimization that relaxes the correspondence between spline points and reference data.

The remainder of this chapter is organized as follows. We first formalize basic spline fitting in Section 4.1, which we will use as initialization and as baseline for comparison. Then, we present our novel methods for non-linear spline fitting in Section 4.2. To further improve the fit, we propose to locate control points near curve apices in Section 4.3, compare Figure 4.4 (B) vs. (C). In Section 4.4 we enable our method to adapt the location of control points, Figure 4.4 (D), and finally relax the correspondence between spline points and reference data to further refine the fit, see Figure 4.4 (E). We evaluate the proposed method with real-world data in Section 4.5. In Section 4.6 we then show how one can use the methods from Chapter 2 to perform further optimization of the resulting trajectories and to robustly execute them in a global reference frame. We discuss related work in Section 4.7 before we conclude the chapter in Section 4.8.

4.1 Basic Splines and Linear Fitting Techniques

In mobile robotics, odd-ordered Bézier splines are a popular parametric path representation, since they can smoothly connect a set of waypoints. See Figure 4.2 for an example, it shows the waypoints as black crosses. We also employ Bézier splines in our path representation for optimization-based trajectory generation in Chapter 2.

As introduced in Section 2.3.2, a spline segment $\hat{\mathbf{s}}(u)$ is a polynomial curve of order n, defined over an internal parameter $u \in [0, 1]$. There exist several formulations to express these polynomials as functions of shape-defining parameters. In Chapter 2 we resorted to the Bézier formulation, but in this chapter we will use the slightly different Hermite formulation. While in the Bézier formulation derivatives of the curve at start and end are

obtained by calculations from the control points, they are specified more directly in the Hermite form. Here, a spline segment is defined by control points $\mathbf{p}_i$ at its start ($\mathbf{p}_s$) and end ($\mathbf{p}_e$). Each $\mathbf{p}_i$ has $K = \frac{n+1}{2}$ parameters $\mathbf{p}_i^k$, with $k = 0, \ldots, K-1$. The $\mathbf{p}_i^k$ are vectors with one component per spline dimension, and specify the k-th derivative of $\hat{\mathbf{s}}(u)$ at the start ($u = 0$) and end ($u = 1$) of the segment. $\hat{\mathbf{s}}(u)$ is then given by the linear combination

$$\hat{\mathbf{s}}(u) = \sum_{k=0}^{K-1} h_s^k(u)\mathbf{p}_s^k + h_e^k(u)\mathbf{p}_e^k\,, \tag{4.1}$$

where $h_s^k(u)$ and $h_e^k(u)$ are polynomials called Hermite basis functions. They are obtained by solving

$$\hat{\mathbf{s}}^{(k)}(0) = \mathbf{p}_s^k, \quad \hat{\mathbf{s}}^{(k)}(1) = \mathbf{p}_e^k, \qquad k = 0, \ldots, K-1\,, \tag{4.2}$$

where $\hat{\mathbf{s}}^{(k)}$ is the k-th derivative of the polynomial $\hat{\mathbf{s}}$. By factoring out the parameters $\mathbf{p}_s^k, \mathbf{p}_e^k$ we obtain the basis functions for cubic, quintic and septic splines shown in Table 4.1. Typically, a spline curve is a concatenation of multiple spline segments. When joining M segments $\hat{\mathbf{s}}_i$, $i \in 0, \ldots, M-1$, the resulting curve $\mathbf{s}(u)$ is defined over $[0, M]$ and given by $\mathbf{s}(u) = \hat{\mathbf{s}}_i(u - i)$. Here, $\hat{\mathbf{s}}_i$ with $i = \lfloor u \rfloor$ is the active segment for a specific u, and specified by $\mathbf{p}_s^k = \mathbf{p}_i^k$ and $\mathbf{p}_e^k = \mathbf{p}_{i+1}^k$. Since adjacent segments share control points, the curve has C^{K-1} continuity, i.e., the curve and all its derivatives up to the $K-1$-th derivative are continuous.

4.1.1 Linear least-squares spline fitting

Given a reference path $\mathbf{z}(t)$, we want to find an accurate parametric approximation of this path with as few parameters as possible. $\mathbf{z}(t)$ is given by the robot position $\mathbf{z}_t = \langle x_t, y_t \rangle$ at each discrete time step $t = 0, \ldots, N-1$ and typically obtained by recording the self-localization of a manually steered robot in a global coordinate frame. We compute the cumulative length of the piecewise linear path given by $\mathbf{z}(t)$ as $l_t = \sum_{i=1}^{t} \|\mathbf{z}_i - \mathbf{z}_{i-1}\|$. Since we are fitting a geometric path and do not model the velocity of the robot, we disregard the timing information of $\mathbf{z}(t)$. We assume sparse $\mathbf{z}_t$ such that the distance between two consecutive points is greater than a threshold τ_l, i.e., $l_t - l_{t-1} > \tau_l$ for all t. With this sparsification one can control the computational complexity of the fitting and prevent fitting problems that can occur when the robot was standing still for some time during the demonstration. To approximate $\mathbf{z}(t)$ with a spline, we assign each $\mathbf{z}_t$ a corresponding u_t by linear interpolation with respect to the arc length:

$$u_t = \frac{l_t}{l_{N-1}} M\,. \tag{4.3}$$

For the basic linear least-squares fit we construct a matrix $\mathbf{X}$ with one row $\mathbf{r}_t$ for each

	Cubic ($K=2$)	**Quintic** ($K=3$)	**Parameter**
h_s^0	$2u^3-3u^2+1$	$-6u^5+15u^4-10u^3+1$	$\mathbf{p}_s^0$
h_s^1	u^3-2u^2+u	$-3u^5+8u^4-6u^3+u$	$\mathbf{p}_s^1$
h_s^2		$-\frac{1}{2}u^5+\frac{3}{2}u^4-\frac{3}{2}u^3+\frac{1}{2}u^2$	$\mathbf{p}_s^2$
h_e^0	$-2u^3+3u^2$	$6u^5-15u^4+10u^3$	$\mathbf{p}_e^0$
h_e^1	u^3-u^2	$-3u^5+7u^4-4u^3$	$\mathbf{p}_e^1$
h_e^2		$\frac{1}{2}u^5-u^4+\frac{1}{2}u^3$	$\mathbf{p}_e^2$

	Septic ($K=4$)	**Parameter**
h_s^0	$20u^7-70u^6+84u^5-35u^4+1$	$\mathbf{p}_s^0$
h_s^1	$10u^7-36u^6+45u^5-20u^4+u$	$\mathbf{p}_s^1$
h_s^2	$2u^7-\frac{15}{2}u^6+10u^5-5u^4+\frac{1}{2}u^2$	$\mathbf{p}_s^2$
h_s^3	$\frac{1}{6}u^7-\frac{2}{3}u^6+u^5-\frac{2}{3}u^4+\frac{1}{6}u^3$	$\mathbf{p}_s^3$
h_e^0	$-20u^7+70u^6-84u^5+35u^4$	$\mathbf{p}_e^0$
h_e^1	$10u^7-34u^6+39u^5-15u^4$	$\mathbf{p}_e^1$
h_e^2	$-2u^7+\frac{13}{2}u^6-7u^5+\frac{5}{2}u^4$	$\mathbf{p}_e^2$
h_e^3	$\frac{1}{6}u^7-\frac{1}{2}u^6+\frac{1}{2}u^5-\frac{1}{6}u^4$	$\mathbf{p}_e^3$

Table 4.1: Hermite basis functions and parameters for cubic, quintic and septic spline segments.

data point $\mathbf{z}_t$:

$$\mathbf{X}=\begin{pmatrix}\mathbf{r}_0\\ \mathbf{r}_1\\ \vdots\\ \mathbf{r}_{N-1}\end{pmatrix}. \tag{4.4}$$

The rows of $\mathbf{X}$ contain the Hermite basis functions for the corresponding u_t. For one-dimensional splines these rows have the form

$$\mathbf{r}_t=\Big(\underbrace{0\dots0}_{iK}\quad h_s^0\dots h_s^{K-1}\quad h_e^0\dots h_e^{K-1}\quad\underbrace{0\dots0}_{(M-i-1)K}\Big), \tag{4.5}$$

where each h^k is a function of u_t-i, and $i=\lfloor u\rfloor$ is the index of the segment active for u_t.

The spline $\mathbf{s}(u)$ at $u = u_t$ is then given by $\mathbf{s}(u_t) = \mathbf{r}_t\mathbf{p}$, where

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_0^0 \dots \mathbf{p}_0^{K-1} & \dots & \mathbf{p}_M^0 \dots \mathbf{p}_M^{K-1} \end{pmatrix}^\top \tag{4.6}$$

is the vector of control point parameters. The zeros in $\mathbf{r}_t$ ensure that only the parameters relevant for the active segment influence the computation and the h^k multiply the parameters with the correct evaluation of the basis function, compare Eq. (4.1). To fit the spline to the reference path $\mathbf{z}(t)$, we can solve the corresponding linear least squares problem in closed form:

$$\tilde{\mathbf{p}} = \arg\min_{\mathbf{p}} \|\mathbf{z} - \mathbf{X}\mathbf{p}\| = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{z}, \tag{4.7}$$

where $\mathbf{z} = \begin{pmatrix} \mathbf{z}_0 & \mathbf{z}_1 & \dots & \mathbf{z}_{N-1} \end{pmatrix}^\top$ is a vector containing the reference path. The parameters $\tilde{\mathbf{p}}$ define a spline $\mathbf{s}(u)$ that minimizes the sum of the squared residual errors

$$r^2 = \sum_{t=0}^{N-1} \|\mathbf{z}_t - \mathbf{s}(u_t)\|^2 . \tag{4.8}$$

For splines with two dimensions as used in this approach, we need to expand each $\mathbf{r}_t$ for the construction of the matrix $\mathbf{X}$. We will assume x- and y-components of our data to be alternating in the reference path vector $\mathbf{z}$ and the parameter vector $\mathbf{p}$, i.e.,

$$\mathbf{p} = \begin{pmatrix} p_{0,x}^0 p_{0,y}^0 \dots p_{0,x}^{K-1} p_{0,y}^{K-1} & \dots & p_{M,x}^0 p_{M,y}^0 \dots p_{M,x}^{K-1} p_{M,y}^{K-1} \end{pmatrix}^\top , \tag{4.9}$$

and

$$\mathbf{z} = \begin{pmatrix} z_{0,x} & z_{0,y} & z_{1,x} & z_{1,y} & \dots & z_{N-1,x} & z_{N-1,y} \end{pmatrix} . \tag{4.10}$$

Therefore, the matrix $\mathbf{X}$ contains two rows for each data point and double the columns compared with the one-dimensional variant. The columns alternate the Hermite basis functions with zeros to select the x-component for the first row and the y-component for the second row:

$$\mathbf{r}_t = \begin{pmatrix} 0 \dots 0 & h_s^0 0 h_s^1 0 \dots h_s^{K-1} 0 & h_e^0 0 h_e^1 0 \dots h_e^{K-1} 0 & 0 \dots 0 \\ 0 \dots 0 & 0 h_s^0 0 h_s^1 \dots 0 h_s^{K-1} & 0 h_e^0 0 h_e^1 \dots 0 h_e^{K-1} & 0 \dots 0 \end{pmatrix} . \tag{4.11}$$

This way, for the now two-dimensional spline $\mathbf{s}(u)$ it holds again that

$$\mathbf{s}(u_t) = \begin{pmatrix} s_x(u_t) \\ s_y(u_t) \end{pmatrix} = \mathbf{r}_t\mathbf{p}_t . \tag{4.12}$$

4.1.2 Constraining start and end of the fitted paths

Typically, mobile robot applications require a high degree of accuracy for the start and end positions. To ensure that the fitted path coincides with the reference path at start and

end, we constrain the spline by removing the corresponding model parameters $\mathbf{p}_0^0$ and $\mathbf{p}_M^0$ from $\mathbf{p}$ and the respective first and last columns from $\mathbf{X}$. Instead, we put the locations $\mathbf{z}_0$ and $\mathbf{z}_{N-1}$ in a constant vector $\mathbf{b}$ for a modified spline fit, $\tilde{\mathbf{p}} = \arg\min_{\mathbf{p}} \|\mathbf{z} - (\mathbf{X}\mathbf{p} + \mathbf{b})\|$. The elements of $\mathbf{b}$ depend on the u_t, and have the form

$$\mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{N-1} \end{pmatrix}, \quad \mathbf{b}_t = \delta_{i=0} h_s^0(u_t - i)\mathbf{z}_0 + \delta_{i=M-1} h_e^0(u_t - i)\mathbf{z}_{N-1}, \tag{4.13}$$

where $\delta_C = 1$ if the condition C in the index is true, and zero otherwise. The vector $\mathbf{b}$ is non-zero only for u_t corresponding to the first and last segment. For these segments, it contains the values that were formerly generated by multiplication of $\mathbf{p}_0^0 = \mathbf{z}_0$ and $\mathbf{p}_M^0 = \mathbf{z}_{N-1}$ with the corresponding value of the basis function.

Furthermore, to constrain the start and end orientations to specified values θ_0 and θ_{N-1}, the first derivative $\mathbf{s}'(u)$ at the start and end has to meet the conditions

$$\mathbf{s}'(0) = \alpha_0 \begin{pmatrix} \cos\theta_0 \\ \sin\theta_0 \end{pmatrix}, \quad \mathbf{s}'(M) = \alpha_M \begin{pmatrix} \cos\theta_{N-1} \\ \sin\theta_{N-1} \end{pmatrix}, \tag{4.14}$$

where α_0, α_M are scalar parameters that control the tangent magnitude at start and end. We remove $\mathbf{p}_0^1$ and $\mathbf{p}_M^1$ from the parameter vector $\mathbf{p}$, and replace them by α_0 and α_M. We further remove the columns in $\mathbf{X}$ corresponding to $\mathbf{p}_0^1$ and $\mathbf{p}_M^1$ and replace them with columns containing the coefficients $\delta_{i=0} h_s^1(u_t - i)\mathbf{s}'(0)$ and $\delta_{i=M-1} h_e^1(u_t - i)\mathbf{s}'(M)$ for the first and last column, respectively. The parameter vector now has the form

$$\mathbf{p} = \left(\alpha_0\, \mathbf{p}_0^2\, \mathbf{p}_0^3 \cdots \mathbf{p}_0^{K-1}\, \mathbf{p}_1^0 \cdots \mathbf{p}_1^{K-1} \quad \cdots \quad \mathbf{p}_{M-1}^0 \cdots \mathbf{p}_{M-1}^{K-1}\, \alpha_M\, \mathbf{p}_M^2\, \mathbf{p}_M^3 \cdots \mathbf{p}_M^{K-1}\right)^\top. \tag{4.15}$$

Solving the least squares fit for the modified $\mathbf{X}$, $\mathbf{p}$, and $\mathbf{b}$ yields a spline that obeys the constraints mentioned above. Nevertheless, the accuracy of the fit depends on the number of spline segments as shown in Figure 4.2. Especially in sharp corners and curves with small radii, the errors can be very high as shown in Figure 4.4 (A). In the next sections, we propose improvements over the basic spline fitting to reduce the number of parameters and the fitting error at the same time.

4.2 Non-linear Fits with Our Path Model

In this section, we will use the path model that we introduced in Chapter 2 in the context of optimization-based trajectory generation to fit a parametric representation to the reference path. For two-dimensional paths (without the independent orientation dimension and heuristics), the model reduces the number of free parameters from six to three per control point. We briefly restate the heuristics of the two-dimensional path model here for convenience.

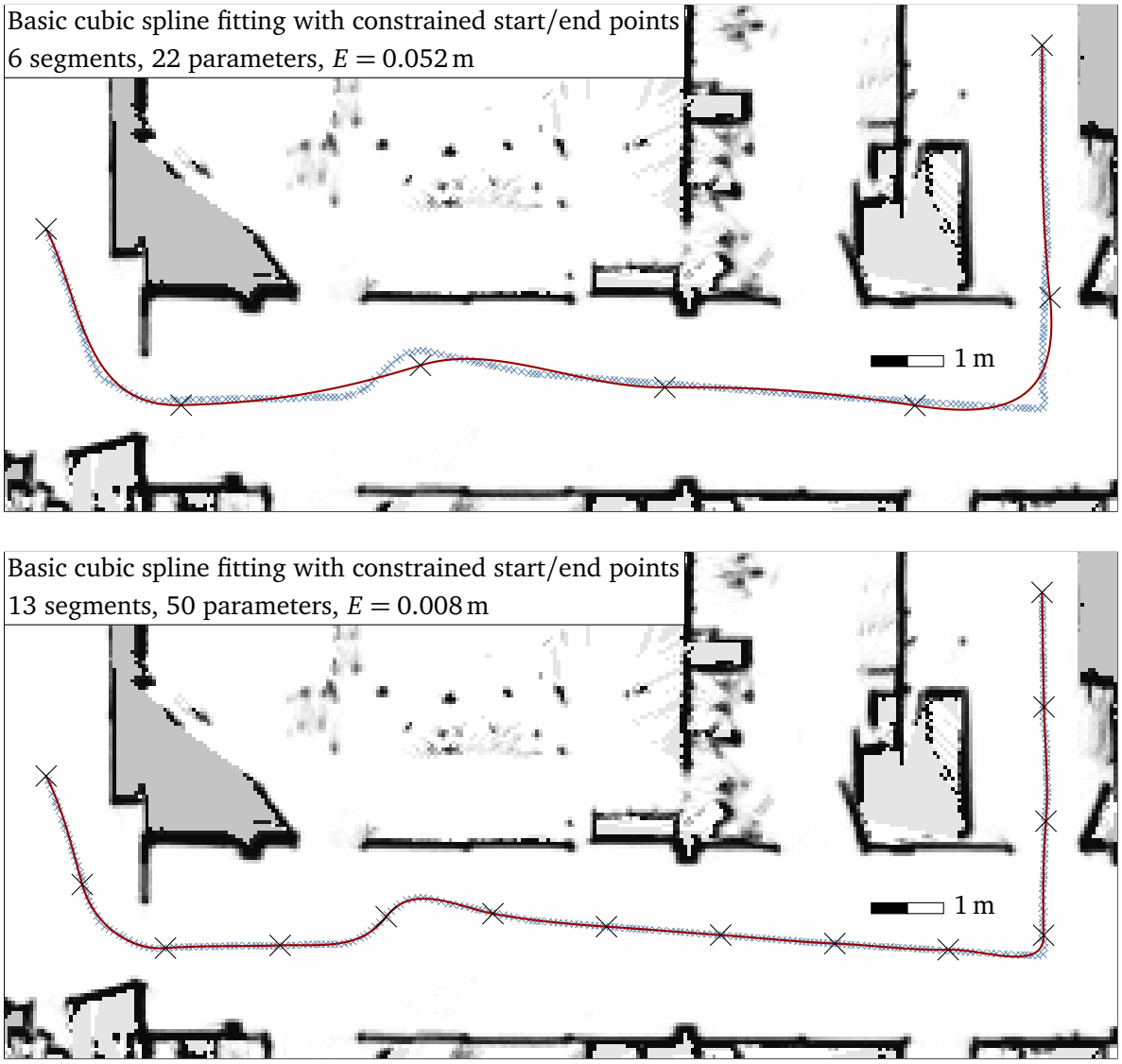


Figure 4.2: Linear least-squares fitting of cubic splines with constrained start/end points. With 22 parameters (top) this overly smoothes the corners, causing a high residual error. With 50 parameters (bottom) the error is comparable to our method with 23 parameters, as shown in Figure 4.1.

4.2.1 Our path model

Similar to the basic splines introduced above, segments in our path model connect a set of consecutive waypoints $\mathbf{p}_i^0$. The model relies on curvature continuous quintic splines, i.e., each segment is a polynomial of degree five. We use a heuristic for the direction of the first derivative, i.e., the tangent of the spline at the waypoints:

$$\mathbf{p}_i^1 = e_i \frac{1}{2}\left(\frac{\mathbf{d}_{i-1}}{\|\mathbf{d}_{i-1}\|} + \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}\right)\frac{1}{2}\min\{\|\mathbf{d}_{i-1}\|, \|\mathbf{d}_i\|\}\,, \tag{4.16}$$

where $\mathbf{d}_i = \mathbf{p}_{i+1}^0 - \mathbf{p}_i^0$ is the vector between the start and end point of segment i. The e_i are scalar parameters that control the magnitude of the tangents at each waypoint, similar to α_0, α_M for the constrained basic spline fitting in the previous section.

The model determines the parameters $\mathbf{p}_i^2$ that specify the second derivative through a heuristic that mimics the behavior of cubic splines. The heuristic sets the second derivative to a weighted average of the (discontinuous) values of corresponding cubic splines at the joint point:

$$\mathbf{p}_i^2 = \frac{\|\mathbf{d}_i\|}{\|\mathbf{d}_{i-1}\| + \|\mathbf{d}_i\|}\lim_{u\to i^-}\mathbf{s}_h''(u) + \frac{\|\mathbf{d}_{i-1}\|}{\|\mathbf{d}_{i-1}\| + \|\mathbf{d}_i\|}\lim_{u\to i^+}\mathbf{s}_h''(u)\,, \tag{4.17}$$

where $\mathbf{s}_h''$ is the piecewise linear second derivative of the cubic spline given by $\mathbf{p}_i^0$ and $\mathbf{p}_i^1$. With these heuristics, a quintic spline is fully specified by the waypoints $\mathbf{p}_i^0$ and the elongation factors e_i. Thus, it has three parameters per control point in the 2D case, whereas a generic cubic spline has four, and a quintic spline has six parameters per control point. We add constraints for the start and end pose in the same way as for the basic splines and name the parameter vector for this model $\mathbf{p}^+$:

$$\mathbf{p}^+ = \begin{pmatrix}\mathbf{p}_1^0 \dots \mathbf{p}_{M-1}^0 & e_0 \dots e_M\end{pmatrix}^\top. \tag{4.18}$$

4.2.2 Fitting with non-linear optimization

To compute splines with our path model, we define a conversion function f, that transforms the parameter vector $\mathbf{p}^+$ to a basic quintic parameter vector according to Eq. (4.16) and (4.17). The least-squares fit with start and end constraints as before is then given by

$$\tilde{\mathbf{p}}^+ = \arg\min_{\mathbf{p}^+}\|\mathbf{z} - \mathbf{X}f(\mathbf{p}^+) + \mathbf{b}\|\,. \tag{4.19}$$

Since f is non-linear, there is no closed form solution to the problem anymore. Instead, we employ optimization using the Levenberg-Marquardt algorithm (Marquardt, 1963). We obtain a good initial guess by computing a linear fit as described in Section 4.1 to initialize the $\mathbf{p}_i^0$ and e_i. We determine the e_i from the $\mathbf{p}_i^1$ by solving Eq. (4.16) accordingly. Figure 4.3 shows an overview of the method so far. Figure 4.4 (B) shows an exemplary output of a fit performed with the method so far. In the following, we present methods to further improve the fit.

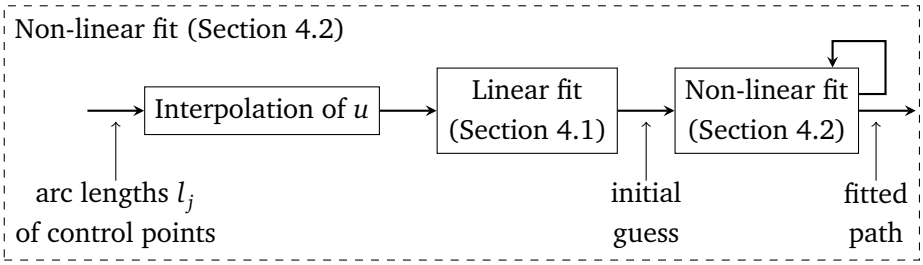


Figure 4.3: Non-linear fit of our path model. After interpolating u, we perform a linear least-squares fit of a basic spline. We use the resulting control points as initial guess for the non-linear optimization of our path model.

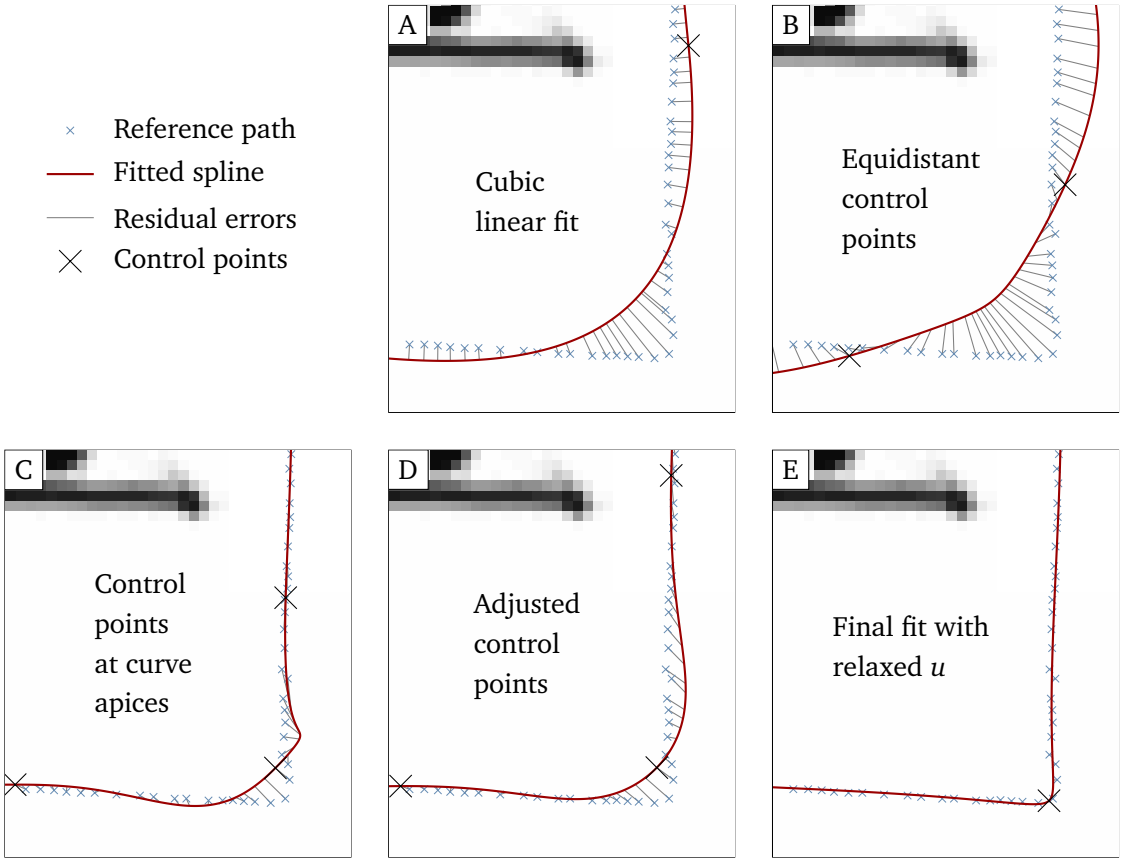


Figure 4.4: Spline fits and residual errors for a given reference path. The plots show the bottom-right corner (1.7 × 1.8 m) of the map in Figure 4.1. (A): linear fit of a basic cubic spline. (B)–(E): We employ several methods to improve the spline fit to the reference path.

4.3 Choosing Control Points

The spline fits in Sections 4.1.1 and 4.2.2 optimize the parameters of the control points $\mathbf{p}_i$. However, to yield low fit errors, their positions are constrained to the vicinity of the demonstration data positions prescribed by the linear interpolation of u for the whole path, see the gray lines in Figure 4.4 (B). This section proposes a method to initialize the control points of our path model at curve apices, which substantially improves the fit quality, see Figure 4.4 (C).

4.3.1 Estimating the location of curve apices

We seek to automatically find curve apices in our training data and denote their location along the reference data with their increasing cumulative arc length l_j, $j = 1, \ldots, J$. To detect these points, we fit a spline $\mathbf{s}_c(u)$ of degree seven to the data as introduced in Section 4.1. We then compute the curvature function $c(u)$, which is the reciprocal value of the curve radius at every point on the spline. The curve apices correspond to extremal values of $c(u)$, which we identify by sign changes in the derivative $c'(u)$, see Figure 4.5. For robustness we perform additional thresholding for a minimum curvature, $|c(u)| > \tau_c$ and we suppress detection of close-by extrema, see for example the non-labeled peak between l_{17} and l_{18} in Figure 4.5. The curvature $c(u)$ and its derivative $c'(u)$ are given by

$$c(u) = \frac{\mathbf{s}_c' \times \mathbf{s}_c''}{\|\mathbf{s}_c'\|^3}, \quad c'(u) = \frac{\mathbf{s}_c' \times \mathbf{s}_c'''}{\|\mathbf{s}_c'\|^3} - \frac{3(\mathbf{s}_c' \times \mathbf{s}_c'')\langle \mathbf{s}_c', \mathbf{s}_c'' \rangle}{\|\mathbf{s}_c'\|^5}, \tag{4.20}$$

where $\mathbf{a} \times \mathbf{b} = a_x b_y - b_x a_y$ and $\langle \mathbf{a}, \mathbf{b} \rangle = a_x b_x + a_y b_y$ for the x and y components of the spline $\mathbf{s}_c(u)$ and its derivatives. We dropped the dependency of $\mathbf{s}_c(u)$ on u for readability. Since $c'(u)$ depends on the third derivative $\mathbf{s}_c'''(u)$, we use a septic spline for $\mathbf{s}_c(u)$, for which $\mathbf{s}_c'''(u)$ is continuous.

We associate the location of each curve apex j with the arc length l_j of the closest point on the piece-wise linear interpolation of the reference data $\mathbf{z}_t$. Here, we treat start and end point like curve apices with $l_0 = 0$ and $l_{J+1} = l_{N-1}$, respectively (see Figure 4.5). We can "anchor" control points of our spline to these l_j by associating them with integer values for u. Then, we interpolate the spline parameter u_t for a data point $\mathbf{z}_t$ at arc length l_t between two anchored control points with indices j, $j+1$ and arc lengths l_j, l_{j+1}:

$$u_t = j + \frac{l_t - l_j}{l_{j+1} - l_j}, \quad \text{with } l_j \leq l_t < l_{j+1}. \tag{4.21}$$

4.3.2 Bayesian Information Criterion for control point selection

Creating control points for all detected curve apices can yield an overly complex spline model. To find a good trade-off between the number of control points and accuracy, we

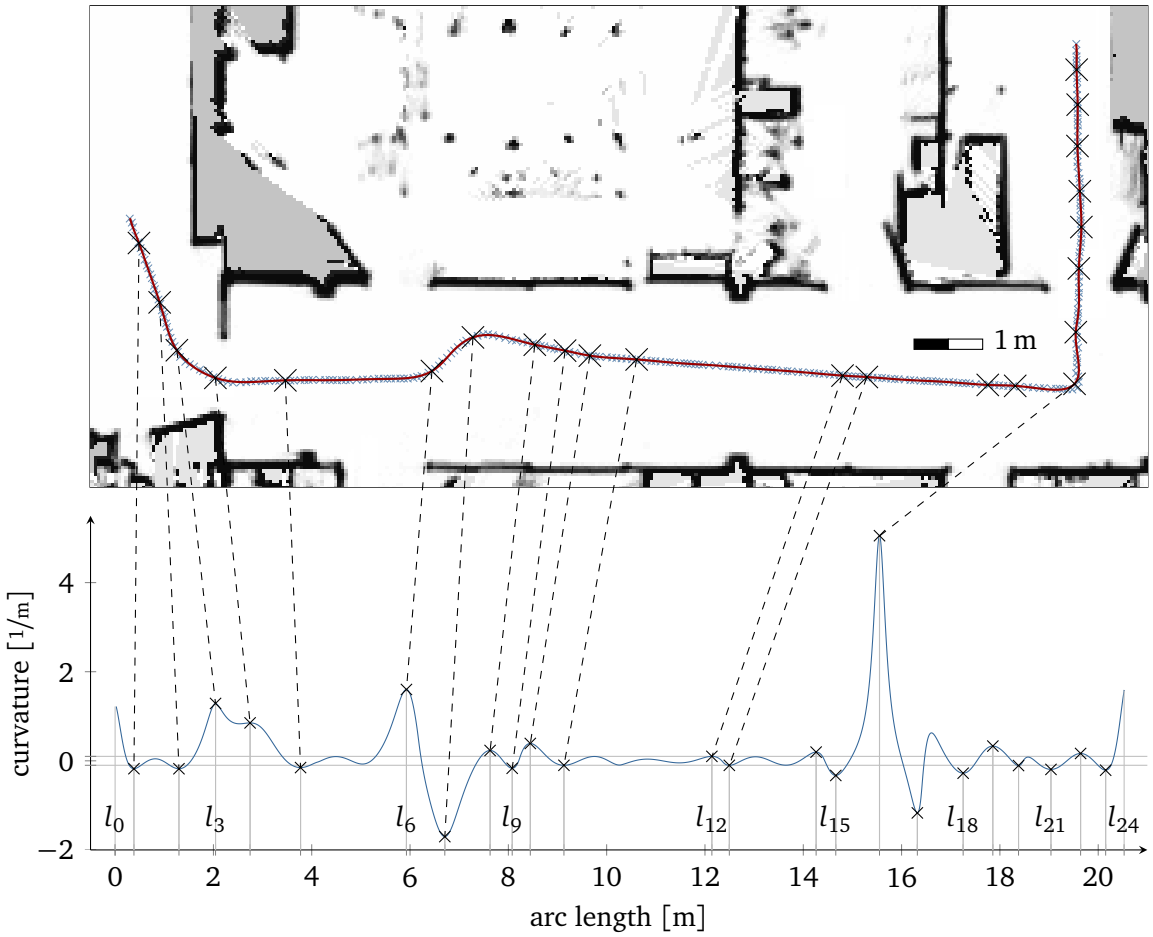


Figure 4.5: A spline of degree seven $\mathbf{s}_c(u)$ was linearly fitted to the demonstration data (top). We compute the curvature of the spline (bottom) to detect curvature extrema that exceed a threshold (crosses) which approximate the position of curve apices along the spline (dashed lines). We represent the extrema with their arc length l_j. For readability, the figure does not label all l_j and correspondences.

propose an error-driven model selection procedure. It builds on the Bayesian Information Criterion (BIC),

$$\mathrm{BIC} = -2\log L + \hat{K}\log N\,, \tag{4.22}$$

where L is the data likelihood given the model, $\hat{K}$ is the number of free parameters in the model, and N the number of data samples. We model the likelihood of a spline fit as a function of the fitting error $r^2 = \sum_{t=0}^{N-1} \|\mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t}\|^2$. It measures the distance from each data point $\mathbf{z}_t$ to the closest point on the spline $\mathbf{s}_{|\mathbf{z}_t}$, which can be efficiently computed as proposed by Schneider (1990). Assuming Gaussian noise and i.i.d. data points, the likelihood is

$$L = \prod_{t=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|\mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t}\|^2}{2\sigma^2}\right) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{r^2}{2\sigma^2}\right). \tag{4.23}$$

The number of free model parameters $\hat{K}$ depends on the number of control points used to model the spline. Since each control point in our model has three parameters and the start and end positions $\mathbf{p}_0^0$ and $\mathbf{p}_M^0$ are given, our model has a complexity of $\hat{K} = 3(M+1) - 4$ parameters. In comparison, the constrained basic 2D splines of order n have $\hat{K} = 2K(M+1) - 6$ parameters, and the unconstrained ones have $\hat{K} = 2K(M+1)$ parameters, with $K = \frac{n+1}{2}$. We use the BIC to choose control points from an initial set with the procedure described in the next section.

4.3.3 Optimization procedure

Using a septic spline fit to find curvature extrema, we obtained a set of arc lengths $\mathscr{L} = \{l_j\}$ at which to potentially place control points. By iteratively applying the following procedure we aim to find a subset $\mathscr{L}^* \subseteq \mathscr{L}$ that minimizes the BIC for the corresponding spline fit. We obtain $\mathscr{L}^*$ by iteratively removing elements from $\mathscr{L}$. In each step, we tentatively remove each element and perform a non-linear spline fit for the remaining control points. The element whose removal improves the BIC the most is permanently removed. The procedure terminates when no removal further improves the BIC. Based on the control point locations in the subset $\mathscr{L}^*$, we refine the spline paths as described in the next section. Figure 4.6 gives an overview of the optimization procedure.

4.4 Refining Spline Fits

All the least-squares techniques for spline fitting described above rely on a fixed correspondence between data points $\mathbf{z}_t$ and internal parameters u_t. This allows to solve the fit in closed form or with a few steps of non-linear optimization, but also limits the expressiveness of the spline. We therefore propose two additional steps to further refine the spline fits. First, we optimize the correspondences, i.e., the position of control points

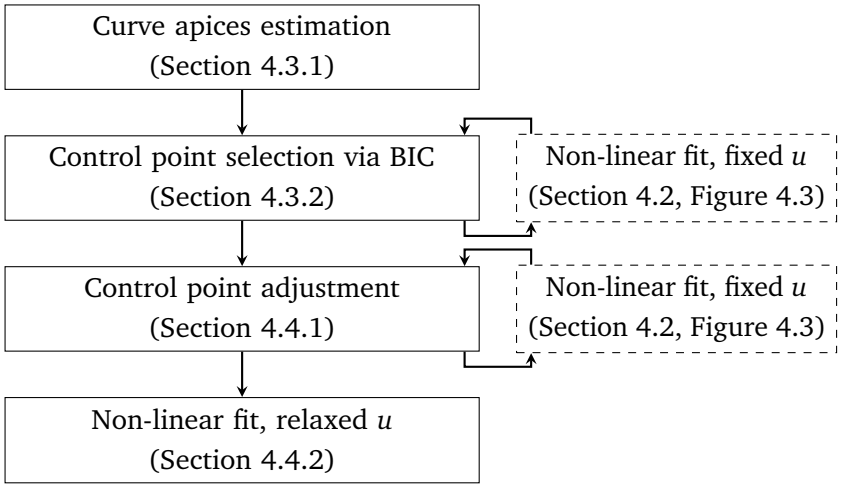


Figure 4.6: Schematic overview of our approach. We describe the individual steps in the indicated sections. For the non-linear fit (dashed) see also Figure 4.3.

along the spline, which improves the spline fit in sharp curves as shown in Figure 4.4 (D). Second, we perform an additional optimization step that relaxes the correspondences to the internal parameter u, which allows the spline to vary its progress in arc length per internal parameter. This way, the spline can use short tangents to achieve accurate fits even in sharp curves as shown in Figure 4.4 (E).

4.4.1 Adjusting control point correspondences

In Section 4.3.1 we defined the set $\mathscr{L}$ of arc lengths along the spline at which we place control points. The method in Section 4.3.3 prunes $\mathscr{L}$ to a subset $\mathscr{L}^*$. By increasing or decreasing the value of a $l_j \in \mathscr{L}^*$, the corresponding control point moves forwards or backwards along the spline. Thereby, we change the correspondence between the points on the spline and the training data points. We employ non-linear optimization using the Levenberg-Marquardt algorithm (Marquardt, 1963) to perform these changes. In every iteration, the optimization adjusts the elements of $\mathscr{L}^*$ and performs new non-linear spline fits to minimize the residual error as shown in Figure 4.6.

We show an example in Figure 4.4 (D), where adjusting the location of the upper control point reduces the tension around the control point in the corner compared with (C).

4.4.2 Relaxing the correspondences to the internal parameter u

In the previous sections, we fitted splines using error measures with fixed correspondences between $\mathbf{s}(u_t)$ and $\mathbf{z}_t$. In this way, the "velocity" of a spline segment, i.e., the arc length per u-interval, remains roughly constant. Relaxing this constraint requires extra effort in

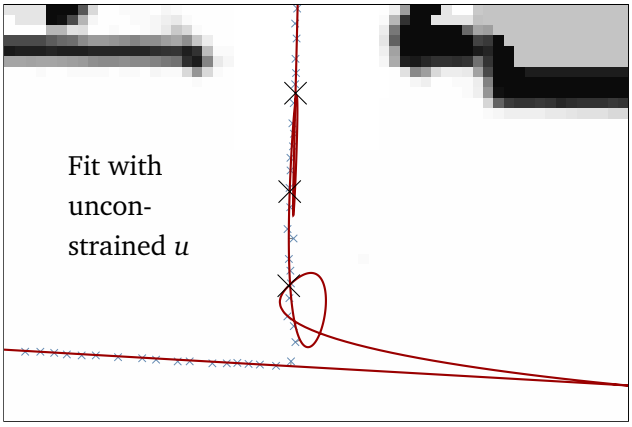


Figure 4.7: Degenerated spline fit when we put no constraints at all on the internal parameter u. The plot shows the same reference data as Figure 4.4.

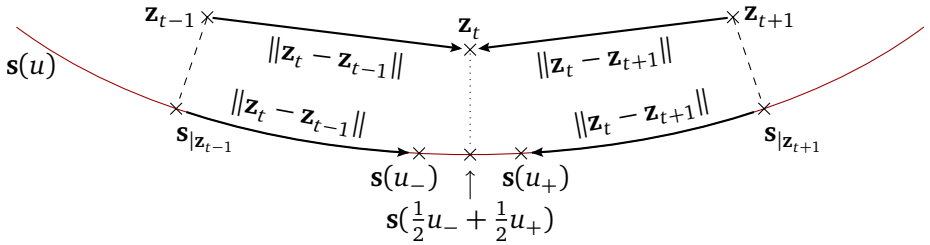


Figure 4.8: Our method for computing the fitting error for $\mathbf{z}_t$. We locate the spline points $\mathbf{s}_{|\mathbf{z}_{t-1}}$, $\mathbf{s}_{|\mathbf{z}_{t+1}}$ closest to the data points $\mathbf{z}_{t-1}$, $\mathbf{z}_{t+1}$. Projecting the arc length between the data points onto the spline yields $\mathbf{s}(u_-)$, $\mathbf{s}(u_+)$. We use their average to determine the error for $\mathbf{z}_t$.

the computation of the fit errors, but allows more accurate spline fits.

A simple error measure for least-squares fitting without u correspondences is the closest distance from each data point to the fitted spline. Without any additional constraints, however, the spline could be close to all data points and still contain deviations and loops without any penalties as shown in Figure 4.7. We propose a novel approach that overcomes this problem.

To compute the fitting error for a data point $\mathbf{z}_t$ and a spline $\mathbf{s}(u)$, we consider the neighboring points $\mathbf{z}_{t-1}$ and $\mathbf{z}_{t+1}$ and the spline points $\mathbf{s}_{|\mathbf{z}_{t-1}}$ and $\mathbf{s}_{|\mathbf{z}_{t+1}}$ closest to them, as illustrated in Figure 4.8. Starting from $\mathbf{s}_{|\mathbf{z}_{t-1}}$ we move the distance $\|\mathbf{z}_t - \mathbf{z}_{t-1}\|$ along the spline to the point denoted by $\mathbf{s}(u_-)$. Similarly, from $\mathbf{s}_{|\mathbf{z}_{t+1}}$ we move to $\mathbf{s}(u_+)$. The points $\mathbf{s}(u_-)$ and $\mathbf{s}(u_+)$ both approximate the point on the spline corresponding to $\mathbf{z}_t$, and we use the average $\mathbf{s}(\frac{1}{2}u_- + \frac{1}{2}u_+)$ to compute the fit error for $\mathbf{z}_t$.

When performing the non-linear spline fitting using this error measure, the spline is not restricted by the correspondences of u and we can fit it to sharp corners with much

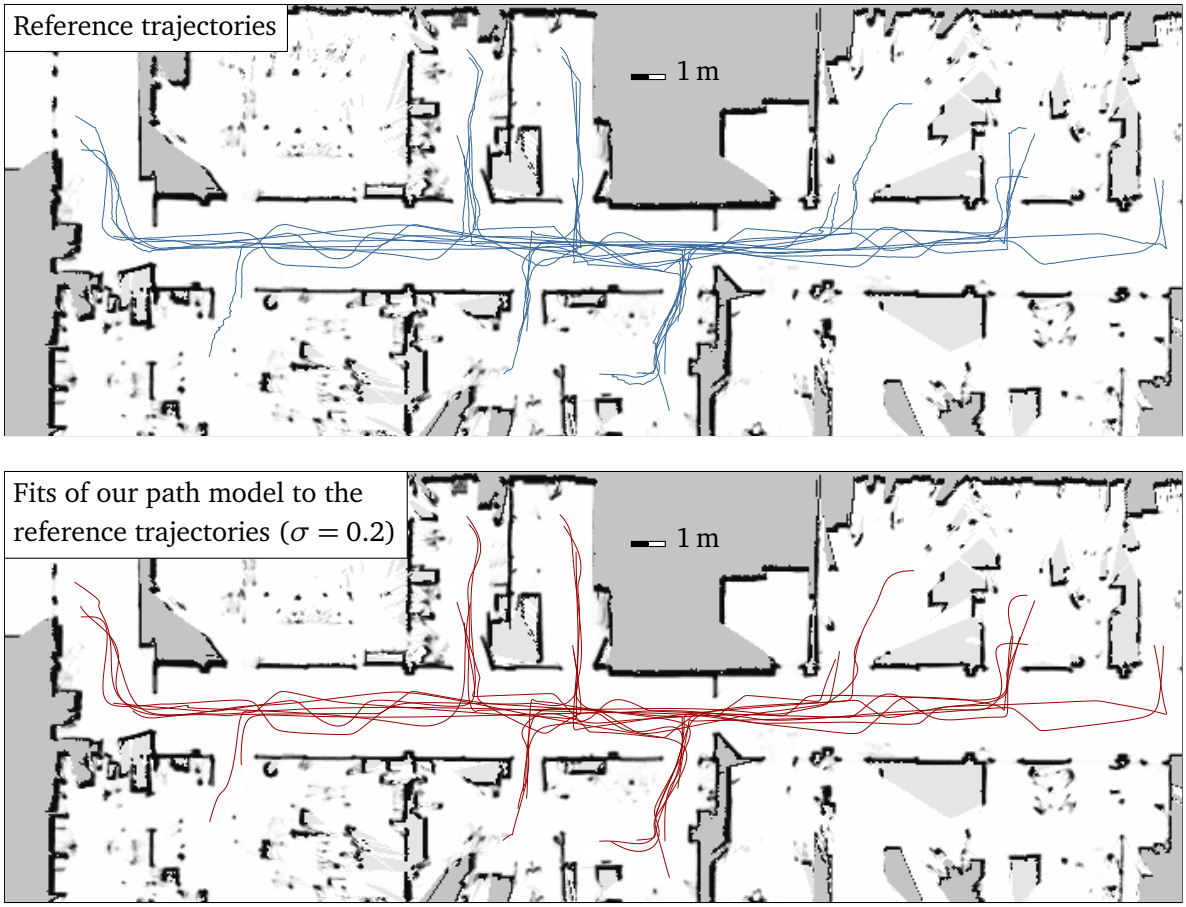


Figure 4.9: The 20 reference trajectories recorded for our experiments (top) and the resulting fits of our path model (bottom).

higher accuracy as shown in Figure 4.4 (E). At the same time, using two neighbors for the closest point search effectively suppresses the degeneration of the fitted spline.

4.5 Experiments

To evaluate our approach we recorded 20 trajectories with a Pioneer P3-DX robot driven by joystick in an office building, as shown in Figure 4.9 (top). As described in Section 4.1.1, we pruned the data with a minimum distance threshold $\tau_l = 0.05\,\text{m}$, which corresponds to the map resolution of the global robot self-localization.

We identified candidate control point locations by the curvature of a septic spline with 0.5 segments per meter using a threshold of $\tau_c = 0.1\,{}^1/_\text{m}$, see Section 4.3.1. These values are appropriate for paths in human environments, but can easily be scaled to miniature or large-size robots.

We fitted our path model to all recorded trajectories, see Figure 4.9 (bottom). De-

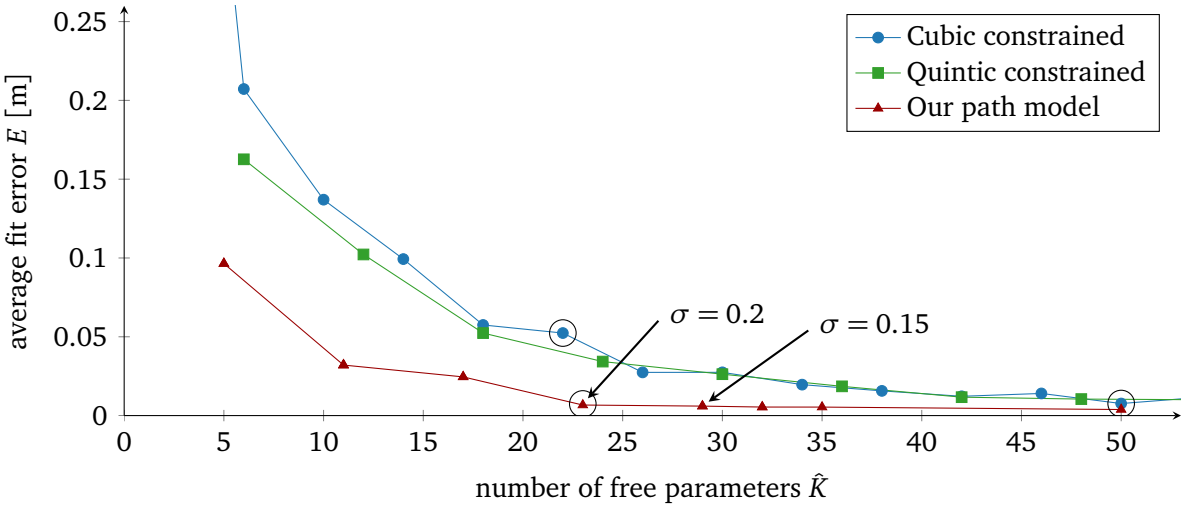


Figure 4.10: Residual fitting errors for varying model complexity of the compared approaches. The splines fits used the reference trajectory shown in Figures 4.1 and 4.2. These figures also show the fits for the combinations marked in the plot (black circles).

pending on the value for σ in Eq. (4.23), our method balances the number of model parameters with the residual fit error. For comparison, we computed constrained and unconstrained linear least-squares fits of cubic and quintic splines (see Section 4.1.1). Here, we used different numbers of segments to achieve different trade-offs between the number of parameters and fit quality.

All of the fitted trajectories were appropriate spline fits and did not suffer from degenerate extra loops like the one shown in Figure 4.7. To compare the fit quantitatively across trajectories and approaches, we computed the fitting error as average distance from the data points to the fitted path:

$$E = \frac{1}{N} \sum_{t=0}^{N-1} \| \mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t} \| \,. \tag{4.24}$$

To efficiently compute the distance $\mathbf{s}_{|\mathbf{z}_t}$ of a point $\mathbf{z}_t$ to the path we employ the algorithm suggested by Schneider (1990).

Figure 4.10 shows errors of different fits to the reference trajectory shown in Figure 4.1. As expected, a higher number of parameters leads to lower errors for all approaches. For our application, $\sigma = 0.15$ or $\sigma = 0.2$ yields a good compromise. In all cases, our approach achieves a lower error for a comparable number of parameters and needs fewer parameters for comparable errors. The biggest improvements occur in sharp corners, see Figure 4.4 (E) vs. (A). The results for the remaining reference trajectories are similar.

Figure 4.11 shows the average and maximum fit error per data point over all trajectories. For the baseline approaches, we also computed the BIC for different numbers of control

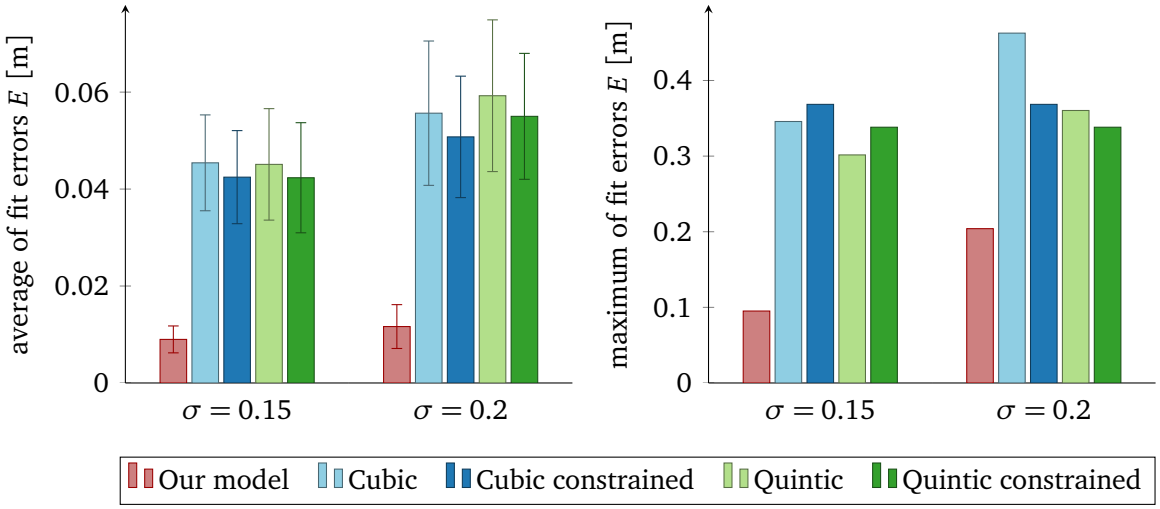


Figure 4.11: Average (left) and maximum (right) residual fitting errors across the 20 trajectories used in the experiments (see Figure 4.9). We selected the linear fits that minimize the BIC for the indicated value of σ.

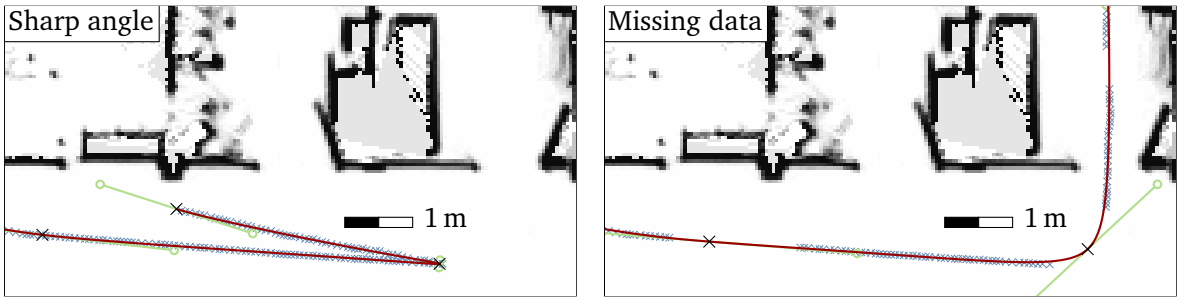


Figure 4.12: Our placement of control points and the resulting spline fits are robust to very sharp angles and missing values in the reference data.

points, and for each trajectory selected the number that minimized the BIC. The fitted curves are therefore the optimal balance between fit error and model complexity for each approach and a given σ. The plot shows that our approach generates substantially lower average and maximum fit errors. The baseline approaches have no significant difference in fit quality for cubic vs. quintic and constrained vs. unconstrained ones, since the BIC selection uses fewer control points for models with more parameters per point.

Figure 4.12 demonstrates the robustness of our approach in challenging situations. Neither extreme directional changes nor a considerable amount of missing data deteriorate our spline fits. With increasing values for σ, the fits are robust against noise in the reference data. Extreme outliers could be filtered out or compensated with robust statistics in the least-squares fits.

Naturally, the computation time for a spline fit depends on the size of the input data. A

crucial factor is the number of curvature maxima used in the combinatorial control point selection, see Section 4.3.

4.6 Applications

As shown in the experiments, our approach generates accurate spline fits with a small number of parameters. The approach uses the path model introduced in Chapter 2. In this section, we present how to employ the optimization proposed in Chapter 2 to improve the shape of demonstrated trajectories. Furthermore, we show how to apply the optimization to robustly follow the fitted reference trajectories generated by the approach proposed in this chapter.

4.6.1 Optimization of the fitted path

After fitting the path model, a robot can follow the path using ad-hoc velocities set by a reactive controller. One can also compute a velocity profile for the path using the method that we proposed earlier in the context of trajectory optimization in Section 2.3.3. This minimizes the traversal time by maximizing the translational velocity, while obeying a set of constraints, e.g., maximum speed and acceleration of the hardware platform, or a bound on the centripetal force. In this case, the path shape remains unchanged. Additionally, one can also employ the optimization procedure in from Chapter 2 to optimize the path shape as well, potentially with user-specified bounds on the allowed deviation from the reference path. The small number of parameters in the fitted trajectory makes this problem computationally feasible. In this way, one can optimize a suboptimal shape of the reference trajectory to yield faster travel times as well, while retaining the topology of the path and avoiding collisions with mapped obstacles.

Figure 4.13 shows the result when applying the trajectory optimization proposed in Chapter 2 to the final fit. The figure also shows the reference path. One can observe how the optimization removed an unnecessary curve and how it widened the sharp curve in the right part of the figure. Thereby, the trajectory optimization was able to reduce the travel time for this trajectory from 37.3 s to 30.6 s in an optimization time of 0.9 s.

4.6.2 Robustly following a fitted trajectory

To be of practical use in industrial application scenarios like virtual rails, the reference trajectories fitted in this chapter need to be defined in a global reference frame. Global localization, however, is often noisy and only available at a relatively low frequency. Therefore, one typically resorts to the odometry frame of the robot to accurately follow a planned trajectory as also done in Chapter 2. There, we transform the trajectories from

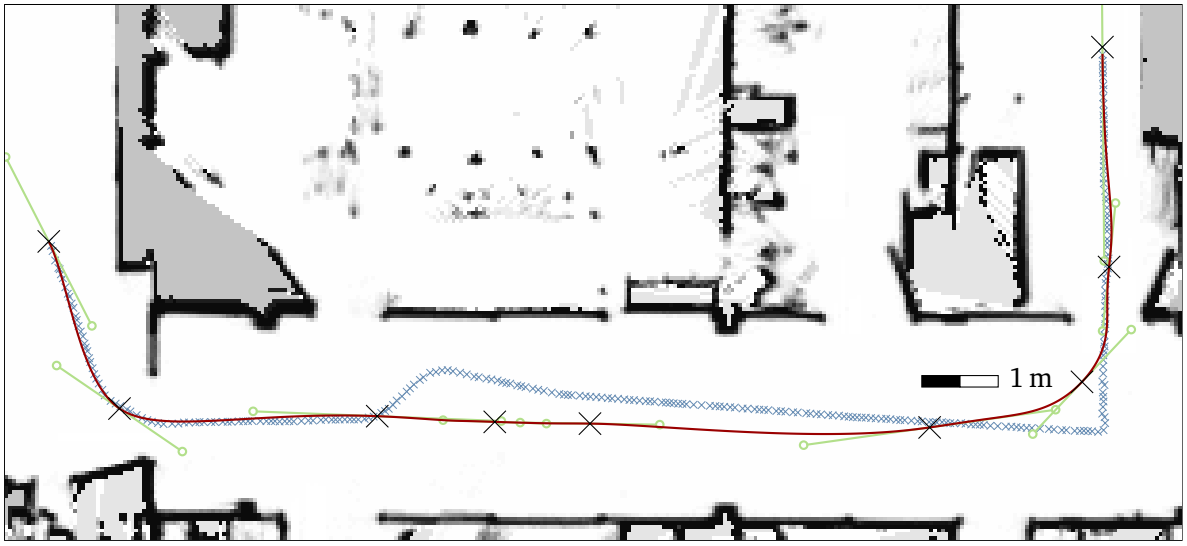


Figure 4.13: Optimizing the final fit of our approach (see Figure 4.1) with the trajectory optimization proposed in Chapter 2. The figure shows the optimized trajectory (red) together with the original reference path (blue crosses).

the global frame into the odometry frame of the robot. As the pose estimates by the wheel encoders of a robot are subject to accumulating drift and to cope with localization errors we proposed regular replanning and updating of trajectories in the global frame in Chapter 2. Here, we present an analogous method to apply this sort of replanning to the execution of the fitted reference trajectories proposed in this chapter.

Suppose that the robot receives an updated estimate of its pose in the global reference frame. Due to execution error and noise in the localization, this pose will be different from the one prescribed by the reference trajectory for this point in time. Now we want to compute a path that drives the robot back to the reference trajectory as fast as possible but at the same time respects the kinodynamic constraints of the platform. To balance these two objectives we will employ the optimization procedure proposed in Chapter 2 with a different set of parameters that determine the point at which the robot will return to the reference trajectory and the shape of the connecting segment.

We create a new spline segment $\mathbf{s}_{\text{new}}$ that connects the current robot pose to the spline of the reference trajectory $\mathbf{s}(u)$. We set the parameters for the start of $\mathbf{s}_{\text{new}}$ to match the current heading and curvature of the robot. To achieve a smooth join to the reference trajectory at a given point $\mathbf{s}(u_d)$ on $\mathbf{s}(u)$, we subdivide the segment of $\mathbf{s}(u)$ that is active at u_d by inserting an extra control point $\mathbf{p}_d$ at u_d. We determine its parameters $\mathbf{p}_d^k$ by evaluating the old segment and its derivatives at that point. To account for the new length of the subdivided segment we need to rescale the obtained values for $\mathbf{p}_d^k$. As we presented in more detail in Section 2.3.2, one can this way subdivide polynomial spline segments without affecting their shape.

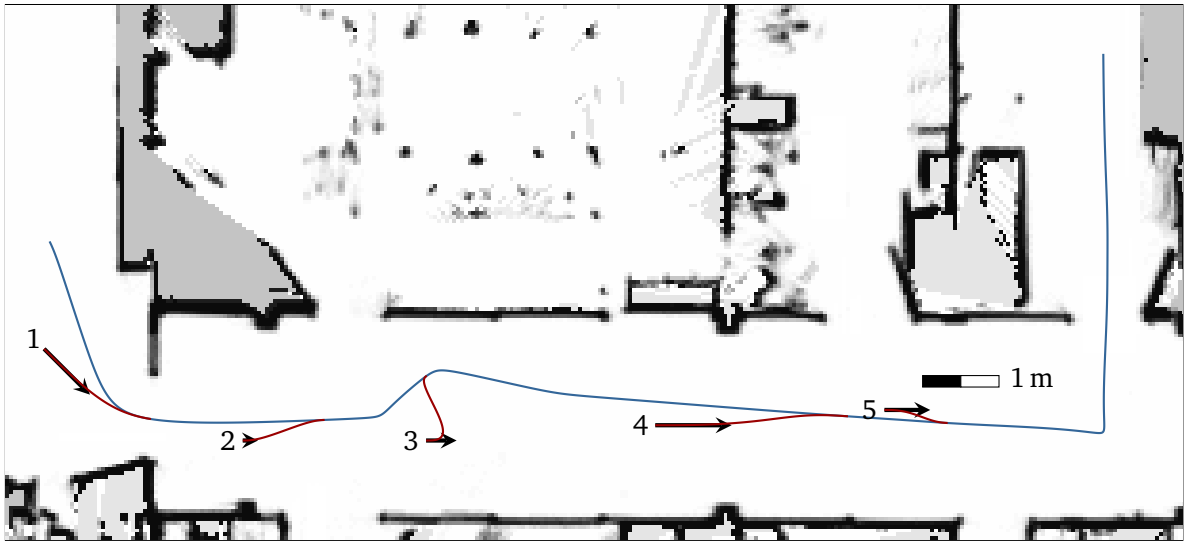


Figure 4.14: Using an adapted version of the optimization from Chapter 2 to robustly follow a reference trajectory fitted by the approach presented in this chapter (blue). The figure shows the connection segments (red) for five different start poses, where the arrows indicate the robot's current orientation and velocity. The reference trajectory is the final fit from Figure 4.1 and travels from left to right.

Finally, we can optimize the shape of the new segment $\mathbf{s}_{\text{new}}$ and the location of the join point u_d. We do this by employing the optimization procedure from Chapter 2 using as cost function the time of travel until connection to the reference trajectory. Free parameters for the optimization are the magnitude of the tangents at the start and end of $\mathbf{s}_{\text{new}}$ as well as the join point u_d to the reference trajectory. To give the optimization more possibilities we drop the sharing of first and second derivative between the adjacent segments at the join point. To nevertheless ensure a smooth, curvature continuous join of the segment $\mathbf{s}_{\text{new}}$ with the reference trajectory $\mathbf{s}(u)$, the orientation of the first derivative at this point does not use the heuristic from the path model but stays fixed to the one given by $\mathbf{s}'(u_d)$. Also, whenever we scale the magnitude of the first derivative of $\mathbf{s}_{\text{new}}$ at $\mathbf{s}(u_d)$ by $a \in \mathbb{R}$, we need to scale the second derivative by a^2 to ensure a curvature continuous join.

Figure 4.14 shows a result of planning back to the reference trajectory defined by the final fit (blue curve) from five different deviating robot poses. To demonstrate the robustness of the approach and for better readability, robot poses 1–4 correspond to unrealistically large deviations from the reference trajectory. One can observe how the optimization takes into account the current velocity and orientation of the robot (arrows) when computing connection segments that respect the kinodynamic constraints of the platform.

4.7 Related Work

Programming by demonstration or more generally learning from demonstration is an area that has received considerable attention, see for example the surveys by Billard et al. (2008) and Argall et al. (2009). The main application scenario in industry is a basic form of programming by demonstration. Here, an operator moves the robotic arm to several key configurations and then uses them to compose a program for the robot (Lozano-Perez, 1983). While at first the operator had to move the robot by directly controlling its joints, over the years more intuitive methods like controlling the position and orientation of the end effector in the workspace with a 6D mouse have become available. For even more intuitive teaching, 6D force sensors exist that one can mount at the end effector of the robot, allowing the operator to "take the robot by the hand" and move it to the key configurations. A new generation of light-weight robotic arms can register forces in each joint and therefore compensate external forces nearly anywhere on the robotic arm. This enables the operator to push and pull the robot into the desired configurations with little effort (Grunwald et al., 2003). In this chapter, we presented an approach that applies the programming by demonstration paradigm to a mobile robot instead of a stationary robotic arm.

In contrast to programming exact repetitions of the demonstrated behavior, there are also approaches that strive to generalize from multiple demonstrations of the desired behavior. Calinon et al. (2010) for example combine Hidden Markov Models with Gaussian mixture regression to generalize from multiple gesture demonstrations. Abbeel et al. (2007) record multiple human demonstrations of aerobatic maneuvers with a remote controlled helicopter. They then apply reinforcement learning to infer a controller that outperforms the human teacher. For the approach presented in this chapter we rely on only one demonstration with the goal of representing it as accurately as possible with some robustness to noise. We presented applications, however, in which we can improve on the human demonstration by computing a velocity profile and by applying the trajectory optimization from Chapter 2, see Section 4.6.

Several authors have applied spline fitting to generate smooth trajectories from discrete reference data points. Billard et al. (2004) fit cubic splines to interpolate trajectories defined by waypoints in a manipulation imitation approach. Aleotti and Caselli (2006) use Non-Uniform Rational B-Splines to approximate trajectories of human hands that they demonstrate through a data glove and a tracking device. Ude (1993) employs splines to compute a continuous approximation of the path of an object in a robot's visual system in the context of teaching by demonstration.

Splines are also used to analyze human motion outside of the programming by demonstration paradigm. Lee (2004) use fourth-order polynomials to perform spline fitting to both position and velocity data of handwriting motions, and Baiget et al. (2008) sample B-splines to analyze human trajectories. Several approaches exist that use splines to

represent the trajectories of mobile robots. We discussed this body of related work in the chapter on our navigation system, in the context of accurate and efficient spline-based robot motion, see Section 2.6.

Basic spline fitting relies on linear least-squares minimization given fixed correspondences between the reference data points and the internal parameter of the spline, which drastically limits the expressiveness of the spline. Wang et al. (2006) presented an error measure to fit B-splines to point cloud data without such fixed correspondences. Their error measure sums the distance from each data point to the spline curve, and applies regularization on the control points. This is problematic for sparse control points as in our application. Therefore, we proposed a novel error measure especially suited for non-linear spline-fitting with sparse control points (see Section 4.4.2).

Hwang et al. (2003) employed the basic spline fitting that we used for baseline comparison for hand drawn robot paths. Like the approach by Macfarlane and Croft (2001), our model uses quintic splines to avoid curvature discontinuities.

In this chapter, we presented an approach to non-linear spline fitting of a specific path model. Compared with standard approaches, our model requires substantially fewer parameters to achieve the same accuracy.

4.8 Conclusion

In this chapter, we presented an approach to robustly fit parametric mobile robot paths to reference trajectories demonstrated by a user. Our method uses the specific path model introduced in Chapter 2 and thereby needs fewer parameters than standard approaches to achieve similar approximation results. We employ the Bayesian Information Criterion in the optimization procedure to calculate the best trade-off between model complexity and accuracy. The experiments carried out on real-world data show that our approach clearly outperforms basic spline fitting methods.

We believe that the presented approach allows intuitive and flexible teaching of robot paths. We have shown that it supports several applications: It is possible to augment fitted paths with time-efficient velocity profiles and to further optimize them to minimize the time of travel by leveraging the methods introduced in Chapter 2. A further application of the trajectory optimization from Chapter 2 enables the execution of the fitted reference trajectories in a global reference frame to become robust against localization and execution errors. Thereby, the proposed method becomes applicable for virtual rails in industrial scenarios that require the robot to accurately follow predefined paths. The combination of the virtual rails presented in this chapter with the fully autonomous approach from Chapter 2 allows to cover multiple levels of autonomy. In addition to full autonomy and strict following of a predefined trajectory, the robot could leave the predefined trajectory if it encounters an obstacle instead of waiting for it to disappear. By selecting as goal a

point on the virtual rail beyond the encountered obstacle one can use the system proposed in Chapter 2 to plan a path around the obstacle in a bounded area. Through appropriate orchestration of the two methods one can deploy a system with the suitable degree of autonomy for the scenario at hand.

The approach presented in this chapter removes the need for external guidance infrastructure in application scenarios in which automated vehicles have to strictly follow virtual rails. While the programming by demonstration paradigm allows intuitive tasking of such systems by non-specialized shop floor workers, the approach still relies on the localization of the robot in a previously built map of the environment. While such a map is anyway needed if robots are to also navigate autonomously in the environment, the overhead and domain-specific knowledge required to build such a map still constitutes a significant burden for scenarios that only require vehicles to follow virtual rails. Therefore, in the next chapter we present a method that enables robots to repeat a demonstrated trajectory without requiring the user to previously build a map of the environment.

5 A Teach-and-Repeat Framework for Accurate Navigation

The previous chapter introduced the concept of virtual rails that users teach to mobile robots through demonstration. This concept enables intuitive and easy-to-use systems that allow non-expert shop floor workers to intuitively instruct transportation systems. Easy-to-instruct systems are a key requirement for the automation of logistics tasks for small batch sizes and flexible production processes. In this chapter, we further reduce the setup time for virtual rails by removing the requirement of a pre-built, globally consistent metric map to localize the robot. To this end, we present a novel laser-based scheme for teach-and-repeat of mobile robot trajectories. Our system relies on scan matching to localize the robot relative to a taught trajectory, which is represented by a sequence of raw odometry and 2D laser data. In addition to the reduced setup time, the direct use of raw sensor data avoids additional errors that might be introduced by the discretization of grid maps. Real-world experiments carried out with a holonomic and a differential drive platform demonstrate that our approach repeats trajectories with an accuracy of a few millimeters. A comparison with a standard localization approach on globally consistent grid maps furthermore reveals that our method yields lower tracking errors for teach-and-repeat tasks. We furthermore present an extension of the framework to optimize user demonstrations and an application in the context of repeatable experiments.

As discussed in the previous chapter, a large number of application scenarios require the robot to strictly follow a predefined route through the environment. The previous chapter followed the programming by demonstration paradigm to enable non-specialized shop floor workers to define such virtual rails. This makes it possible to discard the infrastructure overhead of traditional automated guided vehicles (AGVs) and to thereby increase the flexibility of automated logistics. However, while the instruction phase of the previously introduced method is intuitive, it still requires a globally consistent, metric map of the environment to localize the robot during the user demonstration. This map needs to be built beforehand, a time-consuming process typically conducted by manually driving the robot through the environment and verifying the result of a simultaneous

localization and mapping (SLAM) technique with expert knowledge.

In this chapter, we continue to pursue the teaching by demonstration paradigm and further reduce the setup time and increase the usability for virtual rail scenarios. We present an approach that records all necessary information during the user demonstration itself, thereby abolishing the previously needed mapping phase. We target the system proposed in this chapter towards a natural *one-button* approach that lets the user manually control the robot to demonstrate a new trajectory that is then reproduced with high accuracy and precision. To this end, we follow the teach-and-repeat paradigm from the family of programming by demonstration approaches: During a preliminary *teach* phase, a human operator controls the robot motion while the robot acquires a sequence of sensor data and stores it in a database. Then, during the *repeat* phase, the robot repeats the same path, by comparing the currently observed and the previously recorded sensor data.

Our method represents user-taught trajectories by anchor points consisting of raw odometry and 2D laser data. Similar to the localization at target positions introduced in Chapter 3, it applies a scan matching routine to estimate the actual offset of the current robot position from these anchor points. By comparing the reference offset with the actual offset, our algorithm calculates the error in the configuration space of the robot, which is then used for any desired feedback controller, e.g., a linear quadratic regulator or dynamic feedback linearization.

The method presented in this chapter has an interpretation as extension of the localization and positioning method from Chapter 3 towards time-varying reference configurations. As such, a further application scenario for the proposed method becomes apparent: In industrial automation there are use cases in which a mobile robot has to perform specific docking maneuvers to integrate its task into a process or logistics chain. Often, the specifications for such maneuvers exceed a simple pose configuration but rather require a specific sequence of motions within tight constraints and high accuracy requirements. For these situations, the method presented here can also serve as replacement for the pose-based docking method from Chapter 3, extending the applicability of the autonomous navigation system available through the approaches presented in this thesis so far. By handing over to a docking module based on the method presented in this chapter, the navigation system can connect to process chains that require more complex docking maneuvers. High accuracy and intuitive use are the key properties for our teach-and-repeat framework to be of benefit in both use cases, end to end navigation as well as docking maneuvers. We demonstrate in real-world experiments that with our teach-and-repeat framework, robots can repeat taught trajectories with high accuracy and more accurately than with standard localization on a globally consistent metric grid map. Figure 5.1 shows a time-lapsed view of one of our experiments.

Similar to the optimization applied to the fitted paths in the previous chapter, we also present an extension to our framework to optimize the user-demonstrated trajectories with respect to smoothness and speed and within given, non-stationary bounds for the

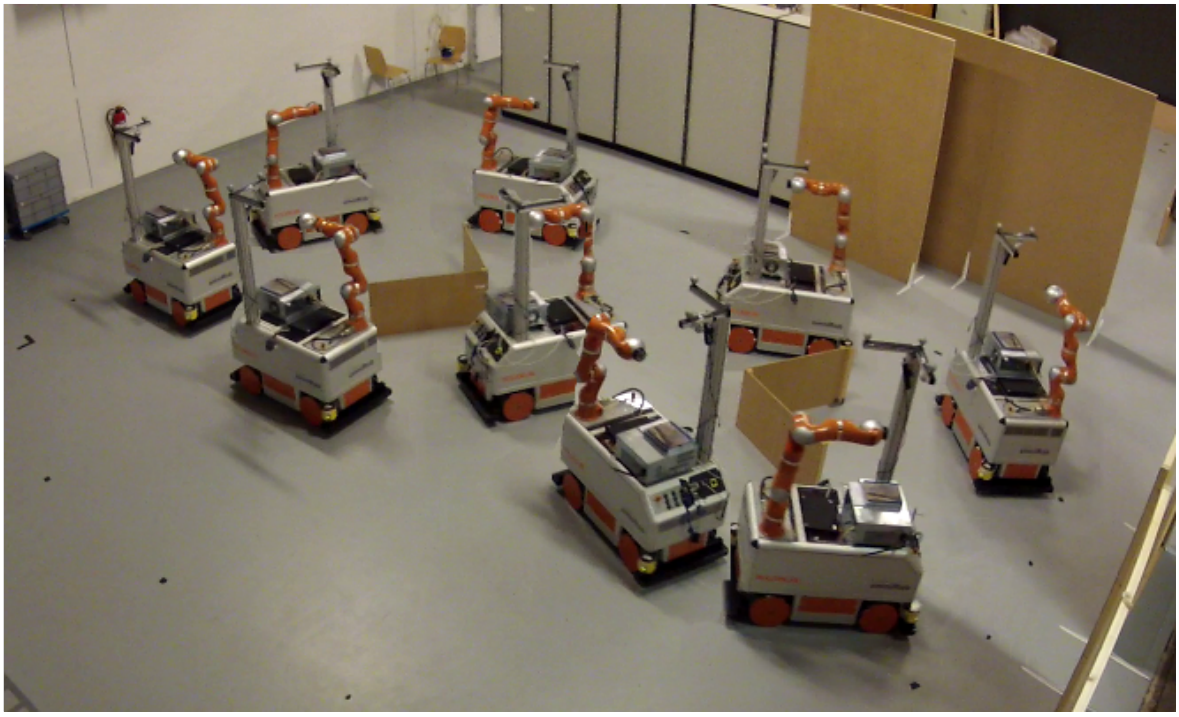


Figure 5.1: Time-lapsed view of the holonomic omniRob robot reproducing the user-taught reference trajectory FigureEight with our approach.

deviation from the original demonstration. As a further use case for our framework we also present an application in the context of experiments in which it is important to reproduce trajectories as accurately as possible to establish comparability.

The remainder of this chapter is organized as follows. First, we describe the task tackled in this chapter more formally in Section 5.1. We then present our approach in Section 5.2. In Section 5.3 we report our experimental evaluation. We propose an extension to the presented framework in Section 5.4 that introduces a methodology to optimize the user demonstration before repetition. Section 5.5 presents the application of our framework to repeatable experiments with mobile robots. Before we conclude the chapter in Section 5.7, we discuss related works in Section 5.6.

5.1 Problem Definition

The objective of our approach is to repeat, with a wheeled mobile robot and no previously built map, a time trajectory previously taught by user demonstration. We assume the ground to be planar, and the robot may be either holonomic or subject to nonholonomic constraints (e.g., a differential drive robot), and it is equipped with a laser scanner mounted parallel to the ground.

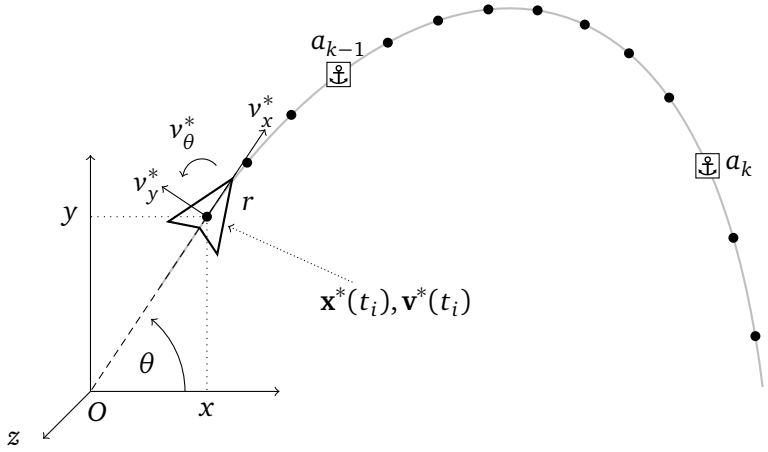


Figure 5.2: During the teaching phase, we record the velocities $\mathbf{v}^*(t_i)$ at discrete time steps t_i, and use them to derive the taught trajectory poses $\mathbf{x}^*(t_i)$ shown as black dots. Along this trajectory, we insert anchor points a_k at which we also record laser range finder measurements. This schematic figure also illustrates the used coordinate systems.

We name r the robot center of rotation, which should track the trajectory. With reference to Figure 5.2, we define the world frame $\mathcal{F}\left(O, x, y, z\right)$. The robot configuration is:

$$\mathbf{x} = \left[x_x, x_y, x_\theta\right]^\top \in SE(2), \tag{5.1}$$

where x_x and x_y represent the Cartesian position of r in $\mathcal{F}$, and $x_\theta \in (-\pi, \pi]$ is the positive counterclockwise orientation of the robot with respect to the x-axis. The control inputs are:

$$\mathbf{v} = \left[v_x, v_y, v_\theta\right]^\top \in \mathbb{R}^3. \tag{5.2}$$

These are respectively the longitudinal translational, lateral translational, and rotational velocities, with sign conventions as shown in Figure 5.2. For a differential drive robot, v_y equals zero.

The goal of this work is to drive the robot configuration $\mathbf{x}(t)$ at time $t \in [0, T]$ to the taught trajectory $\mathbf{x}^*(t)$. We define this trajectory using the recorded robot odometry, as explained below. To realize this task, we rely on two sensors:

Wheel encoders: Measurements of the revolutions of the robot's wheels are available at discrete timesteps t_i ($t_0 = 0$). We estimate the robot velocities $\mathbf{v}$ by differentiating the encoder readings.

Laser scanner: The robot on-board laser scanner provides, at discrete timesteps t_j, a local 2D scan of the environment surrounding the robot.

The robot velocities estimated from the wheel encoders can be used to localize the robot at any time $t \in \left(t_i, t_{i+1}\right]$, employing the well known odometry equation (Siciliano et al., 2009)

$$\mathbf{x}(t) = \mathbf{x}(t_i) \oplus \mathbf{R}_z(\theta(t_i))\,\mathbf{v}(t_i)\left(t - t_i\right), \tag{5.3}$$

with $\mathbf{R}_z$ the rotation matrix about the z axis, and $\oplus$ the compounding operator (Smith et al., 1990). During the teach phase, we record estimates of the robot's velocities. By plugging these recorded velocities $\mathbf{v}^*(0), \mathbf{v}^*(t_1), \mathbf{v}^*(t_2), \ldots, \mathbf{v}^*(T)$, and the initial pose $\mathbf{x}(0)$ into Eq. (5.3), we obtain the taught trajectory $\mathbf{x}^*(t)$ to track. Although it has piecewise constant velocity, this trajectory will provide a sufficiently accurate estimation of the real user-taught one, if the encoder frequency is sufficiently high. Figure 5.2 contains a schematic illustration of a recorded reference trajectory, in which the black dots correspond to the poses $\mathbf{x}^*(t_i)$ at timesteps t_i.

Our approach uses the local 2D scans of the environment in conjunction with a scan matcher to provide relative estimates of the robot's pose. Note that in our approach, those estimates are not integrated and therefore not prone to accumulating drift. During the teach phase, we record scanner readings at regular intervals along the trajectory: We record a new scan as soon as the distance covered since the last one exceeds a fixed threshold $\tau_l > 0$ or the angular displacement exceeds $\tau_\theta > 0$. The rationale behind this is that we want to limit the required memory resources while uniformly scanning the environment (e.g., avoid multiple scan recordings with a standing robot). Based on these scans, we construct the *anchor points* that we will use in the repeat phase. An anchor point $a = (l, \mathbf{x}_a^*)$ consists of a laser scan l with the associated robot pose $\mathbf{x}_a^*$, computed via Eq. (5.3). Figure 5.2 shows two anchor points along a recorded trajectory. During the repeat phase we use all available laser scans l_j. We match them against the scans associated with the anchor points to compute the feedback error.

5.2 Repeating the Taught Trajectory

In this section, we explain how we compute the error signal and how we use it in our trajectory tracking controller. The proposed task is to regulate to zero, at all times, the error between the current and the taught reference pose

$$\mathbf{e}(t) = \mathbf{x}(t) \ominus \mathbf{x}^*(t), \tag{5.4}$$

where $\ominus$ is the reverse compounding operator (Smith et al., 1990).

The computation of this error signal requires precise localization of the robot at time t, i.e., a precise estimate of $\mathbf{x}(t)$. However, as mentioned before, one of the main objectives of our approach is the realization of an intuitive map-free navigation procedure. An alternative to map-based localization consists of estimating $\mathbf{x}(t)$ from odometry alone,

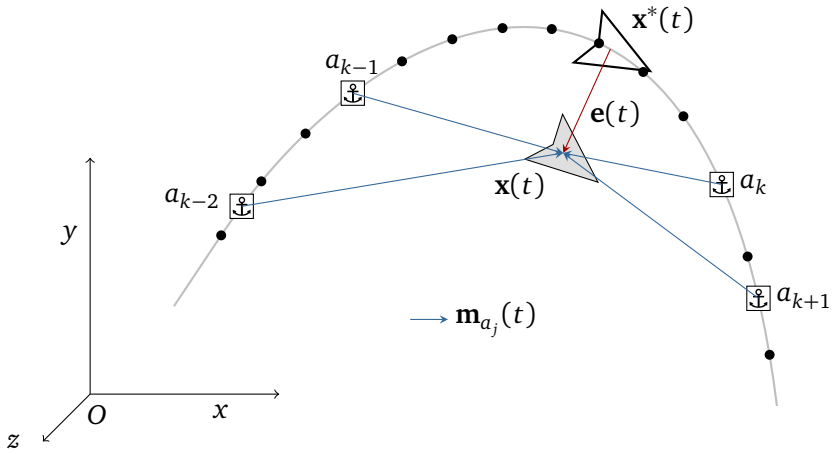


Figure 5.3: During the repeat phase, a scan matcher can provide estimates $\mathbf{m}_{a_j}$ of the robot's offset to nearby anchor points a_j. Together with the odometry $\mathbf{x}^*$ recorded in the teach phase (black dots) we can estimate the feedback error $\mathbf{e}(t)$.

i.e., *dead reckoning*. However, this approach is subject to drift that becomes significant over long paths.

To deal with this issue, we exploit the anchor points introduced in the previous section. At each anchor point we recorded a laser scan during the teach phase. During the repeat phase, a scan matcher can now compare the current readings of the laser range finder with the scan stored at an anchor point. Given enough overlap between both scans as well as a reasonable initial guess, it can thereby estimate the relative offset of the robot from the respective anchor point, see also Section 3.1.1.

From a probabilistic perspective, the measured offsets to multiple anchor points lead to the task of estimating the configuration of the graph in Figure 5.3. We want to retrieve the configuration that best explains the observed scan matcher offsets $\mathbf{m}_{a_j}$ and the relative odometry readings $\mathbf{u}_i = \mathbf{x}^*(t_{i+1}) \ominus \mathbf{x}^*(t_i)$ recorded in the teach phase:

$$\mathbf{x}^*, \mathbf{x}^*(t), \mathbf{x}(t) = \underset{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}^*(t), \hat{\mathbf{x}}(t)}{\arg\max}\ p(\mathbf{m}_{a_1}, \ldots, \mathbf{u}_1, \ldots \mid \hat{\mathbf{x}}^*, \hat{\mathbf{x}}^*(t), \hat{\mathbf{x}}(t))\, p(\hat{\mathbf{x}}(t)), \tag{5.5}$$

where the prior $p(\hat{\mathbf{x}}(t))$ is computed from the estimate of $\mathbf{x}(t_p)$ of the previous time step t_p and the current odometry reading of the robot. Given the most likely configuration of the graph in Figure 5.3, one can then retrieve $\mathbf{e}(t)$ from $\mathbf{x}(t)$ and $\mathbf{x}^*(t)$ via Eq. (5.4). The estimation problem resembles a graph-based simultaneous localization and mapping (SLAM) problem that, under the assumption of Gaussian noise, can be cast to a non-linear least-squares optimization problem, see for example Grisetti et al. (2010).

However, in our approach we approximate the full problem. We expect matching the current laser measurements of the robot against the stored scans at multiple anchor points as well as solving the full optimization problem to yield the best results, but this

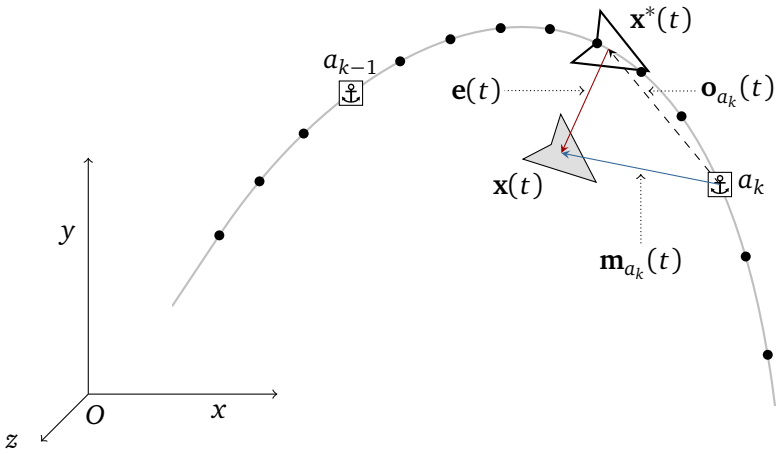


Figure 5.4: During the repeat phase, the reference pose of the robot (hollow dart) at time t, $\mathbf{x}^*(t)$, is expressed as relative offset $\mathbf{o}_{a_k}(t)$ with respect to the current anchor point a_k, see Eq. (5.6). The actual offset $\mathbf{m}_{a_k}(t)$ of the current robot pose $\mathbf{x}(t)$ (shaded dart) from the anchor point a_k is measured by a scan matcher and used to compute the error $\mathbf{e}(t)$ as in Eq. (5.8).

involves considerable computational costs. Since we need to generate the feedback error $\mathbf{e}(t)$ at a high frequency for accurate tracking of the trajectory, we consider only the anchor point closest to the robot. Furthermore, solving the optimization problem and fusing the estimate from the current and the previous time step requires knowledge about the uncertainties of odometry and laser scanner measurements. To make our approach applicable to scenarios in which good estimates for these measurement uncertainties are not available, we also do not fuse the estimates of the previous and current time step in our approach. Our experiments show that despite these approximations we can still achieve highly accurate execution. We expect that fusing estimates would even improve the tracking performance in application scenarios in which adequate estimates of sensor uncertainties are available.

During the repeat phase we only rely on the anchor point closest to the current robot pose to compute the feedback error $\mathbf{e}(t)$. Since we perform all computations relative to the closest anchor point, accumulating drift in the robot odometry will not affect our estimates. Figure 5.4 shows the corresponding graph with the actual robot pose indicated by a shaded dart and the reference pose on the taught trajectory indicated by a hollow dart. Here, anchor point a_k is selected as the one closest to the robot. In the following, we drop the index k for readability.

Given the current closest anchor point $a = (l_a, \mathbf{x}_a^*)$, two steps are necessary to estimate the task error $\mathbf{e}(t)$. First, we compute the reference offset $\mathbf{o}_a(t)$ between the reference pose $\mathbf{x}^*(t)$ and the pose of a at time t as:

$$\mathbf{o}_a(t) = \mathbf{x}^*(t) \ominus \mathbf{x}_a^* . \tag{5.6}$$

Figure 5.4 shows this vector as arrow pointing to the hollow dart. Second, the scan matcher compares the current laser scan with the scan stored at the anchor point. Given anchor point $a = (l_a, \mathbf{x}_a^*)$ and the current laser scanner reading, the output of the scan matcher is the offset between the respective poses at which the robot recorded the scans:

$$\mathbf{m}_a(t) = \mathbf{x}(t) \ominus \mathbf{x}_a^* . \tag{5.7}$$

This vector is the arrow pointing from the anchor to the shaded dart in Figure 5.4. We initialize the scan matcher by compounding the offset computed in the previous time step with the odometry estimate of the robot motion that occurred in the meantime to ensure fast convergence to the correct minimum.

Analyzing Eq. (5.6) and Eq. (5.7), we observe that the reference offset $\mathbf{o}_a(t)$ and the measured one $\mathbf{m}_a(t)$ now represent the reference pose $\mathbf{x}^*(t)$ and the estimated pose $\mathbf{x}(t)$ in the reference frame of the same anchor point a. Combining these equations with Eq. (5.4), we obtain the final expression required for feedback control:

$$\mathbf{e}(t) = \mathbf{m}_a(t) \ominus \mathbf{o}_a(t) . \tag{5.8}$$

Figure 5.4 shows this vector with the red arrow connecting the two darts. This error measure does not require a globally consistent grid map, but relies on the closest anchor point. Therefore, drift cannot accumulate between more than two consecutive anchor points.

As mentioned earlier, a scan matcher is typically initialized with an initial guess of the offset between current scan and reference scan. This increases the probability of converging to the desired minimum and also reduces the computational expenses to do so. To compute this initial guess for the current scan, we keep track of the relative motion of the robot as provided by its wheel encoders. To prevent accumulating odometry drift, we apply dead-reckoning only from the last pose estimated by the scan matcher.

When recording the reference trajectory, we keep track of the offset to the last inserted anchor by scan matching against its scan and store this offset together with the succeeding anchor. Based on the offset from the current anchor point, we can then determine an estimate for the offset from any succeeding anchor point in the taught trajectory in a very efficient manner. We exploit this information in the decision on when to switch the current anchor point. In practice, we keep track of the distance to the current anchor and compute estimates for the distances to the next few anchor points of the recorded trajectory. Should any of these anchor points be closer than the current one, we base the future matchings and error computations on this anchor point.

5.2.1 Controllers for repeating the taught trajectory

A wide range of controllers can employ the task error defined above and the achievable tracking performance will naturally depend on the chosen controller and on its parameterization. The choice of the controller is orthogonal to the proposed feedback error scheme

and outside the scope of this work. Nevertheless, to test our feedback control scheme, we implemented two classical controllers, one for holonomic, and one for nonholonomic kinematics.

For both controllers, we use the velocities $\mathbf{v}^*(t_i)$, recorded during teaching, as the feedforward part. For $t \in \left[t_i, t_{i+1}\right]$, we get $\mathbf{v}^*(t)$ by linear interpolation of $\mathbf{v}^*(t_i)$ and $\mathbf{v}^*(t_{i+1})$. For the holonomic robot, we employ the linear feedback controller introduced in Chapter 2, Eq. (2.15):

$$\mathbf{v} = \mathbf{v}^* - \mathrm{diag}(g_x, g_y, g_\theta)\,\mathbf{e}\,, \tag{5.9}$$

with g_x, g_y and g_θ positive scalar gains. Plugging this into the derivative of Eq. (5.4) yields $\dot{\mathbf{e}} = -\mathrm{diag}(g_x, g_y, g_\theta)\,\mathbf{e}$, for which, as desired, $\mathbf{x}^*$ is a globally asymptotically stable equilibrium.

For the nonholonomic robot we use:

$$\begin{cases} v_x = v_x^* \cos e_\theta - g_x e_x \\ v_\theta = v_\theta^* - g_y e_y - g_\theta e_\theta\,, \end{cases} \tag{5.10}$$

with g_x, g_y and g_θ positive scalar gains. This controller is known to give good performance for trajectory tracking, see for example Siciliano et al. (2009).

To account for communication and execution delays encountered with our robots, we retrieve the feedforward component from the taught trajectory with a lookahead of Δt_{del} into the future, i.e., we use $\mathbf{v}^*(t + \Delta t_{\mathrm{del}})$ instead of $\mathbf{v}^*(t)$ in the controllers in Eq. (5.9) and Eq. (5.10). We do this analogously to Chapter 2, in which we first introduced this paradigm.

Commonly, one uses the odometry readings of the wheel encoders as the heartbeat of a feedback controller for a mobile robot. However, odometry and laser readings are typically not available at coinciding times on real robots. Therefore, we determine the current offset from the anchor point by compounding the offset computed for the last processed scan with the subsequent motion of the robot to retrieve the offset for the current odometry reading.

5.3 Experiments

We implemented the proposed algorithms in C++ and tested them on two real robots: the holonomic KUKA omniRob already used in previous chapters and a differential drive Pioneer P3-DX robot, see Figure 5.5. Both robots are equipped with laser scanners, calibrated with respect to the robot center. The omniRob is equipped with two SICK S300 with a 270 degree field of view and 541 beams. The scanner on the Pioneer is a SICK LMS 291 with a 180 degree field of view and 181 beams. To highlight the performance of the proposed system, we chose to use only the front laser of the omniRob. The approach, however, can easily be extended to multiple laser scanners.

For scan matching we rely on a variant of the iterative closest point algorithm with a point-to-line metric as proposed by Censi (2008). This scan matcher already demonstrated its high accuracy and its ability to work in complex environments with a moderate amount of changes in Chapter 3.

Figure 5.1 shows the environment in which we conducted the experiments. A Motion Analysis motion capture studio with ten Raptor-E cameras provides the ground truth for our evaluations. During both teaching and repeating, the system tracks our marker-equipped robots at 100 Hz, typically with sub-millimeter precision. We use the same system that we introduced in Chapter 3 with one additional camera. We repeat the trajectories, using respectively the controllers in Eq. (5.9) for the omniRob and Eq. (5.10) for the Pioneer, with the error calculated using Eq. (5.8). We compare the performance of our proposed method with a method in which we derive the feedback error $\mathbf{e}$ from localization in a globally consistent grid map. We localize in the global map with Monte Carlo localization as previously described in Section 2.2.

Unless stated otherwise, we used the following parameters for our controllers: For the omniRob, we set the gains to $g_x = 2, g_y = 2, g_\theta = 1$ and $\Delta t_{\text{del}} = 0.15\,\text{s}$, whereas for the Pioneer we used $g_x = 0.6$, $g_y = 14, g_\theta = 3.5$ and $\Delta t_{\text{del}} = 0.2\,\text{s}$.

For the omniRob, we inserted anchor points every $\tau_l = 0.07\,\text{m}$ or $\tau_\theta = 0.05\,\text{rad}$ (2.9 deg) along trajectories. For the Pioneer, laser measurements were available at a higher frequency allowing to insert anchor points approximately every $\tau_l = 0.03\,\text{m}$ or $\tau_\theta = 0.05\,\text{rad}$ (2.9 deg).

To assess the quality of trajectory reproduction with our proposed error design from Eq. (5.8), we compare the motion capture studio recordings of the repetitions with the recordings of the user demonstrations. Since sufficiently accurate time alignment of capture data from teach and reference trajectories was not available, we compute the minimal distance to the polyline path given by the captured points of the reference trajectory as error measure. We then report statistics on these minimal distances over multiple repetitions. To show that our approach is indeed able to track a trajectory and not only a path, we manually align the recordings for one set of runs to also show the evolution of the error over time.

5.3.1 Experimental setup

We test our approach on three different reference trajectories for the omniRob and the Pioneer, yielding a total of six different trajectories. Figure 5.5 shows the reference trajectories as recorded by our motion capture studio, we name them *FigureEight*, *PointToPoint*, and *Fast* for future reference. The trajectories last between 36 s and 70 s with velocities of up to $1\,^{\text{m}}/_{\text{s}}$ (omniRob) and $0.5\,^{\text{m}}/_{\text{s}}$ (Pioneer).

The chosen trajectories aim at representing typical motion patterns for different tasks. The FigureEight trajectory represents a general pattern that requires the robot to travel

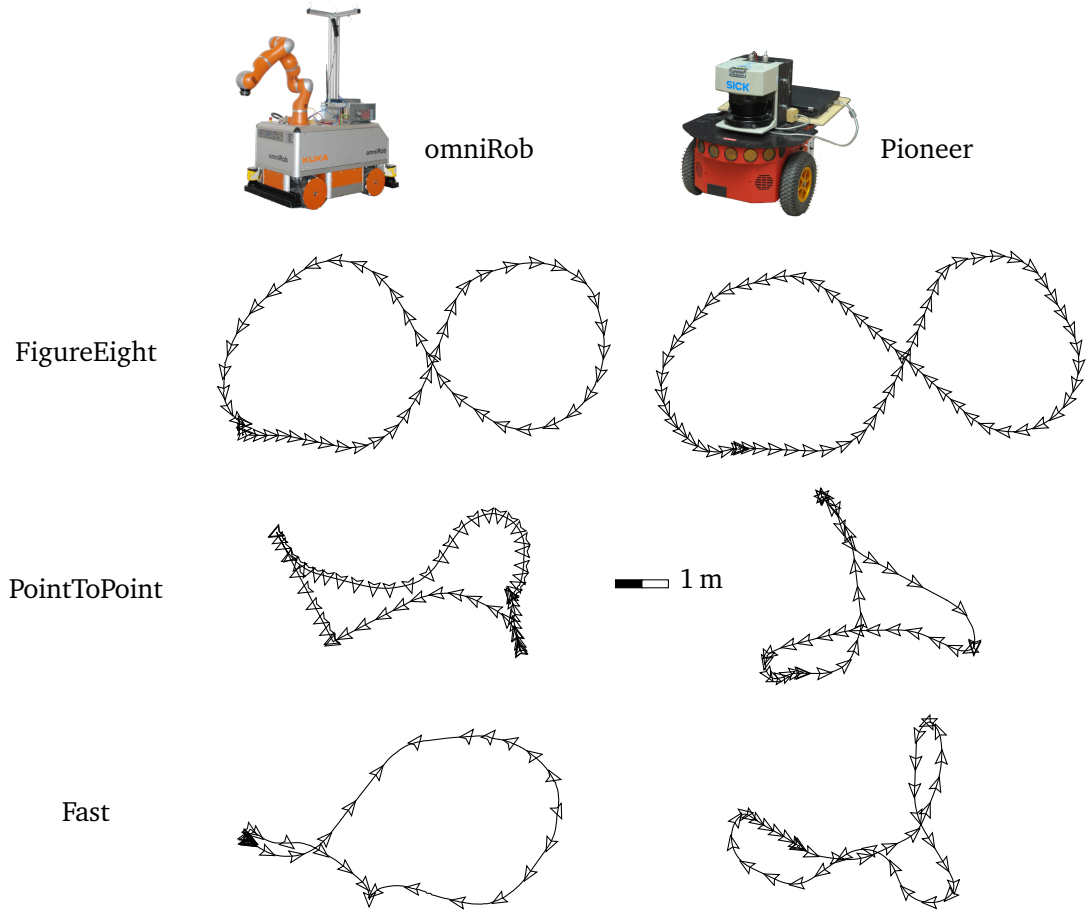


Figure 5.5: The six different taught reference trajectories from our experiments. The darts indicate robot configurations and are drawn every second to convey velocity. All paths use the scale indicated in the center of the figure.

smoothly over longer distances, resembling tasks such as patrolling or monitoring, which require continuous operation. The reference trajectory PointToPoint is less smooth than the FigureEight and contains turns on the spot. The idea behind this type of trajectory is to represent typical pick-and-place tasks or more general point-to-point travel, in which the robot needs to reach a predetermined location, perform a manipulation task and then move to the next location. Finally, we wanted to test the performance of the proposed system when the robot travels near its maximum controllable speed and with more dynamic maneuvers. We perform this test with the Fast reference trajectory.

We also compare our feedback error scheme with a scheme based on global localization on a globally consistent grid map (GLOB) and with a scheme based only on wheel encoders (ODO). For these schemes, we only change the way of measuring the offset $\mathbf{m}_a$ to the current anchor point. For the GLOB scheme, we built a global grid map of the robot environment with a resolution of 0.01 m and augmented the anchor points during the

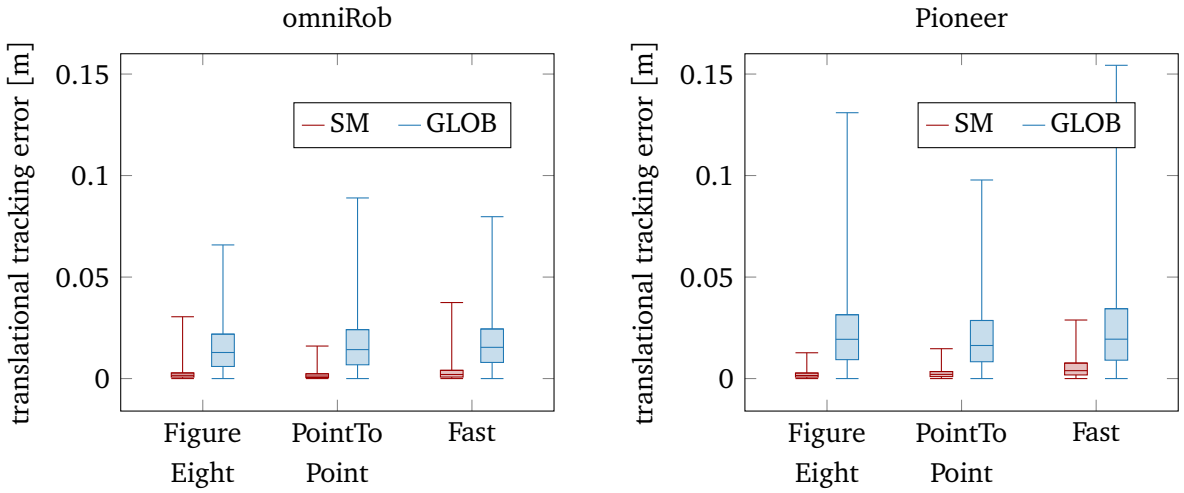


Figure 5.6: Boxplots for the translational tracking error (see text) when reproducing the taught reference trajectories shown in Figure 5.5. In addition to the lower and upper quartile, the plots show the median, as well as the minimum and maximum error over all runs (whiskers). We gathered the data over 98 runs for the proposed scan matching based feedback error (SM) and over 92 runs for the feedback error based on localization on a grid map (GLOB).

teach phase with the respective MCL estimate of the robot's location (see Section 2.2). During repetition, the system then determines $\mathbf{m}_a$ through the current global localization estimate. Note that this latter scheme corresponds to the widespread approach for laser-based navigation. For repeating the trajectory with odometry alone, we use only one anchor point at the start of the trajectory. We determine the offset to this anchor by keeping track of the relative motion of the robot since the start.

5.3.2 Comparison with localization on a globally consistent map

Figure 5.6 shows statistics over the minimum distance error measure (introduced above) for trajectory reproduction for both robots and all reference trajectories. In total, we executed 190 runs to gather the data reported in this figure. To avoid biasing the evaluation with the user errors made in positioning the robot at the start point of the trajectory, we discard data generated in the first 1.0 s (omniRob) and 2.0 s (Pioneer) of the trajectory reproduction.

Figure 5.7 shows example paths taken by the omniRob and the Pioneer for the FigureEight reference trajectories. For these particular examples, Figure 5.8 depicts the norm of the tracking error over time as a result of a manual time alignment of motion capture data from the teach and repeat phases. The low errors over the whole trajectory indicate that we are indeed tracking the reference trajectory and not just the geometric path, i.e.,

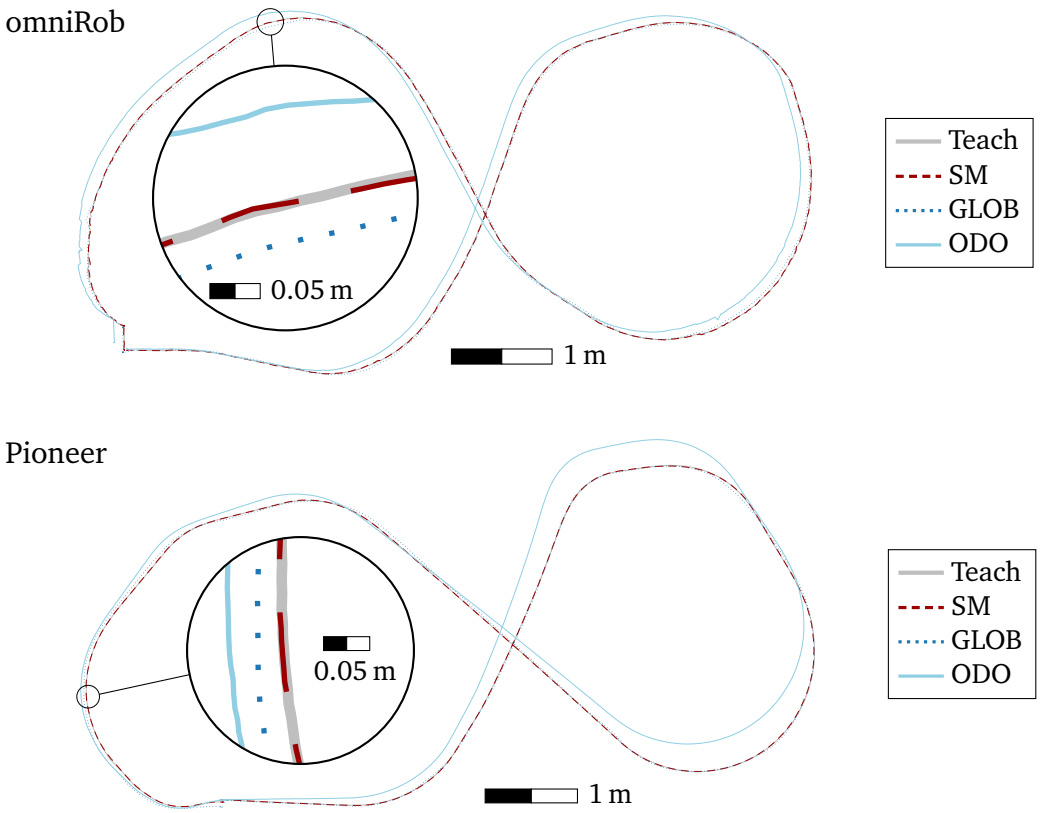


Figure 5.7: Taught and repeated FigureEight trajectories for each approach to feedback error computation with the omniRob and the Pioneer.

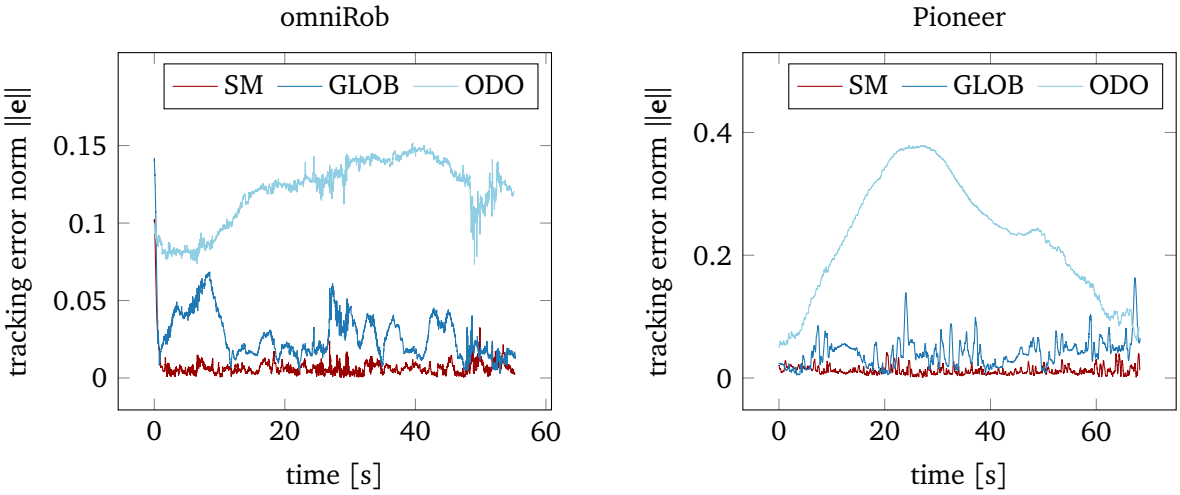


Figure 5.8: Norm of the full tracking error $\|\mathbf{e}\| = \sqrt{\mathbf{e}_x^2 + \mathbf{e}_y^2 + \mathbf{e}_\theta^2}$ over time for FigureEight for omniRob (left) and Pioneer (right) as measured by the motion capture studio for the runs also shown in Figure 5.7.

that our approach is also accurate with respect to the demonstrated velocities. Typically, the errors computed at a specific point in time are higher than the ones considering only the geometry of the tracked path. Moreover, the figure also includes the very start of the trajectory reproduction. Especially in the omniRob experiments, there are higher errors at the beginning, which the user introduced through inaccurate positioning of the robot at the start.

The plots show that both our approach (SM) and a computation of the feedback error based on localization on a globally consistent grid map (GLOB) are able to track the trajectory with considerable accuracy. However, when analyzing the error, we can see that SM achieves an accuracy that is higher by one order of magnitude compared with GLOB. For the Pioneer PointToPoint experiment, the higher errors made it necessary to adapt the controller gains to $g_y = 4.0, g_\theta = 1.0$ for GLOB, as the original values led to oscillation and worse results.

As the plots show, the proposed approach is able to reproduce the taught trajectories with high accuracy and precision, not only on average but also in the worst case. These results are general and do not depend on the type of the robot or on the specific path driven. In other words, our feedback error scheme is able to effectively solve the desired task without relying on a globally consistent, metric map. The system tracks the reference trajectories taught by the user with high precision for both, a holonomic and a differential drive robotic platform. The higher tracking errors of the GLOB tracking scheme can be attributed to the discretization introduced by the grid map and to the local residual inaccuracies resulting from the optimization process that establishes global consistency in the map.

5.3.3 Comparison with wheel encoders

For further analysis and insight, we also compare against the tracking performance of 31 runs with a purely odometry based feedback error computation (ODO) for the FigureEight experiments. The drift of odometry becomes apparent in Figure 5.7 and explains the substantially larger errors in the boxplots in Figure 5.9 and in the tracking error in Figure 5.8. The reason for the still moderate errors is the path shape of the FigureEight experiment: The errors introduced by odometry drift tend to cancel out by the maneuvers, leading to longer periods of moderate tracking error. The analysis of pure wheel encoder based tracking also confirms that the high accuracy achieved by our approach is not simply caused by decent wheel encoders and floor conditions.

5.3.4 Evaluation of anchor point density

In this experiment, we analyze the effects of substantially reduced anchor point density along the taught trajectory. We performed the experiment with seven runs each for

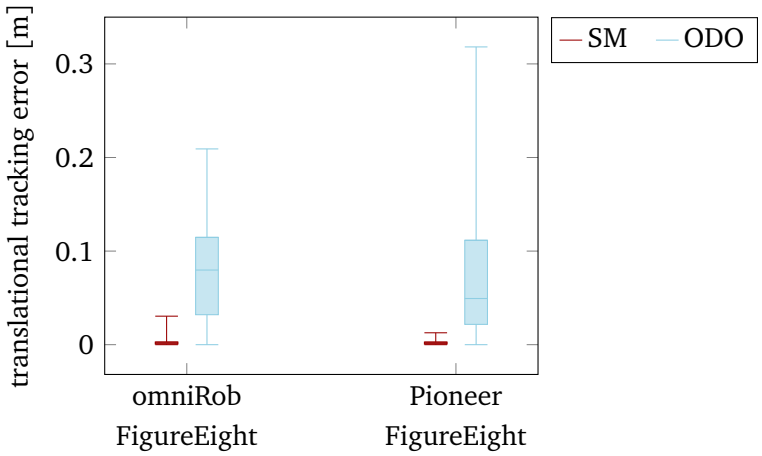


Figure 5.9: Boxplots comparing tracking errors for our scan matching based feedback error computation (SM) and feedback error generation based on odometry only (ODO) for the FigureEight trajectories. The shape of the reference trajectory explains the still moderate errors for odometry, the effects of odometry drift partially cancel out (see Figure 5.7).

the Pioneer reference trajectories, for which we reduced the anchor points to occur approximately every $\tau_l = 0.5\,\text{m}$ or $\tau_\theta = 0.5\,\text{rad}$ (29 deg). This decreased the number of anchor points from 810 to 46 for FigureEight, from 567 to 40 for PointToPoint, and from 444 to 41 for Fast. Figure 5.10 shows the results of this evaluation. Despite the reduction of anchor points by an order of magnitude, the tracking errors increase only moderately. The increase is the highest for the PointToPoint trajectory which contains prominent turns on the spot. Here, the new rotational offset of 29 deg between anchor points has the highest impact, since it severely reduces the overlap between laser scans for the scan matcher. The stable results for reference trajectories with substantially sparsified anchor points show that the high performance of our approach is due to the underlying principle and cannot simply be explained by a huge amount of data expended to tackle the task.

To maintain acceptable tracking results, the sparsification of the anchor points has to stay within limits to maintain a sufficient overlap between laser scans at intermediary positions and the laser scans stored in the anchor points. Otherwise, the scan matcher will fail to compute the offset from the anchor point, as this information is simply not contained in the stored data anymore. Depending on the environment and the sensor field of view, especially a too high value for the rotational offset between anchor points will lead to failures when tracking reference trajectories that contain turns on the spot.

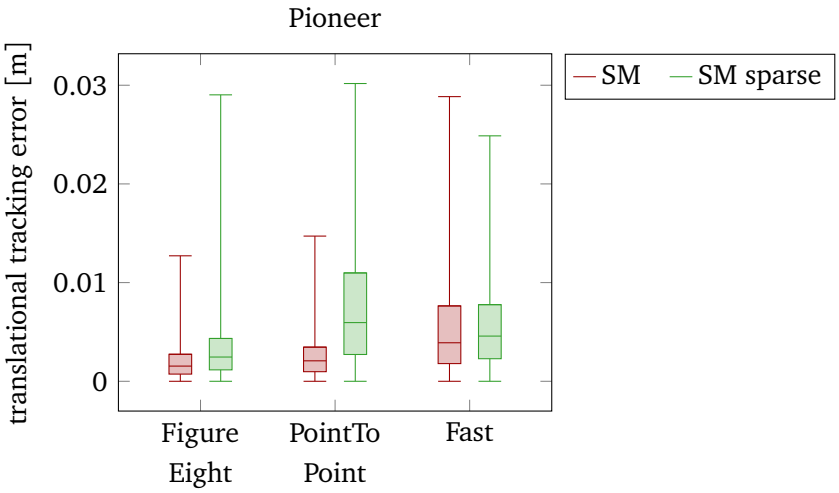


Figure 5.10: Boxplot comparing the performance of our approach (SM) against a version with a substantially reduced number of anchor points (SM sparse) with the Pioneer robot.

5.4 Optimization of Demonstrated Trajectories

As shown in our experiments, our teach-and-repeat framework is able to very accurately reproduce the trajectory demonstrated by the user, including its velocities. This means, however, that the robot also reproduces unnecessary stops, wiggles, accelerations, and in general non-smooth behavior often exhibited by inexperienced users when manually steering the platform. Therefore, post-processing the user demonstration bears potential to increase execution speed and to reduce wear and tear on the robot.

In this section, we present an extension of our teach-and-repeat framework that employs optimization to improve over the user demonstration for omnidirectional robots, extending the paradigm of our approach to teach-*optimize*-repeat. While the framework for recording and executing trajectories remains unchanged, we insert an optimization phase that changes the reference poses $\mathbf{x}^*(t)$ and velocities $\mathbf{v}^*(t)$ for the repeat phase. This is possible without any changes to the repeat phase since our framework does not require the anchor points to lie directly on the reference trajectory, the anchor points just have to be close enough to the reference trajectory to provide sufficient overlap for the scan matcher to provide an offset estimate during execution. This constraint is however dominated by a much stronger constraint inherent to the teach-and-repeat paradigm: The robot is supposed to repeat the user demonstration without substantial deviation since the user implicitly encodes constraints and rules into the demonstration that might not be evident to the robot. Therefore, we constrain the optimization that we perform on the user demonstration to respect bounds for the deviation of the demonstration. To obtain a flexible system, we suggest that the user can indicate different strictness levels for the

Figure 5.11: Extending our framework to teach-optimize-repeat for omnidirectional robots: We optimize the user demonstration (dashed line) within variable, user-defined bounds (shaded area) to obtain a smoother and faster execution (solid line).

deviation bounds during the demonstration by pushing a button on the joystick. We illustrate the concept in Figure 5.11 that shows the initial user demonstration as well as the final optimized reference trajectory. The figure also shows the deviation bounds for the optimization which the user set to be stricter for the narrow passage.

The optimization reduces the time of travel and increases the trajectory smoothness as measured by the integral of squared acceleration. It operates iteratively on convex sub-problems while obeying constraints on maximum velocity and acceleration as well as on the deviation from the original demonstration. The inner workings of the optimization are beyond the scope of this thesis and presented in detail by Mazuran et al. (2015).

We performed experiments with the extended framework and the omnidirectional robot used in the experiments reported above. We recorded four different demonstrations and compared executions of the original recordings with executions of the optimized variants for each demonstration. Again, we used a motion capture studio to evaluate the performance of trajectory execution. However, there is no direct ground truth available for the optimized demonstrations. By replicating the deformations that the optimization performed on the original demonstrations to the motion capture recording of the original demonstrations, we are able to compute an approximation of the execution error. For all four demonstrations we performed approximately ten runs for each setting and the

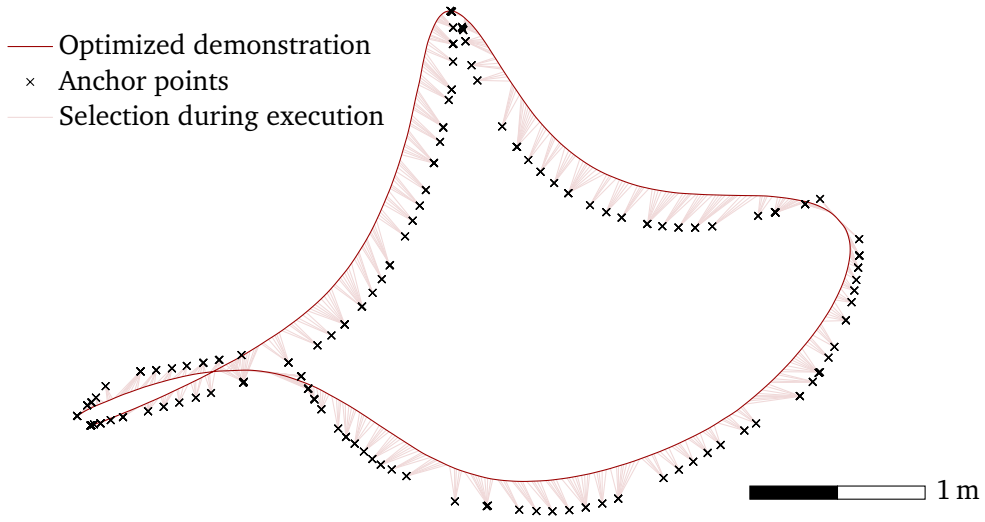


Figure 5.12: Anchor points along the demonstrated path that were selected when tracking the optimized demonstration with 0.2 m allowed deviation from the original. Slight gaps in anchor point selection may occur due to differences in orientation, which are not shown here. One can see how the optimization deviates from the demonstration to save travel time and achieve a smoother trajectory. This figure shows data in the odometry frame of the robot.

execution errors were comparable to the ones of the original demonstrations while the resulting trajectories were smoother and traversed faster than the original demonstrations.

Figure 5.12 illustrates how the optimized demonstration can deviate from the anchor points that lie on the original demonstration. The figure shows which anchor point our framework selected in the repeat phase along the trajectory to determine the feedback error. One can also observe, how the optimized trajectory is smoother with less sharp curves. This effect is also illustrated in Figure 5.13, which shows how the optimization exploits different deviation bounds to optimize the shape of the demonstrated trajectory. We show the effect of the optimization on the trajectory velocities in Figure 5.14: In addition to a smaller duration of the trajectory one can also observe a substantially smoother behavior of the velocities.

The reported experiments show that our framework is capable of executing trajectories that have been moderately altered with respect to the user demonstration. This enables application scenarios that post-process user demonstrations to reduce noise and improve overall trajectory quality to reduce for example wear and tear, as done by the presented optimization.

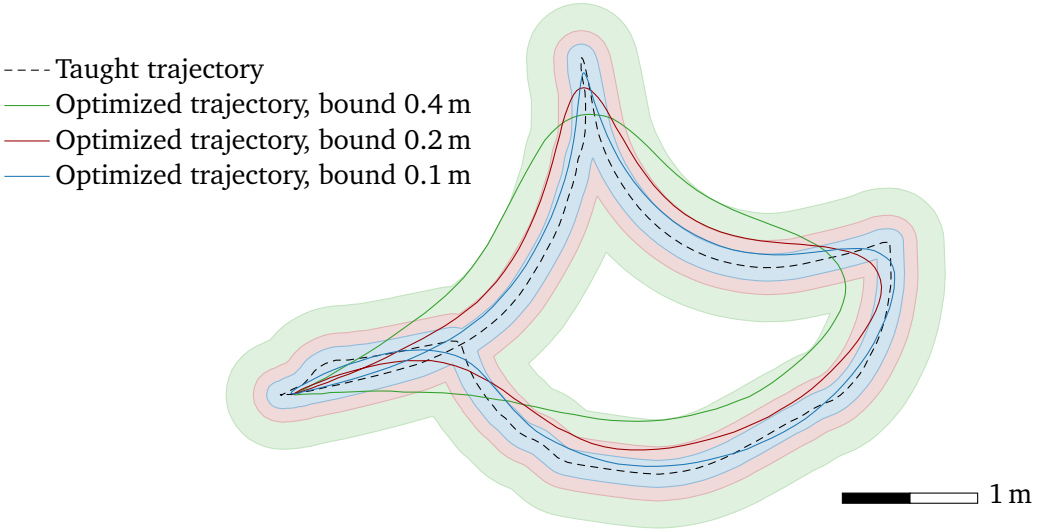


Figure 5.13: Motion capture recordings for trajectory executions of the original user demonstration as well as for optimized variants that used different bounds for the deviation from the original demonstration (shaded areas).

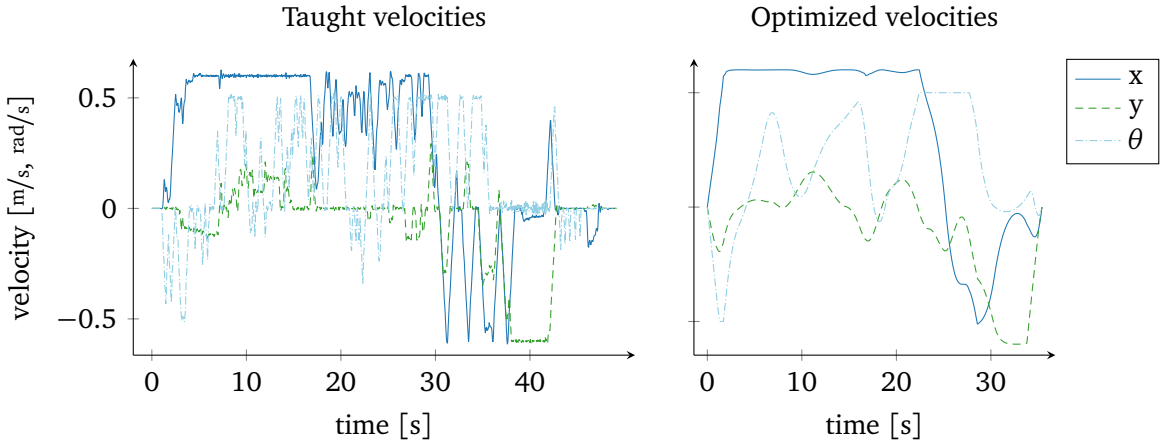


Figure 5.14: Velocities of a user demonstration (left) vs. resulting trajectory velocities after optimization of the user demonstration (right). The optimized trajectory is faster and exhibits a smoother velocity profile.

5.5 Repeatable Experiments with Mobile Robots

In this section, we present an application that extends the original use case of inexperienced users intuitively instructing mobile robots for navigation tasks. In many experiments it is important to reproduce robot behavior as accurately as possible to establish comparability between evaluated approaches. The high accuracy and repeatability of our framework for teach-and-repeat navigation lends itself to the application scenario of such repeatable experiments with mobile robots.

In this application, we evaluate the benefits of splitting the view of a depth camera with a mirror in the context of simultaneous localization and mapping with color and depth information (RGB-D SLAM). For a fair comparison, we need to provide the different sensor setups under test with input that is as similar as possible. Typically, one solves this by operating all sensor setups in parallel. For Kinect-style RGB-D cameras, however, sensor crosstalk renders operation with overlapping fields of view impractical. To still generate comparable and unbiased view sequences for different sensor setups, we leverage the high accuracy and repeatability of our teach-and-repeat navigation. We demonstrated a trajectory to the differential drive Pioneer robot shown in Figure 5.15. Then, we let our framework repeat the demonstrated trajectory, with different sensor setups mounted on the robot for the repetitions. Figure 5.15 shows the trajectory used for evaluation over a map of the environment generated by RGB-D SLAM with one of the tested sensor setups.

Endres et al. (2014) present a thorough analysis of the benefits of splitting the field of view of RGB-D cameras in the context of SLAM and methods to calibrate such setups. They perform a quantitative analysis that employs the teach-and-repeat framework presented in this chapter to generate comparable views for the different sensor setups. This application scenario for our framework for teach-and-repeat navigation showcases the benefits of its high accuracy.

5.6 Related Work

The most widespread approaches to mobile robot navigation are model- and appearance-based approaches. Model-based approaches usually rely on a map of the environment, similar to the navigation system presented in Chapter 2 to Chapter 4 of this thesis. In the previous chapter, we realized teach-and-repeat tasks by fitting a spline-based path representation to user demonstrations that were localized in a global grid map with the help of Monte Carlo localization as described in Section 2.2. Also the work by Baumgartner and Skaar (1994) realizes teach-and-repeat scenarios with a previously given model of the environment that contains visual cues. In the framework presented in this chapter, the goal is to repeat the user input as accurately as possible, without relying on a globally consistent model of the environment.

Figure 5.15: Application of our framework for teach-and-repeat navigation to repeatable experiments with mobile robots. For fair comparison of different depth camera setups these need to work on as identical input as possible. The figure shows the employed differential drive robot with one of the sensor setups on the left. The right part shows the trajectory that we repeated for all sensor setups under comparison over a map of the environment.

Appearance-based approaches to robot navigation do not require an environment model and work directly in the space of the sensor data. The environment is usually described by a topological graph, in which each node corresponds to the description of a position, and a link between two nodes defines the possibility for the robot to move autonomously between the two positions that the link connects. Our approach belongs to the category of appearance-based navigation, however, we store the user demonstration rather as a sequence than as a graph and the edges contain detailed information on the motion of the robot rather than just the connectivity itself. Recently, many researchers have developed appearance-based frameworks for teach-and-repeat navigation. As also implemented in our framework, a human operator controls the robot motion during a preliminary teach phase during which the robot acquires a sequence of sensor data and stores it in a database. Then, during the repeat phase, the task of the robot is to repeat the same path, by comparing the currently observed and the previously recorded sensor data.

Pioneering work in appearance-based teach-and-repeat navigation by Matsumoto et al. (1996) associated a particular motion (e.g., “go forward”, “turn left”) with each recorded image, in order to move from the current to the next image in the database. The work by Royer et al. (2007) builds a three dimensional representation of the taught path from the image sequence and uses a classic path following controller for motion generation. Goedemé et al. (2007) propose a similar approach, but use omnidirectional cameras. In

their work, they represent a path by a sequence of images and reconstruct a local 3D map from them. They follow the path with a sequence of homing vectors to each image and localize the robot with feature tracking. The controller presented by Argyros et al. (2005) exploits angular information regarding the features matched in panoramic images. Similarly, Koch et al. (2010) employ a bearing-only control law to steer the robot.

Cherubini and Chaumette (2009) propose a novel varying reference based teach-and-repeat framework for outdoor navigation with a monocular camera. In further work, Cherubini et al. (2011) compare different visual servo methods for path reaching and following and introduce the use of a laser range finder for obstacle avoidance in the context of visual navigation (Cherubini and Chaumette, 2013). Furgale and Barfoot (2010) present a visual teach-and-repeat approach for long-range navigation. McManus et al. (2013) extend this work and replace the camera images by laser reflectivity images in their approach.

While all works discussed so far rely on images, in this chapter we validated a similar approach, however based solely on odometry and 2D laser scanners, that are generally mounted on industrial robots for safety reasons. Closely related to our work is the approach of Marshall et al. (2008). The main idea of their work is to construct a sequence of local grid maps rather than having a globally consistent, monolithic map. During the teach phase they build these local maps and record the path of the robot. They then repeat the recorded path by localizing the robot in the local maps using the unscented Kalman filter. Our approach differs from this method as we do not rely on grid maps but instead operate directly on the non-discretized sensor data. Krüsi et al. (2014) follow a similar environment representation as Marshall et al. and use point clouds from a 3D laser scanner instead of 2D laser scans.

As discussed in this chapter and the previous chapter, teach-and-repeat systems can help to improve the flexibility of automation in industrial settings. In this context, Kelly et al. (2007) present an infrastructure-free robot that uses three cameras to navigate with naturally occurring visual cues learned during operation. The teach-and-repeat framework for a mass customization manufacturing scenario presented by Malheiros et al. (2009) relies on high-intensity visual markers and stereovision to enable quick and easy use and configuration for non-skilled users. The system presented in this chapter, as well as the system proposed in previous chapter, relies on the measurements of the mandatory safety laser range finders and does not require any additional infrastructure.

All of the related works discussed so far do not aim to reproduce the velocities of the user-demonstrated trajectory. Instead, they follow the geometry of the path with velocities determined by the respective tracking controllers. Some works adapt the velocities to constraints given by the platform and its environment, e.g., Marshall et al. (2008) generate a speed schedule considering the limitations of the vehicle, obstacle distance and path curvature. Similar to our extension to optimize the reference trajectory with respect to duration and smoothness, other authors also propose modifications of the reference signal.

Ostafew et al. aim to increase platform speed over several executions based on previous experiences (Ostafew et al., 2014b) and in further work change the objective towards reducing the tracking error as measured by the robot itself (Ostafew et al., 2013, 2014a). While the optimization scheme proposed in our extension has a different cost function, it furthermore introduces the idea of non-stationary, user-defined bounds on the deviation from the original demonstration.

The teach-and-repeat paradigm falls into the category of programming by demonstration, for which we discussed related works in Section 4.7.

5.7 Conclusion

In this chapter, we presented a framework for intuitive specification of robot navigation tasks. With our approach, the user only needs to demonstrate the desired trajectory once during a teaching phase and does not need to build a globally consistent map of the environment ahead of time. Our approach stores 2D laser data and raw odometry in anchor points during teaching and then uses this data for error feedback control of the robot during the repeat phase. We evaluated our approach with real robots and compared it with alternative ways of feedback error generation for several demonstrated trajectories. Our approach is able to track the taught trajectories with millimeter accuracy and outperforms feedback generation by localization on a globally consistent metrical grid map. We also presented an extension to optimize the demonstration within non-stationary, user-defined bounds towards a smoother and faster trajectory. In addition to the discussed use case of easy to use robot navigation especially in the context of flexible industrial automation, the high accuracy of our approach also lends itself to the presented use case of repeatable experiments with mobile robots.

While in this and the previous chapter we were concerned with repeating user input for navigation, we will return to fully autonomous navigation in the next chapter. More specifically, we will investigate a method to assess and compare the performance of complete navigation systems in changing indoor environments.

6 A Benchmark Protocol for Indoor Mobile Robot Navigation

As discussed throughout this thesis, autonomous mobile robot navigation is an enabling technology for successful industrial applications. To assess and compare the applicability and economic viability of individual systems for a specific scenario, decision makers typically rely on statistics like mean time to failure when hard guarantees are not available for properties of a system. However, the evaluation of navigation systems strongly focused on measuring the performance of individual components, i.e., mapping, localization and motion planning, and relied mainly on datasets or simulations. For real-world evaluations of complete systems, including the interplay of their components, datasets do not suffice due to the active aspect of navigation. Furthermore, realistic scenarios with changing environments challenge the repeatability of results. In this chapter, we present an evaluation methodology for complete navigation systems in changing environments. It enables repeatable results and establishes comparability of experiments conducted with different robots, navigation systems and in different environments. To this end, our benchmark protocol provides detailed definitions to control the standardized test environments and their dynamics. It also introduces a reference system to normalize the performance of navigation systems across experimentation sites. We apply our protocol in experiments conducted by two different research groups, showing the usefulness of the benchmark.

Robot navigation is a well studied topic in robotics due to its cornerstone function for robot autonomy. To assess the applicability and economic viability of a robot navigation system for a specific application scenario, it is necessary to examine the properties of the complete system, including the interplay of its components. While the performance of individual components certainly plays a role, the performance of the complete system in its dedicated task governs the overall decision on whether or not a system is suitable for an application scenario. For practical applications, it is necessary to ascertain statistics like mean time to failure in realistic scenarios. Furthermore, benchmarks of alternative navigation systems need to be comparable across systems and evaluation sites to allow unbiased decisions.

In contrast to these requirements, prior work on benchmarking robot navigation so far primarily focused on simultaneous localization and mapping (SLAM) techniques, and in particular on assessing the accuracy of the generated maps as done by Burgard et al. (2009) and Sturm et al. (2012). These evaluations are useful when the task of the robot is to precisely reconstruct the environment, e.g., for architectural or other surveying purposes. However, when the map is built for autonomous navigation, its metric accuracy does not necessarily relate to the navigation performance of the robot. For robust real-world navigation, the robot has to move, localize and reach its destinations in environments that undergo change with respect to the initial conditions. This includes environments shared with people or environments in which objects like goods, furniture, etc. may be moved around.

For fair benchmarking, performance quantification under repeatable conditions is the key. The goal of this chapter is to present a repeatable and comparable evaluation methodology for autonomous indoor navigation systems. Therefore, we quantify navigation in a fashion similar to other hard sciences where environmental conditions are key for reproducibility and fair comparison: We define a set of performance metrics, a benchmark protocol and a reference system together with means of measuring ground truth. We aim at accommodating for differences in hardware and sensors between compared solutions. In particular, we aim at reproducing identical environments, including environment dynamics between multiple runs at individual experimentation sites.

In other computer science disciplines, such as computer vision and machine learning, benchmarks had a large impact to standardize and to uniform evaluation procedures, see for example the works by Bache and Lichman (2013) and Everingham et al. (2010). Differently from these sciences, robot navigation is not suitable for evaluation with datasets only. The robot is immersed in the environment and interacts with it. The only alternative to real-world evaluations are simulations, which do not fully represent the complexity of the real world.

In our benchmark, we aim to compute statistics about *a simulated year* of continuous robot operation. For this, we provide detailed definitions for the experimental environment and conditions. The experimental setup consists of definitions about the size, the dynamics, the environmental conditions and the overall duration of an experiment. This includes the number and the size of the rooms, the number of people walking in the scene, the kinds and amounts of objects and furniture that are moved and the number of goal poses for each environment. Even under strict definitions, real-world environments are hard, if not impossible, to be precisely controlled. For this reason, we provide a way to compute a baseline navigation performance for each experimental environment based on a *reference system*. We use the reference system to compare and *normalize* the performance of all the other systems under test in each environment.

We applied the benchmarking protocol in experiments conducted by two different research groups using two different kinds of robots, showing the usefulness of the bench-

mark. The complete benchmark protocol along with detailed instructions and our evaluation software is available to the community at `http://research.microsoft.com/brin/`.

We organize the remainder of this chapter as follows. We present our benchmark protocol for navigation in Section 6.1. The protocol comprises a set of challenges for navigation systems that we present in more detail in Section 6.2. To make benchmarks conducted with different robots at different experimentation sites comparable, we introduce a reference system in Section 6.3, in which we also present the measures we use to assess performance along with an overview on how to conduct the benchmark. We apply the proposed protocol in real-world experiments in Section 6.4 and report on results and insights gained from these experiments. In Section 6.5 we discuss related works before we conclude the chapter in Section 6.6.

6.1 Experimental Protocol

In this section, we provide a detailed description of the proposed experimental protocol. The goal of the protocol is to evaluate and fairly compare the performance of complete navigation systems (hardware and software) in real environments over longer periods of time. In order to allow comparison between different navigation systems at different physical locations, we devise means for normalizing the performance across environments and platforms and take measures towards standardization and repeatability of evaluations.

First, we define a standard environment composed of four *areas*. Second, we define a set of *challenges* that the robot has to face. These challenges include changes in environment appearance, geometrical configuration, and dynamic obstacles. Third, we introduce the concept of a *reference robot* and a *reference navigation system* that will be identical across all evaluation sites. Expressing the performance of the tested system relative to this reference system ensures comparability of results across robots and evaluation sites. Finally, we propose a vision-based *ground truth system* to evaluate the navigation performance of both the test and the reference robot.

We propose to simulate an entire year of robot operation, defining twelve *loops*, each corresponding to a virtual month of operation. The experimenter defines eight *waypoints*, two for each area, and creates a route that visits all waypoints and always changes areas between waypoints. The task of the robot is to travel along this route in each loop, facing a different set of challenges for each loop.

Table 6.1 and Table 6.2 show an overview of the experimental protocol. The rows indicate the challenges, while the columns indicate their category, frequency, and configuration/location with respect to each of the twelve loops.

Table 6.1: The proposed benchmark test grid. The table lists the configuration of each challenge for every loop of the benchmark, see Table 6.2 for the second part covering months/loops 7–12.

		Challenge	Cat.[a]	Freq.[b]	Month 1	Month 2	Month 3	Month 4	Month 5	Month 6
All Areas	1	Artificial Lighting	A	D	Off	On	On	Off	Off	On
	2	Lamps On/Off	A	D	On	On	On	On	On	On
	3	Blinds or Drapes Open/Closed	A	D	All Closed	All Open	50/50	All Closed	All Open	50/50
	4	Wall Art Changes	A	Y	Wall Art 1					
	5	Door Open/Closed	G	H	Constantly	Constantly	Constantly	Constantly	Constantly	Constantly
	6	Wall Color Changes	A	Y	Color 1					
Atrium	7	Large Display Monitors Change Content	A	D	Image 1	Image 2	Image 3	Image 1	Image 2	Image 3
	8	Person in Path (ample room to avoid)	O	D		Position 1	Position2	Position 3	Position 4	
	9	Small Group in Path (ample room to avoid)	O	D		Position 1	Position 2	Position 3		Position 1
	10	Person Pushing Cart (ample room to avoid)	O	D		Position 1	Position 2		Position 1	Position 2
Hallway	11	Shipping Boxes on Floor	G	D		1 Box	2 Boxes	3 Boxes	2 Boxes	1 Box
	12	Cart Moves	G	D	Position 1	Position 2	Position 3	Position 4	Position 1	Position 2
	13	Ladders, Tools, Cables	G	Y			Position 1			
	14	Two People Blocking Path (no room to avoid)	O	D	Position 1				Position 2	
	15	Path Completely Blocked (door) for 1 Minute	O	D		Position 1				Position 2
	16	Path Completely Blocked (people) for 1 Minute	O	D			Position 1			
	17	Person Pushing Cart (no room to avoid)	O	D				Position 1		
Lounge	18	Dining Chairs Shift	G	H	Neat	25% Messy	50% Messy	75% Messy	100% Messy	Neat
	19	Coats/Jackets on Coat Racks	G	D		1/2 Full	Full		1/2 Full	Full
	20	Cart Moves	G	D	Position 1	Position 2	Position 3	Position 4	Position 1	Position 2
	21	Caution Sign (Janitor)	G	D		Position 1	Position 2		Position 2	Position 1
	22	Garbage/Recycling Bags	G	D		Black	White			2 Black
	23	Reconfigure Furniture	G	Y	Configuration 1					
	24	Person Vacuuming or Mopping	O	D		Position 1	Position 2		Position 1	Position 2
	25	Large Work/Social Gathering (20-30 people)	O	M			Position 1	Position 1		
Office	26	Whiteboard Contents Change	A	D	Clean	5%	10%	20%	30%	40%
	27	Desk Chairs Shift (less than 1.5 meters)	G	H	Neat	25% Messy	50% Messy	75% Messy	100% Messy	Neat
	28	Coats/Jackets on Chairs	G	D			5%		10%	2%
	29	Bags on Floor Near Desks	G	D		20%	40%	60%		20%
	30	Loose Paper on Floor	G	D	0 Pieces	5 Pieces	0 Pieces	5 Pieces	0 Pieces	5 Pieces
	31	Shelves Contents Change	G	M	20% Full		40% Full		60% Full	
	32	Shelves Move	G	Y	Position 1					
	33	Small Gathering in Work Area (4-8 people)	O	D		Position 1	Position 2		Position 1	Position 2
	34	Social Gathering (10-15 people)	O	M			Position 1			

[a] Challenge category. A: Appearance, G: Geometry, O: (moving) Obstacle

[b] Frequency of occurence. H: Hourly, D: Daily, M: Monthly, Y: Yearly

Table 6.2: Continuation of Table 6.1, the proposed benchmark test grid.

		Challenge	Cat.[a]	Freq.[b]	Month 7	Month 8	Month 9	Month 10	Month 11	Month 12
All Areas	1	Artificial Lighting	A	D	On	Off	Off	On	On	Off
	2	Lamps On/Off	A	D	Off	Off	Off	Off	Off	Off
	3	Blinds or Drapes Open/Closed	A	D	All Closed	All Open	50/50	All Closed	All Open	50/50
	4	Wall Art Changes	A	Y	Wall Art 2					
	5	Door Open/Closed	G	H	Constantly	Constantly	Constantly	Constantly	Constantly	Constantly
	6	Wall Color Changes	A	Y	Color 2					
Atrium	7	Large Display Monitors Change Content	A	D	Image 1	Image 2	Image 3	Image 1	Image 2	Image 3
	8	Person in Path (ample room to avoid)	O	D	Position 1	Position 2	Position 3	Position 4		Position 1
	9	Small Group in Path (ample room to avoid)	O	D	Position 2	Position 3		Position 1	Position 2	Position 3
	10	Person Pushing Cart (ample room to avoid)	O	D		Position 1	Position 2		Position 1	Position 2
Hallway	11	Shipping Boxes on Floor	G	D		1 Box	2 Boxes	3 Boxes	2 Boxes	1 Box
	12	Cart Moves	G	D	Position 3	Position 4	Position 1	Position 2	Position 3	Position 4
	13	Ladders, Tools, Cables	G	Y		Position 2				
	14	Two People Blocking Path (no room to avoid)	O	D			Position 3			
	15	Path Completely Blocked (door) for 1 Minute	O	D				Position 3		
	16	Path Completely Blocked (people) for 1 Minute	O	D	Position 2				Position 3	
	17	Person Pushing Cart (no room to avoid)	O	D		Position 1				Position 3
Lounge	18	Dining Chairs Shift	G	H	25% Messy	50% Messy	75% Messy	100% Messy	Neat	25% Messy
	19	Coats/Jackets on Coat Racks	G	D		1/2 Full	Full		1/2 Full	Full
	20	Cart Moves	G	D	Position 3	Position 4	Position 1	Position 2	Position 3	Position 4
	21	Caution Sign (Janitor)	G	D		Position 1	Position 2		Position 2	Position 1
	22	Garbage/Recycling Bags	G	D	2 White			2 Black	2 White	
	23	Reconfigure Furniture	G	Y	Configuration 2					
	24	Person Vacuuming or Mopping	O	D		Position 1	Position 2		Position 1	Position 2
	25	Large Work/Social Gathering (20-30 people)	O	M			Position 2	Position 2		
Office	26	Whiteboard Contents Change	A	D	50%	60%	70%	80%	90%	100%
	27	Desk Chairs Shift (less than 1.5 meters)	G	H	25% Messy	50% Messy	75% Messy	100% Messy	Neat	25% Messy
	28	Coats/Jackets on Chairs	G	D	20%	4%	30%	6%	40%	8%
	29	Bags on Floor Near Desks	G	D	40%	60%		20%	40%	60%
	30	Loose Paper on Floor	G	D	0 Pieces	5 Pieces	0 Pieces	5 Pieces	0 Pieces	5 Pieces
	31	Shelves Contents Change	G	M	20% Full		40% Full		60% Full	
	32	Shelves Move	G	Y	Position 2					
	33	Small Gathering in Work Area (4-8 people)	O	D		Position 1	Position 2		Position 1	Position 2
	34	Social Gathering (10-15 people)	O	M				Position 2		

[a] Challenge category. A: Appearance, G: Geometry, O: (moving) Obstacle

[b] Frequency of occurence. H: Hourly, D: Daily, M: Monthly, Y: Yearly

6.1.1 Areas

We devise a standardized test environment consisting of four distinct areas: *atrium*, *lounge*, *office* and *hallway*. These areas group the challenges in the leftmost column of Table 6.1 and Table 6.2. The environment should contain at least one doorway and at least two different floor surfaces (e.g., carpet, tile, wood, cement). Ideally, the environment should not be a dedicated testing facility but rather a real building. Where possible, the test areas should have the capability of being artificially illuminated. Additionally, blinds or drapes are necessary to alter the environmental illumination.

We devise each of the four areas to present unique characteristics to the navigation system. The atrium is a predominately open space, at least of 15 m × 15 m, unfurnished in more than 90% of its area. For some robotic systems, these large and open spaces are beyond their sensing capabilities, challenging the employed localization system. The lounge is a social seating/dining area with an intended size of at least 12 m × 12 m. The furniture that clutters the environment is a challenge for navigation systems due to narrow passages and complex obstacle profiles such as sets of chairs and table legs. The office is densely occupied by desks, office chairs and shelves and has a recommended minimum size of 10 m × 10 m. The hallway has an intended length of at least 15 m and should have a low number of geometric and visual features. The high perceptual aliasing of the environment is the intended challenge for navigation systems that may get lost along the longitudinal axis of hallways.

The dimensions suggested above are recommendations, we encourage the experimenter to respect the relative size of the areas in case of space limitations. Figure 6.3 and Figure 6.5 show pictures of the real environments that we used in our experiments.

6.1.2 Challenges

We define a set of rules to govern the dynamics of the changes in the environment (challenges). This standardizes comparison by enforcing repeatability of experimental conditions. Both, the system under test and the reference robot, will face the same challenges at the same time in the benchmark. We list each challenge as a numbered row in Table 6.1 and Table 6.2. The challenges are representative of events and dynamics that are highly likely to occur at least once over a year-long deployment of a robot in a typical indoor environment. They fall into three main categories that Table 6.1 and Table 6.2 show next to the challenge description:

Appearance (A) This category comprises visual appearance changes in the environment such as changing artwork, whiteboard contents and lighting conditions. We design these changes to assess the robustness of vision-based approaches. For systems with range sensors only, these changes do not substantially change the perception of the environment.

Geometry (G) Challenges of this category include movable objects like doors, boxes, chairs, and ladders. These challenges simulate the everyday variation in environment configurations and the different states of articulated objects such as doors. They test the robustness of navigation systems against structural changes with respect to the initial setup and mapping phase. In addition to vision sensors, challenges in this category also affect range sensors. The changes range from small movement of objects to substantial rearrangement of furniture in a room. Both types of changes pose challenges to localization by slightly moving known correspondences or eliminating them completely. Additionally, relatively small unknown objects such as shipping boxes may already occlude the view of the known and mapped environment substantially.

Moving Obstacles (O) This category includes dynamic objects such as moving people, people transporting objects, or group gatherings that can also (completely) block the path of the robot for an extended period of time. This challenge tests the capabilities of a navigation system to replan while moving and to negotiate stalling situations. Note that groups of people close to the robot can substantially hinder the robot's perception of the environment.

All dynamic and moving elements have a designated frequency of occurrence and a designated location. The frequency can be hourly (H), daily (D), monthly (M) or yearly (Y), as indicated by the column next to the challenge category in Table 6.1 and Table 6.2. The tables show the designated location/configuration of a challenge in the respective column for each loop of the benchmark. Depending on the individual system under test, the experimenter can omit some challenges from benchmark. In Section 6.2 we motivate the individual challenges in detail to better guide the decision on which challenges one can omit. In the following, we present details on how to instantiate the test grid of challenges from Table 6.1 and Table 6.2 into an experimental script.

6.1.3 Benchmark test grid

To ensure that the robot faces challenges and environment variations in a standardized and reproducible fashion, we devise a benchmark test grid that regulates the experimental evaluation. While the robot is traveling along its designated route, we modify the environment according to the test grid in Table 6.1 and Table 6.2. The test grid contains instructions that describe the challenges that the robot has to address. For each challenge, the tables list the specific configuration for each of the 12 benchmark loops. For instance, the moving cart challenge (row 12) describes four different positions on which the cart appears in the hallway throughout the benchmark. In the first loop, it is to appear at "Position 1" while in loop 6 it should appear at "Position 2".

To decide on meaningful values for the individual configurations required from the test grid, the experimenter has to first devise positions for the waypoints 1–8. Then, the experimenter defines the order in which the robot has to visit the waypoints, taking care to avoid traveling between two waypoints in the same area. One complete visit of all waypoints counts as one loop, or a benchmarking month, for the evaluation. With the knowledge of the robot's default path the experimenter is then able to provide meaningful positions for the generic configurations of challenges like "Two People Blocking Path (no room to avoid)" (row 14), or "Person in Path" (row 8). It also falls into the responsibility of the experimenter to concretely define configurations for the qualitative settings of the environment dynamics, e.g., a configuration change from "Neat" to "Messy" in an experiment script. We give an overview over all steps needed to implement the benchmark protocol in Section 6.3.3, once we have introduced all necessary components. For a concrete instantiation of the test grid into an experiment script see also Section 6.4.1, in which we present the instantiation of one benchmark loop for one of our experiments.

6.2 The Challenges

In this section, we describe the challenges of our benchmark protocol in more detail. All challenges aim to provoke situations and simulate events that are likely to occur at least once over a year-long deployment of a robot in an indoor navigation scenario. In this context, we put an emphasis on situations that have the potential to trigger and expose failure modes of navigation systems. Table 6.1 and Table 6.2 contain the complete list of challenges for our proposed benchmark protocol. In the following, we will refer to individual challenges by their row number in these tables. Figure 6.1 shows several instantiations of challenges from one of our real-world experiments.

The challenges from the category Appearance (A) aim at testing the robustness of vision-based approaches to navigation. Appearance changes of various degrees are the result of the changes in wall art (4), wall color (6), display content (7), and whiteboard content (26). Another test point for vision-based navigation are changes of lighting conditions. We test robustness to these changes by challenges that modify artificial room lighting (1), the presence of sunlight through blinds and drapes (3) and by a challenge that modifies illumination only for smaller areas through lamps (2). If the navigation system under test does not rely on camera images but instead employs range measurements, one can skip the above challenges from the benchmark protocol.

Challenges from the category Geometric (G) are typically changes in the arrangement of movable objects in the environment. These challenges change how the environment looks like to the sensors of the robot. In difference to the appearance changes above, these challenges also affect navigation systems that rely on range senors. Changes in the robot's perception of the environment with respect to an initially mapped state pose a challenge

path blocked with simulated door (15)

varying number of boxes in the hallway (11)

group of people on the path of the robot (9)

substantial reconfiguration of furniture (23,32)

Figure 6.1: Exemplary instantiations for a selection of challenges of our benchmark protocol, the numbers identify the challenges by their row in Table 6.1 and Table 6.2. The images show situations from one of our experiments, see also the experiment description in Section 6.4.1. The brighter rectangles highlight changes between different loops.

to the localization component of the navigation system. Furthermore, paths to waypoints might need to adapt to new object locations to either maintain a suitable distance to obstacles or to take different routes through the environment should the default route become blocked.

A set of challenges realizes objects that appear in changing configurations with respect to the initial setup phase. They consist of a varying number of boxes in the hallway (11), carts that appear at different positions (12, 20), bags and backpacks on the floor (29), garbage bags (22), changing shelf contents (31), a caution sign (21) and doors that change their state between open and closed (5). Since the robot did not perceive the respective objects in the changing configurations during the setup phase, the localization component will face a challenge as it is not able to find a correspondence for the changed sensor readings. The boxes appearing in the hallway force the robot to adapt its path and drive through a smaller passage.

A further set of challenges from the geometric category realizes more subtle changes in appearance. Shifting chairs (18, 27) and coats or jackets appearing on racks and chairs (19, 28) possibly allow the localization to keep the association of sensor data with the initially mapped state. However, the slight differences challenge the localization to not wrongly adapt its estimate of the robot's position with the shifted appearance.

A set of stronger appearance changes from the geometric category consist of moving shelves (32) and a furniture reconfiguration (23). These challenges realize substantial changes in the environment appearance with respect to the setup phase and target the testing of robustness in long-term operation. One can omit these challenges from the benchmark protocol if long-term operation is not of concern for the evaluation at hand.

The remaining challenges from the geometric category test how the robot copes with obstacles and changing floor conditions that it potentially cannot perceive with its sensors. The robot might get tangled in tools or cables (13). Loose paper on the floor (30) can cause odometry problems. The experimenter may skip these challenges if the evaluation focuses on partially structured environments such as flexible automation on industrial shop floors or logistics. Here, floor conditions are typically kept constant.

The third category of challenges are moving obstacles. A first set from this category consists of obstacles that slightly move but do not strongly interfere with the motions of the robot. This set includes a person vacuuming or mopping (24) and groups of people gathering in the environment (25, 33, 34). Similar to the challenges in the geometric category, these challenges block and change the robot's view of the environment. They also slightly influence the path of the robot but do not force the navigation system to reroute the robot from its default path topology. These challenges explicitly test the navigation system against various degrees of interaction with people in the benchmark.

We test the robustness of navigation systems with challenges in which people block the default path of the robot. This set of challenges consists of a person (8), a small group of people (9) or a person pushing a cart (10) hindering or interfering with the path

of the robot. These challenges test the ability of the navigation system to plan around moving obstacles that are not present in the initial setup of the environment. In addition to these obstacles, we propose challenges that aim to block the path of the robot to the next waypoint. This include closed doors (15) and people blocking the robot (14, 16, 17). These specific challenges last for different amounts of blocking time: either for a short time interval (14, 17) or for at least one minute (15, 16). Thereby, we test the ability of the system to cope with (prolonged) stalling situations. In addition to testing system stability and path planning, these challenges may reveal failure modes in certain implementations of localization methods when handling a standing robot.

6.3 Evaluation Methodology

In this section, we describe how we propose to evaluate and compare navigation systems with the benchmark. To establish comparability we first introduce a reference system. Subsequently, we describe the measures that we use to quantify performance. We can then compare different systems with the help of performance ratios between the tested systems and the reference system.

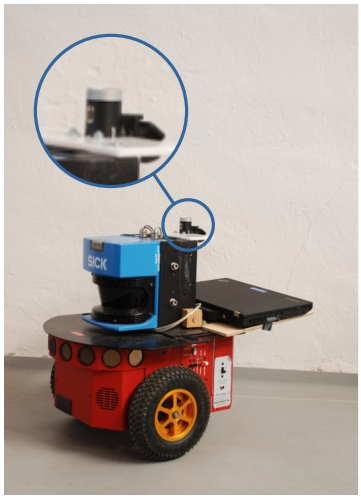
6.3.1 Reference robot and navigation system

For the baseline, we deploy the widespread commercial platform Pioneer P3-DX by adept mobilerobots (2015) as *reference robot* in the same environment, running a *reference navigation software*. The software builds on the ARNL navigation stack shipped with the Pioneer, and is available at `http://research.microsoft.com/brin/`. We use ARNL 1.7.5.1 and BaseARNL 1.7.5.2 and change from the default configuration only the parameters *SecsToFail* to 90, *GoalOccupiedFailDistance* to 500 and *UseSonar* to “false”.

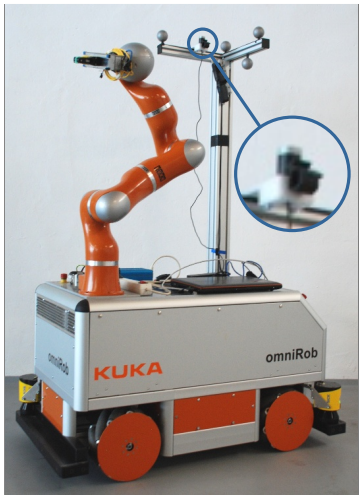
The reference robot will visit the same waypoints in the same order as the robot under evaluation. Thanks to the test grid introduced in the previous section it will also face the same challenges and configuration changes in a comparable manner. Figure 6.2 (left) shows one of the reference robots used in the experiments.

6.3.2 Ground truth evaluation

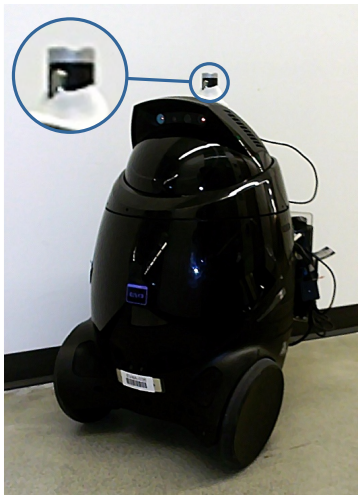
We employ the affordable ground truth system developed by Kikkeri et al. (2014) to automatically detect if and when the robot has successfully reached a waypoint. The system consists of visual markers placed on the ceiling and an up-facing camera mounted on the robot. A dedicated software component, independent of the navigation system, is responsible for capturing the images from the camera at the waypoints and for determining the positioning accuracy. The system requires an initial calibration in which the user



P3-DX (reference robot)



omniRob



MSR-P1

Figure 6.2: The robots used in our experiments. We equipped all robots with an additional up-facing camera for ground truth marker detection (zoomed-in circles). *Left:* The reference robot, a Pioneer P3-DX with a SICK LMS 200 laser scanner. *Middle:* The omniRob used in the environment ALU-FR. *Right:* The Microsoft Robotics Prototype 1 (MSR-P1), used in the environment MS.

manually drives the robot to the waypoints and registers their position within the reference software. The visual markers are black-and-white checkerboards printed on foam-boards, and thus cheap and disposable, see Figure 6.10. Whenever the robot reports an arrival at a waypoint, the ground truth system determines whether it actually reached the waypoint, the accuracy with respect to the marker and the time elapsed from the last waypoint. Using the ground truth system and the default path length between waypoints (see Section 6.3.3), we compute the following performance statistics:

- total number of failures,
- mean and maximum time to failure,
- mean and maximum distance to failure,
- average speed,
- accuracy at the goal.

The total number of failures is the number of segments in which the navigation system has been unable to arrive at a waypoint. The time to failure is the operational time between consecutive failures, counted from the last restart to the last successfully visited waypoint. We compute the distance to failure analogously. When testing navigation systems for

deployment in real application scenarios, the mean time to failure and, depending on the application, the accuracy at the goal are the most important performance measures.

6.3.3 Conducting the benchmark

The first step is to find a suitable environment in which to conduct the benchmark, fulfilling the requirements listed in Section 6.1.1 as accurately as possible. Then, the experimenter chooses the location of waypoints for the benchmark route and mounts optical markers for the ground truth system at these locations, see Section 6.3.2. Also, the experimenter needs to equip the reference robot and the system under test with an up-facing camera for the ground truth system.

The next step is to build a map of the environment with the system under test as well as with the reference system, typically by manually driving the robot around the environment to record sensor data. Once the reference system and the system under test have acquired a model of the environment, we can teach the waypoint locations to both navigation systems as well as to the ground truth software. We do this by driving the robots to each waypoint location and then recording the location in the map of the respective navigation system. At the same time we trigger the ground truth software to record a reference view of the fiducial marker from the waypoint location.

After deciding on the order in which the systems have to traverse the waypoints for every loop of the benchmark, the experimenter runs a first loop without any challenges. Here, the experimenter observes the default path of the robot and records the distance traveled for the respective path segments for the evaluation. Now, the experimenter can instantiate the test grid into a experiment script by defining concrete, repeatable configurations for all challenges in Table 6.1 and Table 6.2. For the challenges that will interfere with the default path of the robot, extra care is necessary to define meaningful locations. As a rule of thumb, we found it useful to mark configuration locations on the floor to ensure repeatability and to simplify instruction of extras during the benchmark, see Figure 6.4 for a map which shows some marked configuration locations. In addition to the challenge configurations, the experimenter also defines their time of occurrence in the respective loop. There might be several loop segments eligible for a specific challenge. For repeatability it is important that these events always occur on the same segment. To give an intuition, Table 6.3 shows an instantiation of loop 3 of the benchmark for one of our experiments. We describe the table in detail with the experiment in Section 6.4.1. To ensure that the designed experiment script is complete and feasible, we suggest to perform a dry run of the benchmark prior to conducting the actual experiment.

Finally the benchmark can begin: The system under test and the reference system have to perform all loops. With the gathered data, the experimenter can compute the performance statistics and also the performance ratios between the system under test and the reference system. When comparing benchmark results obtained at different

experimentation sites and with different systems under test, the performance measures for the reference system can help to judge whether the benchmarks have been run in the intended way. Values that greatly differ for the reference system between different benchmarks can indicate problems or reduced comparability of results.

6.4 Experiments

In our experiments we evaluated two different navigation systems on two different robotic platforms in two different experimental setups. Furthermore, two separate groups of people conducted the benchmark experiments. We prepared two environments for the experiments. We built the first setup (environment ALU-FR) in a large experimental area at the University of Freiburg, Germany. The second (environment MS) is a large, real office environment in the Microsoft Research building in Redmond, Washington, USA.

In the environment ALU-FR, we benchmarked the navigation system proposed in Chapter 2 and Chapter 3 of this thesis, combining optimization-based trajectory generation with accurate docking at target locations. As platform we used the previously introduced omnidirectional robot omniRob, shown also in Figure 6.2 (middle).

In the environment MS, we evaluated an in-house experimental Microsoft navigation software on the Microsoft Research Prototype 1 (MSR-P1) shown in Figure 6.2 (right). The robot performs both SLAM and motion generation by using only the Microsoft Kinect depth stream, gyroscope, and wheel odometry. In both environments, we have run the reference software on the reference platform Pioneer P3-DX, see Figure 6.2 (left) and Section 6.3.1.

6.4.1 Environment ALU-FR

We furnished the environment to make each dedicated area look realistic to the robots. This included tables, cupboards, chairs, couches and computers. In particular, we have used wooden panels to subdivide the environment and fixed the fiducial markers at the waypoints at a height of approximately 2.45 m. The complete environment measures 19 m × 12 m, the atrium 7.5 m × 11 m, the lounge 6 m × 9 m the office 5.5 m × 12 m, and the hallway is 7 m long. In Figure 6.3 we show several views of the prepared environment.

We instantiated the test grid from Table 6.1 and Table 6.2 into a concrete test script for our experiments. This is important to ensure that the test robot and the reference robot face the same challenges at the same time of each run. Figure 6.4 shows the laser-based occupancy grid map used for localization and motion generation of the omniRob along with the eight waypoints and some of the devised challenge positions. We specified a route by ordering the waypoints as follows: $0 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 0$. In this way, the robot always travels between different areas. Additionally, the travel distance

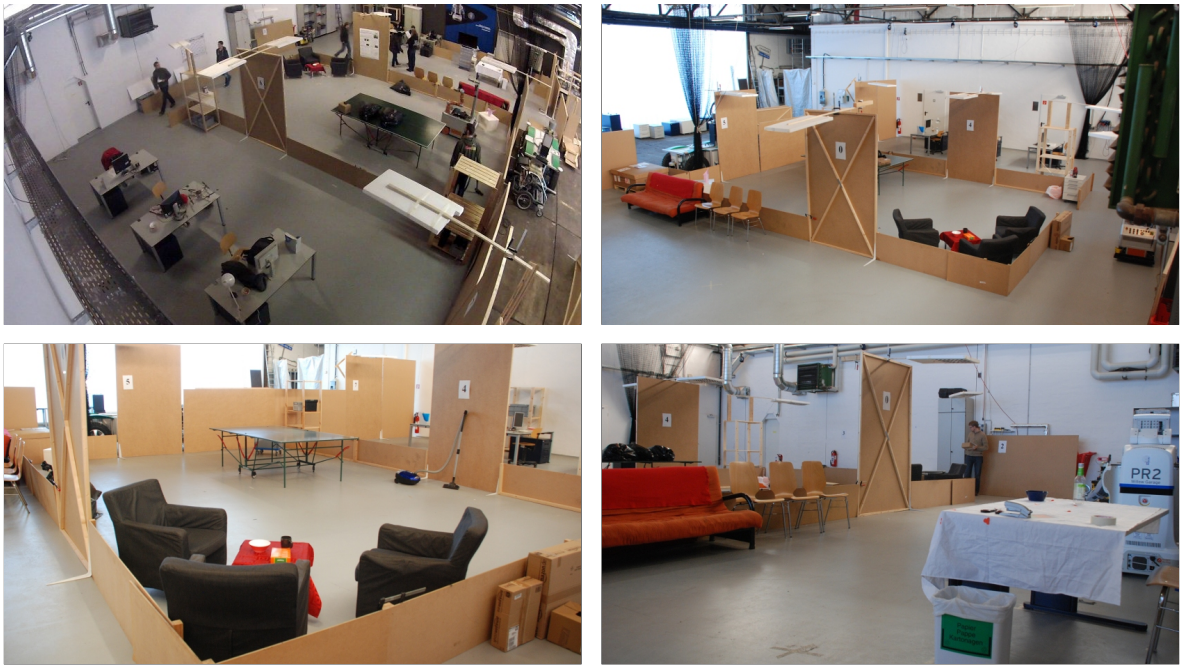


Figure 6.3: Overall views of the ALU-FR environment. Office (top-left), atrium (top-right) and detail views of the lounge (bottom-left) and the atrium (bottom-right).

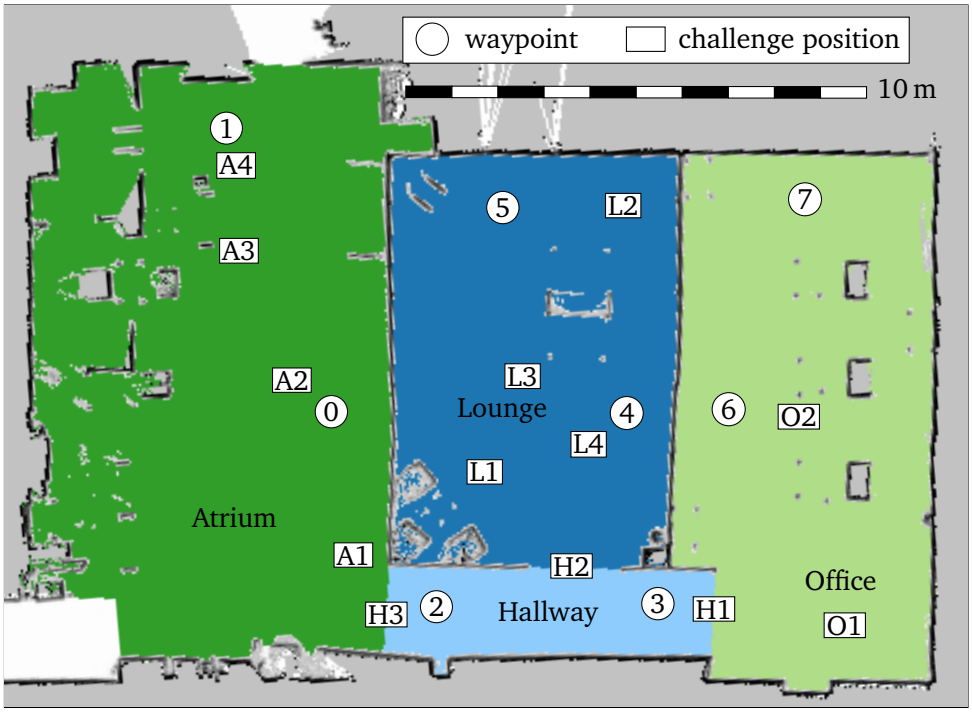


Figure 6.4: The occupancy grid map used for the omniRob experiments in the ALU-FR environment. The map indicates the four areas by color and also shows the locations of waypoints and some of the test grid challenges.

Table 6.3: Excerpt of the instantiation of the test grid (see Table 6.1 and Table 6.2) to an evaluation script for loop 3 of environment ALU-FR. It shows the specific challenges and their locations (see Figure 6.4) for each segment of the loop.

Loop Segment	Area	Challenge/Configuration	Row in Table 6.1, 6.2
0→2	Atrium	person with cart at A2	10
2→4	Hallway	2 boxes on the floor	11
	Hallway	2 people block at H2 for 1 min	16
	Lounge	move chairs by 0.2 m	18
	Lounge	cart at L3	20
	Lounge	1 garbage bag on the floor	22
4→6	Office	move chairs by 0.2 m	27
	Office	1 jacket on chair	28
	Office	2 bags next to desks	29
	Office	group of 4 people at O2	33
	Office	shelves 40% filled	31
6→3			
3→5	Lounge	group of 8 people at L1	25
	Lounge	person vacuuming at L2	24
5→1	Atrium	group of 4 people at A2	9
1→7	Atrium	person at A2	8
	Office	group of 8 people at O1	34
7→0			

between waypoints varies from short to long. We devised positions for people to gather at and move to. Marking these positions on the floor is helpful for ensuring repeatability in the experiments.

Creating an experiment script from the test grid in Table 6.1 and Table 6.2 requires particular care on how to design the challenges and which of them are possible to omit. The design of environment and challenges needs to ensure that a path exists for the robot. As the omniRob is larger than the reference robot, we had to increase the size of doors and hallways. The navigation system of the omniRob and the one of the reference robot are not based on vision sensors but instead make only use of laser range finders. Therefore, we omitted challenges which have no or only minor effects on laser range finders such as changing artificial lighting, opening/closing blinds, wall art changes, wall color changes, and whiteboard content changes, i.e., rows 1–4, 6, 7, 26 from Table 6.1 and Table 6.2.

Furthermore, we did not put ladders, tools, cables and the cart in the hallway because it is mechanically dangerous for the wheels of the omniRob that are designed for smooth

industrial-shop floors (rows 12, 13). For the same reason, we omitted the loose paper challenge (row 30). Moreover, we skipped the constant opening and closing of doors (row 5), the lounge coat racks (row 19), the janitor sign (row 21), modified the garbage bags to only be black (row 22) and limited the size of the biggest social gathering to eight people (rows 25, 34).

The test grid only defines the challenges per loop but not at which time in the loop they occur. It is up to the experimenter to define when the robot faces the challenges in each loop. Table 6.3 contains an excerpt of our experiment script. It shows all the travel segments for month/loop 3 of our test script that we derived from Table 6.1 and specifies which challenge configurations apply for each loop segment. It is a detailed instruction procedure for the experimenter on how to modify the environment during the evaluation to ensure repeatability of the experiments: For example, while the robot travels between $2 \rightarrow 4$, it encounters two parcel boxes in the hallway and two people block the door H2 for 1 min. All chairs in the lounge move by 0.2 m with respect to their position while mapping the environment. We place the cart of the lounge at L3 and put one garbage bag on the ground, see also Figure 6.4.

6.4.2 Environment MS

The second environment consists of several areas of the Microsoft Research building 99 in Redmond, Washington, see Figure 6.5. The atrium measures 25 m × 20 m, the lounge 20 m × 12 m, the office 10.5 m × 7.8 m and the hallway 17 m × 1.75 m. Figure 6.6 shows the grid map used by the reference navigation system for this environment. This environment includes an open floor plan in the atrium and lounge areas. It has substantial daylight coming in through the glass ceilings and the entrance. The lounge area includes a coffee shop, with multiple round tables and chairs, as well as tall rectangular tables with high chairs, couches and armchairs. The areas have carpet, linoleum, rough tile and hardwood as floor surface.

Where practical, we chose the waypoint locations close to interesting or meaningful locations when creating the test script for this environment, such as adjacent to the coffee stand, in front of the elevators and near the receptionist desk. Figure 6.6 shows the waypoint locations that the robots had to traverse in the order $6 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 0 \rightarrow 6$ for each loop. The figure also shows an artificial barrier that we inserted into the ARNL map of the reference system using the “forbidden area” functionality of the Adept Mapper 3 software. During setup testing, the reference robot would sometimes plan a path from waypoint 1 to 3 that would not go through the lounge as expected/required, but instead first go around the lounge and then to the atrium. We designed the forbidden area such that it would not affect the other paths through atrium during any of the other challenges.

The environment included a doorway between the hallway and the office as well as

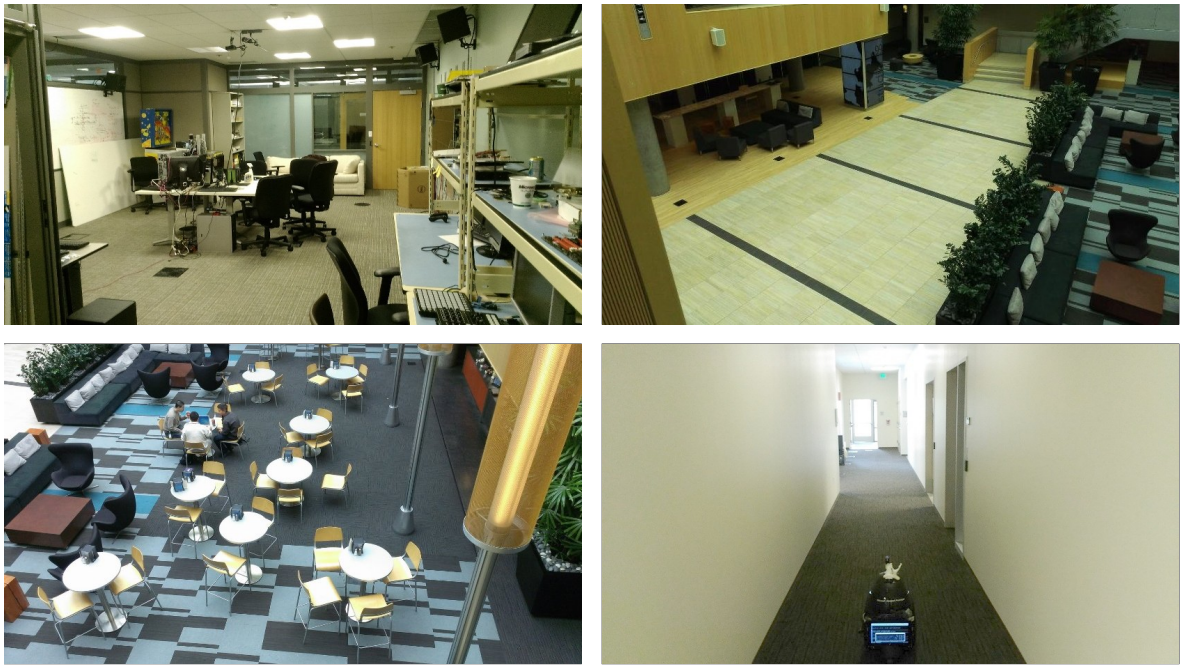


Figure 6.5: The four areas of the environment MS: office (top-left), atrium (top-right), lounge (bottom-left) and hallway (bottom-right).

one additional doorway into an unmapped adjacent space that was alternately opened or closed for each loop. The Microsoft robot and the reference robot are both small enough to allow motion in about the same spaces as an average person, so we instantiated all challenges including doorways, ladders, tools, cables, boxes, carts, etc. in the experiment script. When placing the ladder and cables for challenge 13 at position 1, we taped a 1.25 cm thick cable to the floor in the path of robot using gaffer's tape. For challenge 13 at position 2, we blocked the default path with a ladder, toolbox, and a large mound of thick cable that reached about 0.2 m from the ground.

As we used a Microsoft Kinect depth sensor for navigation, we omitted the challenges involving lighting or appearance changes from the script, including rows 1–4, 6, 7, and 26 from Table 6.1 and Table 6.2. No shelf was available for the office, so we omitted challenges 31 and 32. Due to a lack of the required number of people we also omitted challenges 25 and 34. To avoid non-replicable disturbances by direct sunlight or non-scripted interactions with people, we started the experiments in the evening.

6.4.3 Results

We list the performance of the different systems in the two environments in Table 6.4 and Table 6.5, respectively. The last column of each table shows the *relative* performance of a navigation system with respect to the respective reference system. Thanks to the

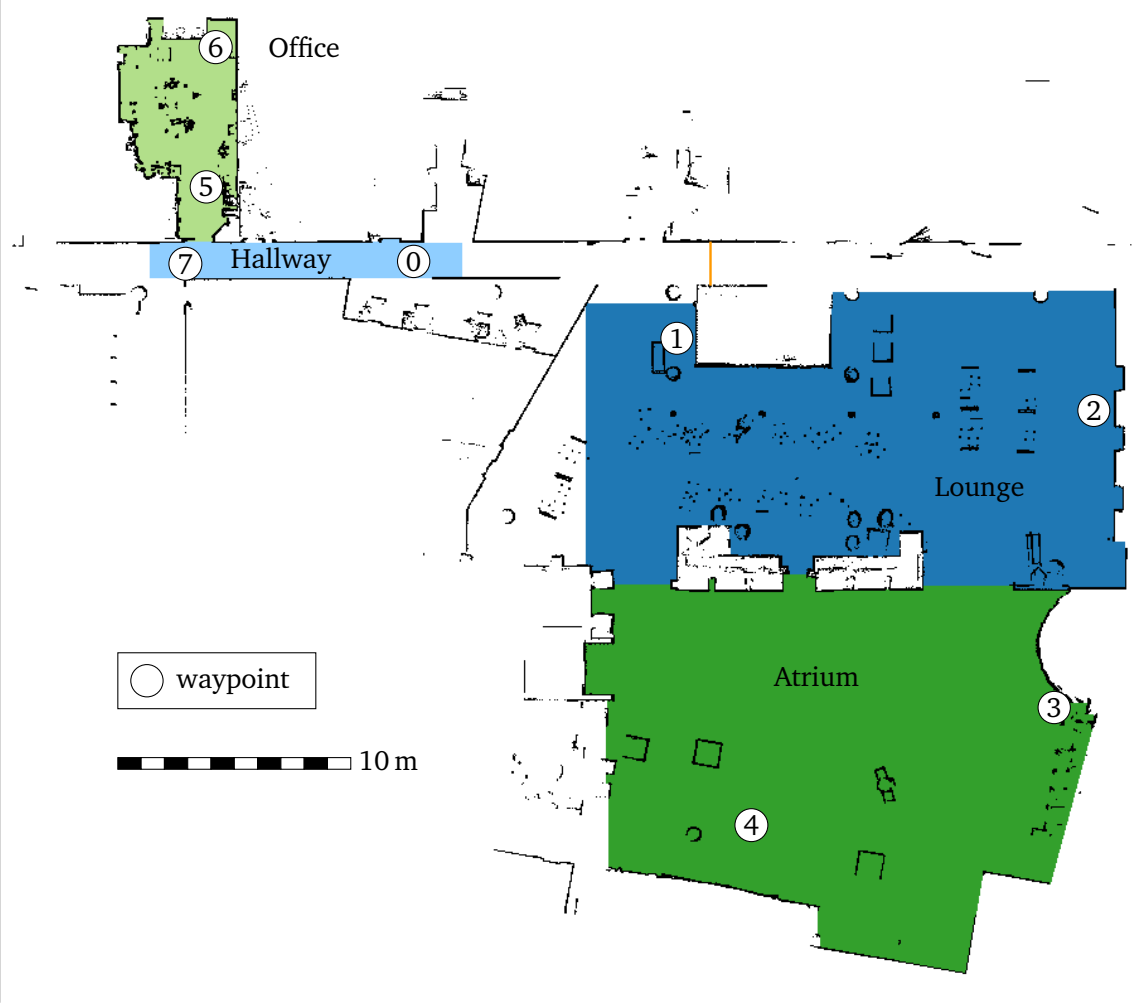


Figure 6.6: The occupancy grid map used by the reference navigation system for the experiments in the MS environment. The map marks the four areas by color and also shows the locations of the waypoints. The orange line above the lounge area is a virtual barrier that keeps the reference system from traveling outside of the intended benchmark area.

Table 6.4: Benchmark results in the environment ALU-FR.

Performance	ALU-FR	Reference	Ratio
Number of failures	0	0	-
Mean time to failure	-	-	-
Maximum time to failure	4343 s	5125 s	0.85
Mean distance to failure	-	-	-
Maximum distance to failure	1423 m	1349 m	1.05
Average speed	0.33 $^{m}/_{s}$	0.26 $^{m}/_{s}$	1.27
Average positioning error	0.005 m ± 0.007 m	0.05 m ± 0.04 m	0.10

benchmark protocol, it is now possible to determine how accurate a system is with respect to a standardized baseline and environmental conditions.

Environment ALU-FR In environment ALU-FR, neither the omniRob nor the reference system failed during the circa 1.5 km of navigation, which they performed in circa 70 min. In environment ALU-FR, the robots can always observe sufficient structure to properly localize themselves. Figure 6.7 shows the waypoint configurations and an overlay of all driven paths by the omniRob system for the experiment in environment ALU-FR. The larger deviations of some paths from the majority of the paths correspond to reactions of the system to the path blocking challenges. Figure 6.8 shows a snapshot of the omniRob traveling the segment 3 → 5 in loop 3 of the benchmark. The top part of the figure shows how the challenges change the robot's perception of the environment while the bottom part provides a video frame of the situation. As defined in the experiment script in Table 6.3 for this loop segment, the robot is facing a group of people and a person vacuuming at the defined positions, see also Figure 6.4. The state of the cart, the trash bags and the slight misalignment of the chairs (lower left corner of lounge in the grid map) remain unchanged from the previous visit to the lounge in this loop, see the travel on segment 2 → 4 in Table 6.3.

The results in Table 6.4 show that the omniRob was driving faster than the reference system and reached a higher accuracy at the waypoints. The higher accuracy at the goal is due to the explicit docking we perform with this system at the goal location, see Chapter 3.

Environment MS In environment MS, the experiments covered circa 2.1 km and took 6 hours to conduct for each robot. Both robots, the MSR-P1 and the reference system encountered failures, as shown in Table 6.5. The failures modes for the MSR-P1 robot were a software problem (1 failure), localization inaccuracies (3 failures), and localization divergence (1 failure). We attribute the localization failures to the limited range of the Kinect sensor in the wide spaces of the atrium area.

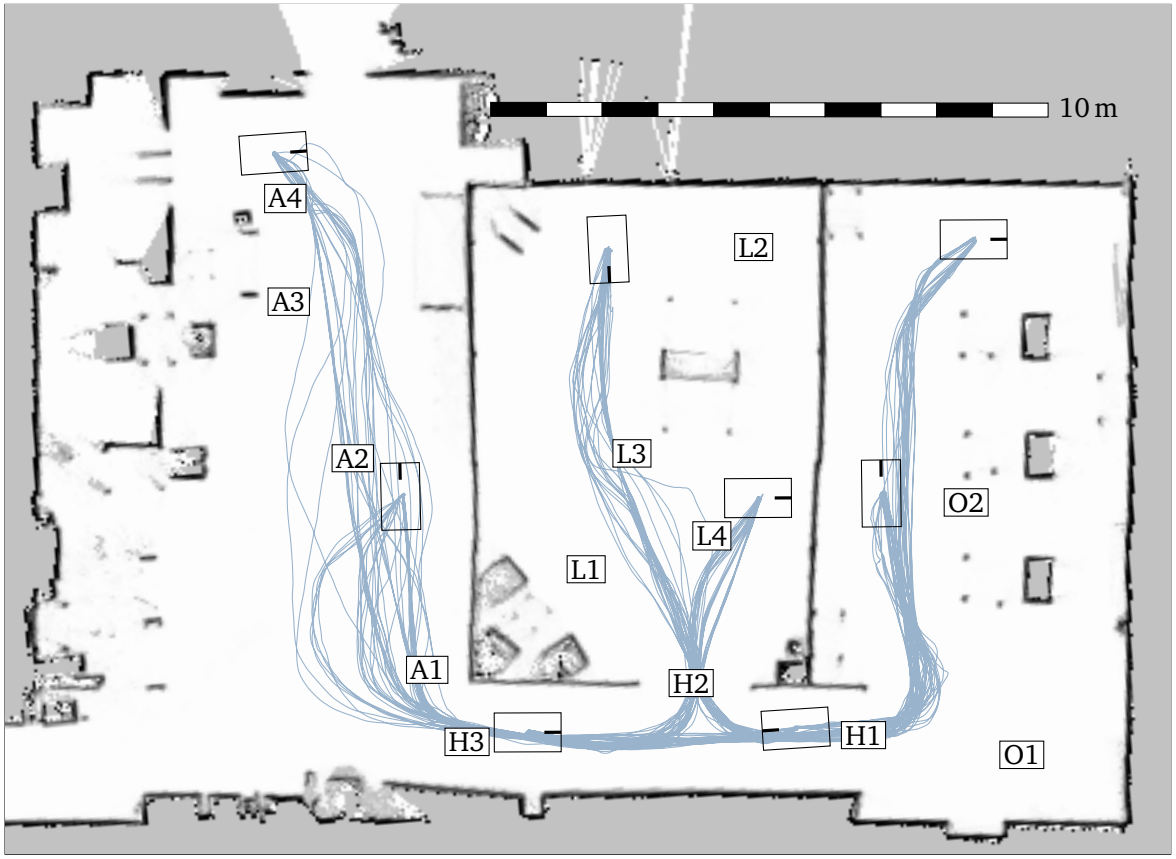


Figure 6.7: The waypoint configurations and all paths driven during the benchmark by the omnirob in the environment ALU-FR. The figure shows the robot position as reported by the localization component of the navigation system during the circa 1.5 km that the robot traveled in approximately 70 min. Individual robot paths deviate from the more frequently taken default path to cope with certain challenges, see for example locations A1, A2, or L1.

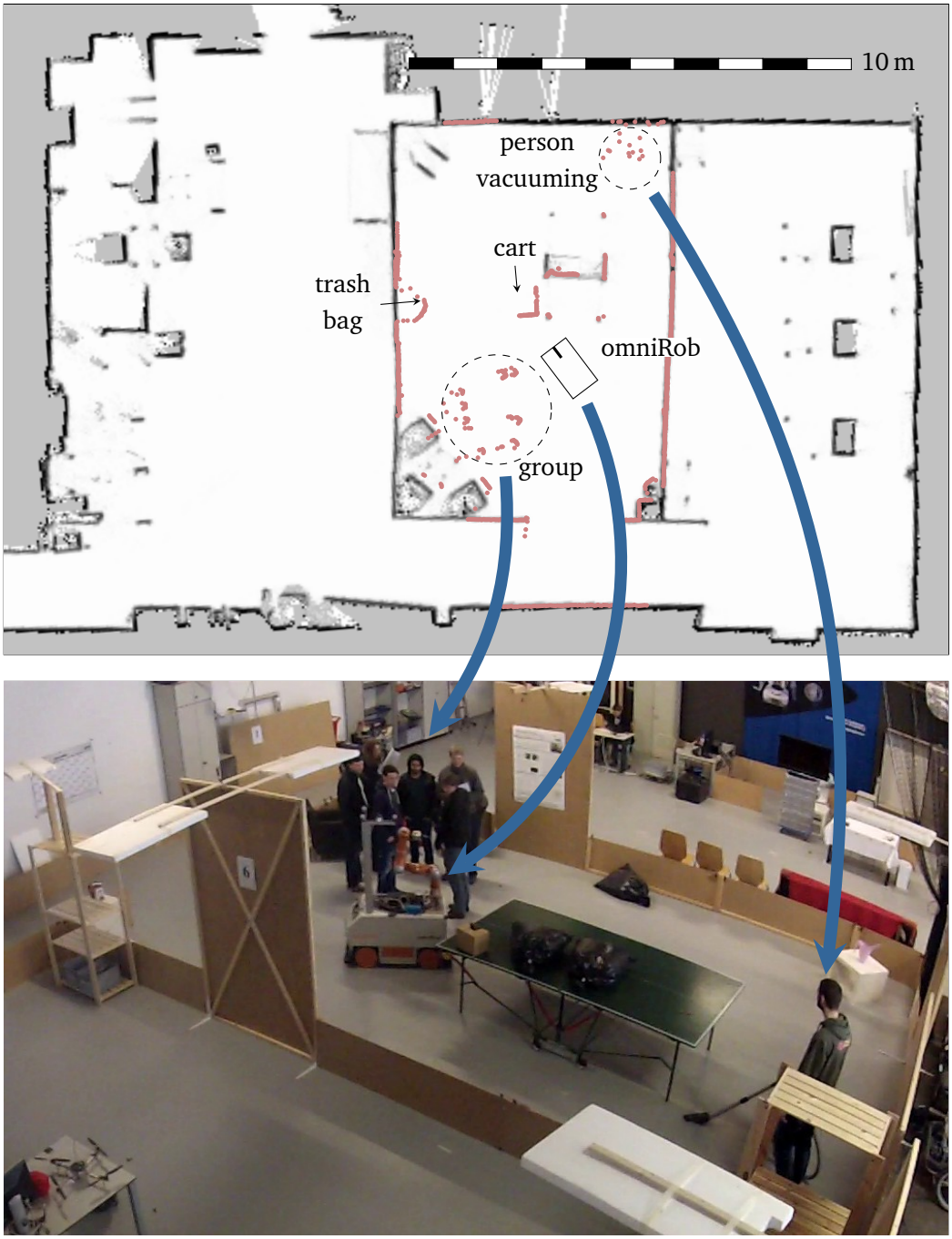


Figure 6.8: The omniRob traveling the segment 3 → 5 in benchmark loop 3 in environment ALU-FR. The top part shows the position of the robot as estimated by the localization module and the current readings of the laser scanners in light red. The bottom part shows a picture of the same situation. The omniRob is facing the challenges as determined in the experiment script for this loop, see Table 6.3 and compare Figure 6.4 for the challenge locations. The table occludes the cart in the picture.

Table 6.5: Benchmark results in the environment MS.

Performance	MS	Reference	Ratio
Number of failures	5	9	0.56
Mean time to failure	2265 s	726 s	3.12
Maximum time to failure	5023 s	1971 s	2.55
Mean distance to failure	367 m	183 m	2.01
Maximum distance to failure	860 m	472 m	1.82
Average speed	0.16 $^{m}/_{s}$	0.25 $^{m}/_{s}$	0.64
Average positioning error	$0.23\,m \pm 0.2\,m$	$0.22\,m \pm 0.1\,m$	1.05

The reference robot encountered failures due to software problems (1 failure), localization inaccuracies (1 failure), localization divergence (1 failure), faulty obstacle perception (3 failures), path oscillation for more than 5 minutes (1 failure), not finding a path around a new obstacle (1 failure) and not detecting a low obstacle (1 failure). The localization failures of the reference robot correspond to a known bug in the current ARNL version that the upcoming version 1.8 is going to fix. The failures occurred in the long hallway and the open space of the atrium.

In two situations, the path computed by the reference system oscillated between two alternatives for a long time. In the first situation this occurred between cluttered lounge chairs and lasted for more than 5 minutes, leading to a failure. The second situation involved a group of people but resolved itself after 4 minutes, therefore not leading to a failure. In the failure that resulted from not detecting a low obstacle the reference robot got caught in cables on the floor. This challenge was successfully negotiated by the MSR-P1 robot that was able to detect the obstacles on the floor. Overall, the MSR-P1 system drove more reliably than the reference system with a comparable goal accuracy and at a lower speed as indicated by the ratio values in Table 6.5. The benchmark revealed defects in several key areas of navigation including planning, localization, static and dynamic obstacle avoidance, reactive re-planning, remapping, and endurance, consistent with the limitations of each software.

Three months prior to the experiments in environment MS, we conducted a stripped down version of the benchmark with older MSR-P1 navigation software. Table 6.6 lists the performance corresponding to this pre-test. We found that in the full benchmark the MSR-P1 showed dramatic improvements with respect to the pre-test (5 failures vs. 12), consistent with the improvements in navigation software done in the meantime. We also found that the reference system performed worse in the full benchmark (9 failures vs. 5). This before and after experiment confirms the ability of the proposed benchmark to expose the effects of both software and environmental changes.

We believe the results accurately reflect the capabilities and performance of all tested systems. In our observation this is primarily due to the wide coverage of possible failure

Table 6.6: Benchmark results for the pre-test in the environment MS.

Performance	**MS Pre-test**	**Reference**	**Ratio**
Number of failures	12	5	2.40
Mean time to failure	373 s	1589 s	0.23
Maximum time to failure	1109 s	2547 s	0.43
Mean distance to failure	54 m	407 m	0.13
Maximum distance to failure	185 m	674 m	0.27
Average speed	0.14 $^{m}/_{s}$	0.25 $^{m}/_{s}$	0.50
Average positioning error	$0.47\,m \pm 0.21\,m$	$0.09\,m \pm 0.09\,m$	5.22

modes. Moreover, the amount of challenges in our protocol seemed appropriate. The relatively small cumulative runtime seems sufficient to capture a good performance representation. However, as navigation systems get better, one might need to increase the total runtime.

Using the ratios between the tested systems and the reference system, we can compare their normalized performance benchmarked on different robots at different times and in different experimental settings. Figure 6.9 compares these ratios with a bar plot, the data corresponds to the ratio columns in Table 6.4, 6.5, 6.6 and the dashed line marking a ratio of 1 in the plot corresponds to the performance of the reference system. The plot shows how the docking module of the ALU-FR navigation system yields a substantially lower positioning error at waypoints. One can also see the drastic improvements of the MS system from the pre-test to the full benchmark, especially in mean time to failure and positioning error at the waypoints. This pre-test allowed the MS group to promptly detect the major failure modes of their initial navigation system and to develop an improved version in less than two months. We believe that such quick improvements would not have been possible without a valid and rigorous evaluation benchmark like the one we propose in this chapter. Moreover, the possibility of repeatable evaluation allowed a rigorous quantitative characterization of the improvement in performance, which would not have been possible without the normalization with respect to the reference system.

The different number of failures for the reference system in environments ALU-FR and MS advises to use some caution when comparing the results. The benchmark in environment MS was harder for the reference system than the one in environment ALU-FR, which we attribute to a large degree to the fact that the environment ALU-FR needed to accommodate for the larger footprint of the omniRob system. With its smaller footprint the reference system therefore had wide space to maneuver. Nevertheless, the normalization and the strict scripting of environment dynamics enable us to establish comparability. Here, the similarity of reference system performance across different experimentation sites acts as an indicator for the admissibility of the comparison.

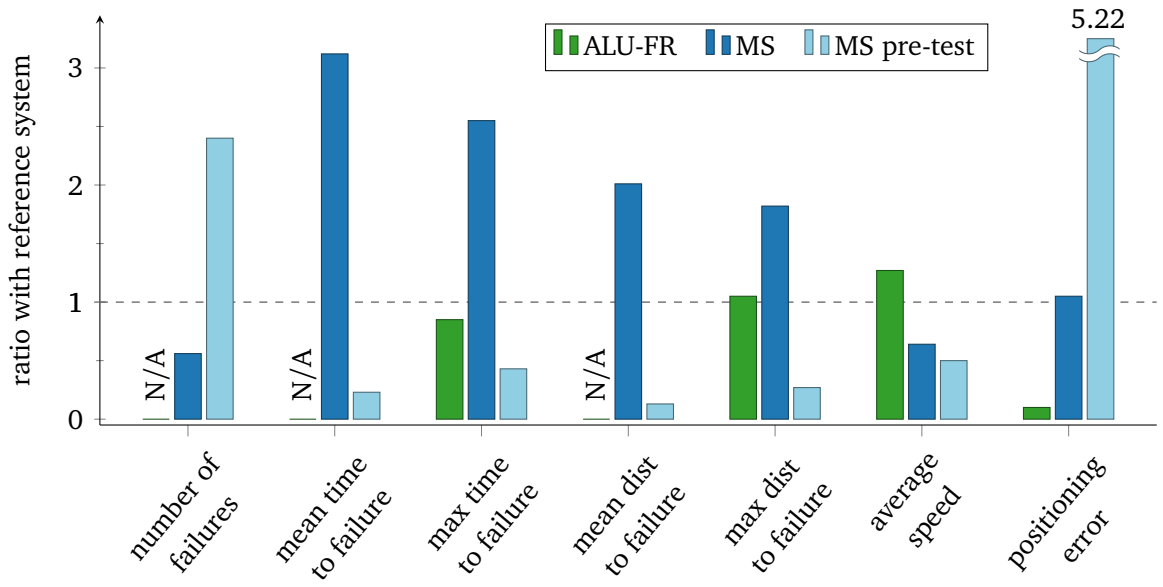


Figure 6.9: Normalization with respect to the reference system establishes comparability of results between navigation systems tested on different robots in different environments. The plot shows the ratio between each tested system and the reference system in the respective environment for a number of performance measures, compare the “Ratio” columns in Table 6.4, 6.5, 6.6.

6.4.4 Discussion

Comparing autonomous navigation solutions based on their performance in real environments is an arduous task. We have identified three important aspects to consider during the setup of the experiments and their evaluation.

A first aspect concerns the comparison of different systems at different locations. The reference robot is instrumental in providing a sense of the complexity of each environment. However, one must consider that the shape and the size of the robot has a certain degree of influence on the results. The chosen benchmark targets navigation in office environments, thus slightly favoring small and circular robots. When the system under test differs from the reference robot in size, shape or even locomotion principles, the environment and the protocol should be slightly adapted to allow a fair comparison. This happened, for instance, when we evaluated the omniRob system, as described in Section 6.4.1.

A second aspect lies in the fiducial-based ground truth system that determines whether the robot successfully reached a waypoint. The location of the camera on the robot is very important as the relative distance between the waypoint markers and the camera defines the *success range* for the system. A larger relative distance between the markers and camera allows successful detection from further away, see Figure 6.10.

A third aspect of more practical nature is to arrange the experiment script in a way

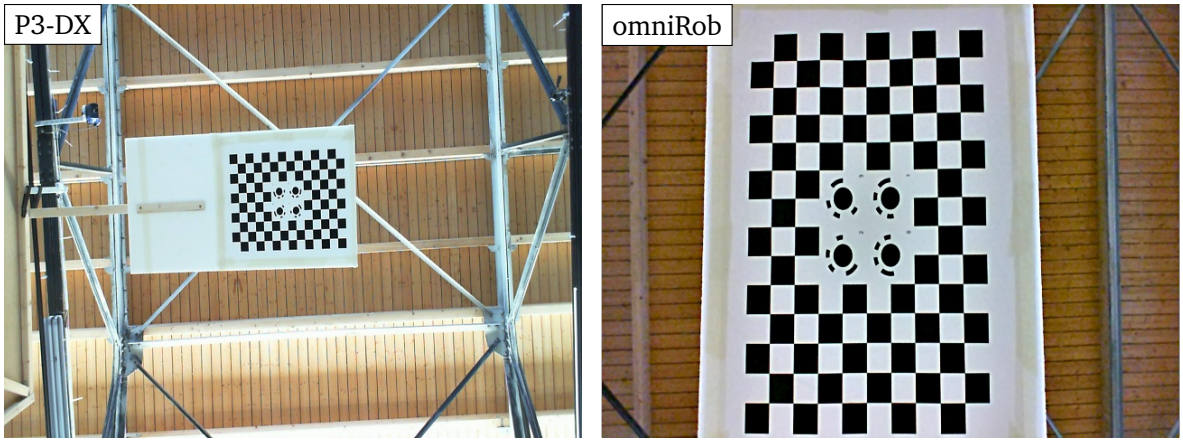


Figure 6.10: The camera mounting influences the tolerance for ground truth marker detection. The pictures show camera views for waypoint 1 in environment ALU-FR. The camera is mounted at a height of 0.45 m on the reference robot P3-DX, see also Figure 6.2 (left). On the omniRob, the camera is mounted rotated by 90 deg with respect to the camera of the reference robot and at a height of 1.7 m, see also Figure 6.2 (middle).

that requires large groups of people at a specific time and not spread out during testing. We found that it is a considerable effort to manage a large number of idle people over extended periods of time.

An important insight is that the benchmark helps to understand failure points of navigation systems. This happened with the pre-test of MSR-P1 in the MS environment. The issues discovered in the first pre-test benchmark helped to dramatically improve the performance on the next test.

Another insight stems from the localization bug of the reference system in the MS environment. In addition to stressing the importance of strictly defined software versions for the reference system, this shows that large open areas and hallways are an important factor to capture localization failure modes. The ALU-FR environment was probably too small to elicit the failure.

6.5 Related Work

Benchmarking plays an important role for comparison and evaluation in science. In particular, there are many benchmarking works in several fields related to robotics, including machine learning, computer vision and artificial intelligence. Machine learning is probably the field that received most attention, thanks to the use of very large evaluation datasets for different tasks, see for example the works by Bennett and Lanning (2007); Pang and Lee (2008); Bache and Lichman (2013). Similarly, computer vision has many procedures

and benchmarks available, e.g., the works by Deng et al. (2009) and Everingham et al. (2010) as well as the CAVIAR dataset (CAVIAR, 2004) and the PETS dataset (Ferryman and Shahrokni, 2009).

Despite being one of the most studied field in robotics, there is only a relatively small amount of literature related to benchmarking robot navigation. The reason is probably that one cannot evaluate complete robot navigation systems on a dataset. During navigation the robot interacts with the environment and its future sensor readings depend on past actions. In NaviGates, Knotts et al. (1998) present an early benchmark for robot navigation. Here, they concentrate on robot skills and architecture but they do not take into account how to systematically evaluate the robot performance in a changing environment. Gutmann et al. (1998) presented a set of extensive experiments evaluating the accuracy and robustness of localization systems using datasets. Calisi et al. (2008) propose a benchmark framework that concentrates only on the evaluation of vehicle motion algorithms. Borenstein and Feng (1995) introduce a method for measuring odometry errors of mobile robots. Specifically, their method focuses on quantitative evaluation of systematic and non-systematic errors. The work of Nowak et al. (2010) presents an investigation for an evaluation of two specific robot tasks, namely path planning and obstacle avoidance. This work considers a static environment. Dillmann (2004) and Del Pobil et al. (2007) survey efforts in quantification for a set of robot tasks, including robot cleaning, robot rescue and autonomous driving.

Another way of evaluating navigation systems is to let them compete in a challenge such as the DARPA urban challenge (DARPA, 2007) or the robocup competitions (Kitano et al., 1997a,b), for which specialized leagues emerged over the years, e.g., robocup rescue (Kitano and Tadokoro, 2001), robocup @home (Wisspeintner et al., 2009) or robocup logistics (Niemueller et al., 2013). However, such challenges typically require to transport all robot systems to one location and their outcome is more a ranking of systems rather than a complete analysis. In a challenge, the experimental setup is mostly identical for all evaluated approaches. For some competitions the specific experimental environment is also standardized beforehand, allowing to replicate the environment in research groups. However, this typically only covers static aspects of the environment. Meaningful comparisons that include dynamically changing environments still require participation at the challenge.

6.6 Conclusion

With this chapter, we introduce a novel experimental protocol to evaluate a robot navigation system as a whole in a repeatable fashion. We describe a ground truth system and a detailed procedure on how to perform such an evaluation and provide the necessary tools to the community. To ensure repeatability, we introduce the concept of challenges as

a paradigm to define a detailed set of environment changes and dynamics. We further introduced the concept of a reference platform equipped with a reference navigation system. This allows different groups at different sites and with different platforms that are equipped with different sensors to conduct comparable experiments. While the proposed benchmark protocol is useful to compare and assess complete navigation systems and their component interplay for application scenarios, it has also proven beneficial in the development process of new navigation systems.

In the context of navigation for industrial automation this benchmark protocol provides a methodology for meaningful and realistic performance assessments and comparisons of candidate navigation systems. It also provides key metrics such as the mean time to failure and the accuracy at the goal. The scope of the benchmark as presented in this chapter is rather broad, it applies to indoor navigation in general. As discussed in this chapter, the challenges can be easily reduced, adapted and amended for more constrained or structured application domains such as shop floor navigation.

7 Conclusion

In this thesis, we proposed several contributions to the field of mobile robot navigation. We focused our contributions around the challenge of enabling automated vehicles to flexibly navigate in the context of industrial automation. For all proposed approaches, we presented theoretical descriptions, discussed related works and conducted extensive experiments.

We first proposed a complete navigation system for omnidirectional robots in industrial settings. Our system exhibits high navigation accuracy, i.e., the behavior of the robot closely matches its planned motions. We achieve this accuracy through smooth, curvature continuous trajectories that account and plan for constraints on the velocities and accelerations of the vehicle. This property ensures the safety of the vehicle and its environment. Our trajectories also incorporate constraints like obstacle-dependent velocity limits. This yields anticipatory behavior like smooth deceleration when approaching narrow passages or obstacles which increases the acceptance and perceived safety by human personnel sharing the workspace. At the same time, our navigation system employs a trajectory optimization scheme that leverages the high maneuverability of omnidirectional robots and ensures fast travel time to the target location. The efficiency of the optimization allows frequent updating of planned trajectories to react to changes in the environment. Altogether, these properties make our system well-suited for autonomous navigation on the factory floor.

When reaching their target location, autonomous vehicles in industrial applications typically have to exchange loads or dock to machinery to successfully connect individual processes to a process chain. This requires high positioning accuracy of the autonomous vehicle at its target location. To remove the need for the additional infrastructure typically expended for this purpose, we proposed a highly accurate localization system that only relies on the mandatory safety scanners for mobile robots. Nonetheless, we demonstrated docking accuracies of a few millimeters in our experiments. The integration of this system with our proposed navigation system yields highly flexible and accurate navigation for industrial automation.

There are application scenarios in which instead of fully autonomous motion strict adherence to predefined paths is the behavior of choice for automated vehicles. Possible reasons include a high cost of encoding rule sets and constraints, e.g., for dedicated vehicle driving lanes, or concerns and expectations of decision makers and human workers sharing the shop floor. To accommodate such scenarios, we proposed a method to intuitively

instruct mobile robots with predefined paths through user demonstrations. We fit the internal path representation of our navigation system to the user demonstration and thereby extend our navigation system with a mode that robustly follows predefined routes, still relying only on the mandatory safety scanners and wheel encoders. We thereby achieve a very flexible system that does not rely on additional infrastructure and is quick and intuitive to instruct and reconfigure, also for route following tasks. In addition to fully autonomous motion and strict route following, one can also orchestrate the components of our system to realize different levels of autonomy, e.g., to autonomously circumnavigate obstacles on a predefined route within a given bound.

The navigation system proposed so far relies on a previously built, globally consistent map of the environment. This means that prior to instruction of the mobile robot with a concrete task, the user has to build a representation of the environment in an initial mapping phase. For scenarios that only require predefined routes, we proposed a teach-and-repeat framework that eliminates the time-consuming mapping phase when instructing mobile robots to follow predefined paths. Through extension of the ideas from our high accuracy docking system towards time-varying reference poses, our system achieves tracking of user-demonstrated trajectories with millimeter accuracy. Our system enables non-expert users to define virtual rails for automated vehicles through a single, intuitive demonstration. The absence of additional infrastructure and a mapping phase make this system highly flexible and efficient, especially for application scenarios in which tasks and environment are subject to frequent and substantial changes.

Actual deployment of a navigation system in real-world application scenarios requires prior knowledge about the performance, robustness and reliability of such a system. Here, the overall performance of the complete system is critical. It is not sufficient to perform evaluations of individual components like mapping, localization or path planning, which are suitable for benchmarking with recorded sensor data. Instead, we need to assess all components and their interplay during interaction with a realistic environment. Therefore, we proposed a benchmark protocol for indoor navigation that serves two purposes. First, it assesses the performance of a complete navigation system in a standardized way that also controls the dynamics of the test environment. Second, the use of a reference navigation system establishes comparability between benchmark results obtained with different systems on different robots. The reference system normalizes the benchmark performance such that also results obtained in different instantiations of the test environment become comparable. With the proposed benchmark protocol, we provide the necessary tool to decide whether a specific navigation system is suitable for deployment and to choose between different alternative systems.

In summary, we proposed in this thesis a complete navigation system for industrial applications as well as the means to evaluate this system and alternative systems. Our system emphasizes ease of use and flexibility and is capable of highly precise, fully autonomous motion that accounts for environment constraints. The proposed approaches

also apply to constrained motion along predefined routes and their combination can realize multiple levels of autonomy. They use only mandatory safety scanners and wheel encoders as sensors and therefore do not require any additional infrastructure. Through our experimental evaluation, we believe that our contributions bring flexible navigation closer to deployment in real-world industrial settings. As a first visible impact, KUKA Roboter GmbH ships a version of the proposed navigation system with the omniRob mobile manipulator.

Outlook There are several areas in which further research can extend the scope of the proposed navigation system. As described in this thesis, our navigation system is in general applicable to any robot that can turn on the spot, e.g., omnidirectional robots and differential drive robots. To extend the system to drive types that require a minimum turning radius like the Ackermann drive, adaptions are necessary to the path model and the planner that provides the initial path for the optimization.

Our system for highly accurate localization is in general agnostic of the drive type of the robot. However, if the robot is to correct the reported offset to the target pose, a controller needs to perform this compensatory motion. A lateral offset to the target configuration is inherently easier to correct for with an omnidirectional robot. For a differential drive, the maneuvers necessary to compensate a lateral offset might introduce a relatively large error themselves. Still, highly accurate motion is possible along predefined paths as shown in our experiments for the teach-and-repeat framework with a differential drive robot. A promising area for further research is therefore a controller that compensates for a lateral offset with a larger longitudinal motion while keeping track of the target configuration.

To account for changes in the environment, our navigation system overlays the current safety scanner readings on its map of the environment. This discards older readings and can lead to situations in which the system is no longer aware of an earlier perceived blocking of a passage. To make the system more robust against such situations, it is promising to integrate research results from the area of life-long mapping to consistently integrate the stream of environment perceptions without deteriorating the map. A special case of changes in the environment are other vehicles that follow predictable motion patterns. Here, it would be beneficial to inform the navigation system about the predicted motion. The navigation system could then exploit this information to avoid interpreting vehicles traveling in its front as blockages.

Our experimental evaluations showed a potential bias of the proposed benchmark protocol when the size of the footprint differs substantially between the system under test and the reference system. Since the environment needs to accommodate the dimensions of the larger robot, the navigation tasks automatically become easier for the smaller robot. Further development of the benchmark to simulate an appropriately larger footprint for the smaller robot could help to eliminate this bias.

List of Figures

Bibliography

P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, 19:1, 2007.

adept mobilerobots. `http://mobilerobots.com`, 2015. Online, accessed 2015-08-18.

T. Albrecht. Automated guided vehicles with laser navigation. Fraunhofer-Institut für Materialfluss und Logistik IML, Dortmund, `http://www.iml.fraunhofer.de/content/dam/iml/en/documents/OE140/Flyer_AGV_with_Lasernavigation_20110606.pdf`, 2011. Online, accessed 2015-08-18.

J. Aleotti and S. Caselli. Robust trajectory learning and approximation for robot programming by demonstration. *Robotics and Autonomous Systems*, 54(5):409–413, 2006.

H. Andreasson, A. Treptow, and T. Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

A. A. Argyros, K. E. Bekris, S. C. Orphanoudakis, and L. E. Kavraki. Robot homing by exploiting panoramic vision. *Autonomous Robots*, 19(1):7–25, 2005.

K. Bache and M. Lichman. UCI machine learning repository. University of California, Irvine. `http://archive.ics.uci.edu/ml`, 2013. Online, accessed 2015-08-18.

P. Baiget, E. Sommerlade, I. Reid, and J. Gonzáles. Finding prototypes to estimate trajectory development in outdoor scenarios. In *Int. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, September 2008.

D. J. Balkcom, P. A. Kavathekar, and M. T. Mason. Time-optimal trajectories for an omni-directional vehicle. *Int. Journal of Robotics Research*, 25(10):985–999, 2006.

E. T. Baumgartner and S. B. Skaar. An autonomous vision-based mobile robot. *IEEE Transactions on Automatic Control*, 39(3):493–502, 1994.

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

J. Bennett and S. Lanning. The netflix prize. In *KDD cup and workshop at the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Minig*, 2007.

M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In *European Robotics Symposium*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 143–157, 2006.

P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47:69–77, 2004.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Springer handbook of robotics*, pages 1371–1394. Springer Berlin Heidelberg, 2008.

J. E. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. Journal of Robotics Research*, 4(3):3–17, 1985.

J. Borenstein and L. Feng. Umbmark: A benchmark test for measuring odometry errors in mobile robots. In *Proc. SPIE 2591, Mobile Robots X*, 1995.

J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *Int. Journal of Robotics Research*, 21(12):1031–1052, 2002.

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

A. Byravan, B. Boots, S. Srinivasa, and D. Fox. Space-time functional gradient optimization for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard. Learning and reproduction of gestures by imitation: an approach based on Hidden Markov Model and Gaussian mixture regression. *IEEE Robotics and Automation Magazine*, 17:44–54, 2010.

D. Calisi, L. Iocchi, and D. Nardi. A unified benchmark framework for autonomous mobile robots and vehicles motion algorithms (MoVeMA benchmarks). In *RSS Workshop on experimental methodology and benchmarking in robotics research*, 2008.

CARMEN. Robot navigation toolkit. `http://carmen.sourceforge.net`, 2006. Online, accessed 2015-08-18.

CAVIAR. Data sets. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR`, 2004. Online, accessed 2015-08-18.

A. Censi. An accurate closed-form estimate of ICP's covariance. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

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

A. Cherubini and F. Chaumette. Visual navigation with a time-independent varying reference. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

A. Cherubini and F. Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *Int. Journal of Robotics Research*, 32(2):189–205, 2013.

A. Cherubini, F. Chaumette, and G. Oriolo. Visual servoing for path reaching with nonholonomic robots. *Robotica*, 29(07):1037–1048, 2011.

D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *Int. Conf. on Pattern Recognition*, 2002.

H. Cho and S. Kim. Mobile robot localization using biased chirp-spread-spectrum ranging. *IEEE Transactions on Industrial Electronics*, 57(8):2826–2835, 2010.

J. Connors and G. Elkaim. Manipulating B-Spline based paths for obstacle avoidance in autonomous ground vehicles. In *National Meeting of the Institute of Navigation*, 2007.

DARPA. urban challenge rules. `http://archive.darpa.mil/grandchallenge/docs/Urban_Challenge_Rules_102707.pdf`, 2007. Online, accessed 2015-08-18.

A. P. Del Pobil, R. Madhavan, and E. Messina. Benchmarks in robotics research. In *IROS Workshop on Benchmarks in Robotics Research*, 2007.

F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. Journal of Robotics Research*, 25(12): 1181–1203, 2006.

F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the Condensation algorithm for robust, vision-based mobile robot localization. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999a.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo Localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999b.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

R. Dillmann. Ka 1.10 benchmarks for robotics research. `http://www.cas.kth.se/euron/euron-deliverables/ka1-10-benchmarking.pdf`, 2004. Online, accessed 2015-08-18.

M. Đakulović, C. Sprunk, L. Spinello, I. Petrovic, and W. Burgard. Efficient navigation for anyshape holonomic mobile robots in dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the Association for Computing Machinery*, 40(5):1048–1066, 1993.

F. Duvallet and A. Tews. Wifi position estimation in industrial environments using Gaussian processes. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

P. Elinas and J. Little. σMCL: Monte-Carlo localization for mobile robots with stereo vision. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

F. Endres, C. Sprunk, R. Kümmerle, 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.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *Int. Journal of Computer Vision*, 88(2):303–338, 2010.

B. Ferris, D. Haehnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *Twelfth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, 2009.

M. Foskey, M. Garber, M. C. Lin, and D. Manocha. A voronoi-based hybrid motion planner. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

D. Fox. Adapting the sample size in particle filters through KLD-sampling. *Int. Journal of Robotics Research*, 22(12):985–1003, 2003.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal on Artificial Intelligence Reserach (JAIR)*, 11:391–427, 1999.

T. Fraichard and V. Delsart. Navigating dynamic environments with trajectory deformation. *Journal of Computing and Information Technology*, 17, 2009.

Frog AGV Systems. `http://frog.nl`, 2015. Online, accessed 2015-08-18.

T. Fujimoto, J. Ota, T. Arai, T. Ueyama, and T. Nishiyama. Semi-guided navigation of AGV through iterative learning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.

T. Goedemé, M. Nuttin, T. Tuytelaars, and L. V. Gool. Omnidirectional vision based topological navigation. *Int. Journal of Computer Vision*, 74(3):219–236, 2007.

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

H.-M. Gross, A. Koenig, H.-J. Boehme, and C. Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

H.-M. Gross, A. Köning, C. Schröter, and H.-J. Böhme. Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

G. Grunwald, G. Schreiber, A. Albu-Schaffer, and G. Hirzinger. Programming by touch: the different way of human-robot interaction. *IEEE Transactions on Industrial Electronics*, 50(4):659–666, 2003.

E. Guizzo. Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45(7): 26–34, 2008.

S. Gulati. *A Framework for Characterization and Planning of Safe, Comfortable, and Customizable Motion of Assistive Mobile Robots*. PhD thesis, The University of Texas at Austin, 2011.

S. Gulati and B. Kuipers. High performance control for graceful motion of an intelligent wheelchair. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. Technical report, University of British Columbia, 1992.

A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta. Navigation in three-dimensional cluttered environments for mobile manipulation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

J.-H. Hwang, R. C. Arkin, and D.-S. Kwon. Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

K. Hyyppa. Method of navigating an automated guided vehicle, 1989. US Patent 4,811,228.

D. Jourdan and N. Roy. Optimal sensor placement for agent localization. *ACM Trans. on Sensor Networks*, 4(3):1–40, 2008.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1): 47–64, 2004.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research*, 30(7):846–894, 2011.

S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

A. Kelly, B. Nagy, D. Stager, and R. Unnikrishnan. An infrastructure-free automated guided vehicle based on computer vision. *IEEE Robotics and Automation Magazine*, 2007.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5:90–98, 1986.

H. Kikkeri, G. Parent, M. Jalobeanu, and S. Birchfield. An inexpensive method for evaluating the localization performance of a mobile robot navigation system. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proc. of the Int. Conf. on Autonomous Agents*, 1997a.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997b.

R. Knotts, I. Nourbakhsh, and R. Morris. NaviGates: A benchmark for indoor navigation. In *Int. Conf. and Exp. on Robotics for Challenging Environments*, 1998.

O. Koch, M. R. Walter, A. S. Huang, and S. Teller. Ground robot navigation using uncalibrated cameras. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

J. Kolter and A. Ng. Task-space trajectories via cubic spline optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

Y. Koren. *The Global Manufacturing Revolution*. John Wiley & Sons, Inc., 2010.

P. Krüsi, B. Bücheler, F. Pomerleau, U. Schwesinger, R. Siegwart, and P. Furgale. Lighting-invariant adaptive route following using iterative closest point matching. *Journal of Field Robotics*, 2014.

M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Proc. of Robotics: Science and Systems (RSS)*, 2012.

M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard. Online generation of homotopically distinct navigation paths. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

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

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

R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 2014.

F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 20(6):967–977, 2004.

B. Lau, C. Sprunk, and W. Burgard. Improved updating of Euclidean distance maps and Voronoi diagrams. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

B. Lau, C. Sprunk, and W. Burgard. Incremental updates of configuration space representations for non-circular mobile robots with 2D, 2.5D, or 3D obstacle models. In *Proc. of the European Conf. on Mobile Robotics (ECMR)*, 2011.

B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10): 1116–1130, 2013.

S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *Int. Journal of Robotics Research*, 20(5), 2001.

C. Lee. A phase space spline smoother for fitting trajectories. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1):346–356, Feb. 2004.

S.-Y. Leea and H.-W. Yang. Navigation of automated guided vehicles using magnet spot guidance method. *Robotics and Computer-Integrated Manufacturing*, 2012.

J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. Journal of Robotics Research*, 28(8):933–945, 2009.

Y. Liu, J. J. Zhu, R. L. Williams II, and J. Wu. Omni-directional mobile robot controller based on trajectory linearization. *Robotics and Autonomous Systems*, 56(5), 2008.

T. Lozano-Perez. Robot programming. *Proc. of the IEEE*, 71(7):821–841, 1983.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

S. Macfarlane and E. Croft. Design of jerk bounded trajectories for online industrial robot applications. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2001.

M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007.

H. Mäkelä and T. von Numers. Development of a navigation and control system for an autonomous outdoor vehicle in a steel plant. *Control Engineering Practice*, 9(5): 573–583, 2001.

P. Malheiros, P. Costa, A. Moreira, and M. Ferreira. Robust and real-time teaching of industrial robots for mass customisation manufacturing using stereoscopic vision. In *Annual Conf. of IEEE Industrial Electronics (IECON)*, 2009.

C. Mandel and U. Frese. Comparison of wheelchair user interfaces for the paralysed: Head-joystick vs. verbal path selection from an offered route-set. In *Proc. of the European Conf. on Mobile Robotics (ECMR)*, 2007.

E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

J. Marshall, T. Barfoot, and J. Larsson. Autonomous underground tramming for center-articulated vehicles. *Journal of Field Robotics*, 25(6-7):400–421, 2008.

Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using viewsequenced route representation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1996.

K. Maček, G. Vasquez, T. Fraichard, and R. Siegwart. Towards safe vehicle navigation in dynamic urban scenarios. *Automatika*, 50(3-4):184–194, 2009.

M. Mazuran, C. Sprunk, W. Burgard, and G. D. Tipaldi. LexTOR: Lexicographic teach optimize and repeat based on user preferences. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot. Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287, 2013.

E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based Monte-Carlo localisation with omnidirectional images. *Robotics and Autonomous Systems*, 48(1): 17–30, 2004.

D. Meyer-Delius, M. Beinhofer, A. Kleiner, and W. Burgard. Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

J. Minguez and L. Montano. Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1), 2004.

D. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

P. Muir. *Modeling and Control of Wheeled Mobile Robots*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1988.

W. Niehsen. Information fusion based on fast covariance intersection filtering. In *Proc. of the Int. Conf. on Information Fusion*, 2002.

T. Niemueller, D. Ewert, S. Reuter, A. Ferrein, S. Jeschke, and G. Lakemeyer. Robocup logistics league sponsored by festo: A competitive factory automation testbed. In *Proc. of the RoboCup Symposium*, 2013.

N. J. Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

W. Nowak, A. Zakharov, S. Blumenthal, and E. Prassler. Benchmarks for mobile manipulation and robust obstacle avoidance and navigation. *BRICS Deliverable D3.1*, 2010.

C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot. Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014a.

C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier. Speed daemon: Experience-based mobile robot speed scheduling. In *Proc. of the Conf. on Computer and Robot Vision (CRV)*, 2014b.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

A. Piazzi, C. Guarino Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi. Quintic g^2-splines for the iterative steering of vision-based autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):27–36, 2002.

O. Purwin and R. D'Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54:13–22, 2006.

M. Quigley, D. Stavens, A. Coates, and S. Thrun. Sub-meter indoor localization in unmodified environments with inexpensive sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1993.

N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Int. Conf. on Neural Networks*, 1993.

R. Rojas and A. G. Förster. Holonomic control of a robot with an omnidirectional drive. *Künstliche Intelligenz*, 20(2):12–17, 2006.

J. Röwekämper, C. Sprunk, G. D. 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 Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *Int. Journal of Computer Vision*, 74(3):237–260, 2007.

M. Rufli, D. Ferguson, and R. Siegwart. Smooth path planning in constrained environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal. Normal distributions transform monte-carlo localization (NDT-MCL). In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson. Landmark selection for vision-based navigation. *IEEE Transactions on Robotics and Automation*, 22(2):334–349, 2006.

P. J. Schneider. Solving the nearest-point-on-curve problem. In A. S. Glassner, editor, *Graphics Gems*, pages 607–611. Academic Press, Inc., 1990.

J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Proc. of Robotics: Science and Systems (RSS)*, 2013.

S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

Z. Shiller and Y. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7, 1991.

K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Berlin Heidelberg, 2009.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

C. Sprunk, B. Lau, P. Pfaff, and W. Burgard. Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

C. Sprunk, B. Lau, and W. Burgard. Improved non-linear spline fitting for teaching trajectories to mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

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.

C. Sprunk, J. Röwekämper, G. Parent, L. Spinello, G. Tipaldi, W. Burgard, and M. Jalobeanu. An experimental protocol for benchmarking robotic indoor navigation. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2014.

C. Sprunk, B. Lau, and W. Burgard. The dynamicEDT3D package for updatable Euclidean distance transforms in 3D. Open source software released as part of the octomap package, `https://github.com/OctoMap/octomap`, 2015. Online, accessed 2015-08-18.

C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics and Automation Magazine*, 19(4):72–82, 2012. `http://ompl.kavrakilab.org`.

swisslog. `http://swisslog.com`, 2015. Online, accessed 2015-08-18.

A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto. Local path planning and motion control for agv in positioning. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems. The Autonomous Mobile Robots and Its Applications*, 1989.

S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1):99–141, 2001.

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

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

G. D. Tipaldi, L. Spinello, and W. Burgard. Geometrical flirt phrases for large scale place recognition in 2D range data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

G. D. Tipaldi, M. Braun, and K. O. Arras. FLIRT: Interest regions for 2D range data with applications to robot navigation. In O. Khatib, V. Kumar, and G. Sukhatme, editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 695–710. Springer Berlin Heidelberg, 2014.

N. Tomatis. Bluebotics: Navigation for the clever robot [Entrepreneur]. *IEEE Robotics and Automation Magazine*, 18(2):14–16, 2011.

A. Ude. Trajectory generation from noisy positions of object features for teaching robot paths. *Robotics and Autonomous Systems*, 11(2):113–127, 1993.

R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal. Localization in highly dynamic environments using dual-timescale NDT-MCL. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

M. Vitus and C. Tomlin. Sensor placement for improved robotic navigation. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.

W. Wang, H. Pottmann, and Y. Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (TOG)*, 25:214–238, 2006.

K. Watanabe. Control of an omnidirectional mobile robot. In *Proc. of Int. Conf. on Knowledge-Based Intelligent Electronic Systems*, 1998.

D. J. Webb and J. van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

M. Werling and L. Gröll. Low-level controllers realizing high-level decisions in an autonomous vehicle. In *IEEE Intelligent Vehicles Symposium (IV)*, 2008.

T. Wisspeintner, T. Van Der Zant, L. Iocchi, and S. Schiffer. Robocup@ home: Scientific competition and benchmarking for domestic service robots. *Interaction Studies*, 10(3): 392–426, 2009.

P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19, 2008.

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

M. Zaeh and M. Prasch. Systematic workplace and assembly redesign for aging workforces. *Production Engineering*, 1(1):57–64, 2007.

J. Ziegler and C. Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

J. Ziegler, M. Werling, and J. Schröder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *IEEE Intelligent Vehicles Symposium (IV)*, 2008.