Autonomous Navigation for Miniature Indoor Airships

Jörg Müller

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Autonomous Navigation for Miniature Indoor Airships

Jörg Müller

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

Dekan	Prof. Dr. Yiannos Manoli
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Maren Bennewitz
	Albert-Ludwigs-Universität Freiburg
Tag der Disputation	05.06.2013

Zusammenfassung

Eingebettete Systeme sind heutzutage ein wesentlicher Bestandteil unserer Technik in Industrie und Alltag. Zum Beispiel Geräte der Unterhaltungselektronik, moderne Technik zur Automatisierung von Gebäuden, Fahrerassistenzsysteme im Automobilbereich und natürlich Roboter sind üblicherweise mit Sensoren und eingebetteten Systemen ausgestattet, um intelligent mit Menschen oder ihrer Umgebung zu interagieren oder Situationen zu erkennen und entsprechend zu reagieren. Im Rahmen dieser Anforderungen entstehen jedoch einige Herausforderungen. Viele eingebettete Systeme werden in mobilen Anwendungen genutzt und werden daher möglichst miniaturisiert entwickelt, um sie möglichst flexibel einsetzen zu können. Außerdem besteht in den meisten Anwendungsbereichen ein enormer Kostendruck, der die verwendete Hardware stark einschränkt. Folglich müssen eingebettete Systeme energieeffizient sein, haben oft nur begrenzte Rechenleistung und sind in der Regel nur mit kostengünstigen und schwachen Aktuatoren sowie mit kleinen und meist entsprechend unpräzisen Sensoren ausgestattet.

In der Robotik als großem Anwendungsbereich von eingebetteten Systemen gilt die Fähigkeit autonom zu navigieren als eine der Grundvoraussetzungen, um den Menschen in Industrie und Alltag flexibel unterstützen zu können. Zum Beispiel müssen Staubsauger- und Transportroboter in der Lage sein, in ihrer Umgebung systematisch und zuverlässig zu navigieren. Eine solche Navigation erfordert eine genaue Lokalisierung, auf deren Basis der Roboter in der Planung geeignete Aktionen auswählen und somit zu seinem Ziel navigieren und seine Aufgabe erfüllen kann.

Bei der Lokalisierung bestimmt der Roboter die eigene Position und Orientierung in einer gegebenen Umgebungskarte. Dabei macht er üblicherweise eine Vorhersage seiner Bewegung auf Basis der ausgeführten Kontrollkommandos und nimmt seine Umgebung über einen oder mehrere Sensoren wahr. Da Sensoren unterschiedlicher Art zum Einsatz kommen können, müssen die gegebenenfalls heterogenen Daten fusioniert werden, um eine genaue Zustandsschätzung zu ermöglichen. Für die Lokalisierung stellen die schwachen Aktuatoren und ungenauen Sensoren, die in kleinen, kostengünstigen Systemen üblicherweise verwendet werden, eine große Herausforderung dar. Um eine genaue und zuverlässige Lokalisierung zu ermöglichen, muss die größtmögliche Menge an Information aus den vorhandenen fehlerbehafteten Daten extrahiert werden. Eine hierfür geeignete Technik ist die probabilistische Sensordatenfusion, die in der Regel jedoch eine hohe Rechenleistung erfordert, da sie die gesamte Wahrscheinlichkeitsverteilung über

dem Zustandsraum des Systems schätzt. Obwohl es hierfür effiziente Approximationen wie zum Beispiel den Kalman Filter gibt, basieren diese jedoch auf Annahmen, die nicht immer erfüllt sind.

Bei der Planung berechnet der Roboter einen Pfad von seiner in der Lokalisierung geschätzten Pose zu dem vorgegebenen Ziel. Insbesondere wenn hierbei schnelle Bewegungen erforderlich sind oder kleine, kostengünstige Systeme mit schwachen Aktuatoren nahe an die Grenzen ihrer Kraft kommen und damit nicht in jeder Situation jede Aktion ausführen können, muss in der Planung die Kinematik und Dynamik des Systems berücksichtigt werden. In diesem sogenannten kinodynamischen Planungsproblem muss der betrachtete Zustandsraum um die Geschwindigkeiten des Systems erweitert werden. Folglich ist die häufig angewandte Dimensionsreduktion durch die Entkopplung der Planung der Trajektorienform von der Planung des Geschwindigkeitsprofils im Allgemeinen unmöglich.

In dieser Arbeit zeigen wir am Beispiel der autonomen Navigation für Miniaturluftschiffe, wie man mit den Herausforderungen von kleinen, kostengünstigen eingebetteten Systemen, nämlich begrenzter Rechenleistung, schwachen Aktuatoren und ungenauen Sensoren, umgehen kann. Luftschiffe sind eine beliebte Plattform in der Robotik, da sie relativ leise und gefahrlos navigieren können und durch ihren geringen Energieverbrauch auch für Langzeitanwendungen in Frage kommen. Damit decken sie weite Einsatzbereiche ab, zum Beispiel Umweltbeobachtung, Überwachung, Katastropheneinsätze, Kommunikation und Werbung, und können auch ohne Risiko in der Öffentlichkeit in der Nähe von Personen genutzt werden.

Luftschiffe bringen jedoch einige Herausforderungen für die autonome Navigation mit sich. Luftschiffe für den Innenbereich müssen klein sein und daher ist ihre Tragkraft entsprechend ihres Volumens eingeschränkt. Folglich sind die Kapazität der Batterien, die verfügbare Rechenleistung, die Wahrnehmungsfähigkeiten der eingesetzten Sensoren und die Leistung der Rotoren in der Regel stark begrenzt. Insbesondere stehen meist nur ungenaue und oft nicht eindeutige Messdaten einer geringen Anzahl leichter und kleiner Sensoren zur Verfügung. Außerdem dominiert der Luftwiderstand die Dynamik von Luftschiffen, so dass deren Bewegung nicht leicht vorhergesagt werden kann. Des Weiteren sind Luftschiffe meistens träge und untermotorisiert, so dass kinodynamische Bewegungsplanung für ein nichtlineares System in einem hochdimesionalen Zustandsraum erforderlich ist.

In dieser Arbeit stellen wir das Design und die Realisierung eines Miniaturluftschiffs für die autonome Navigation vor. Unser Luftschiff ist mit verschiedenen miniaturisierten Sensoren, nämlich Sonarsensoren, Luftstromsensoren und einer inertialen Messeinheit (IMU), ausgestattet. Im Gegensatz zu Kameras liefern diese Sensoren niederdimensionale Messwerte, die keine komplexe Merkmalsextraktion erfordern und damit auch auf Prozessoren mit geringer Rechenleistung verarbeitet werden können. Unser

Luftschiff ist in Leichtbauweise konstruiert, kann durch sein modulares Konzept flexibel an die Anwendung angepasst werden und ist ein hervorragendes Beispiel für kostengünstiges, leichtes und energiesparendes Design in der Robotik.

Diese Arbeit leistet einen Beitrag im Bereich der autonomen Navigation für mobile Roboter im Kontext von ressourcenbeschränkten eingebetteten Systemen, indem wir genaue und effiziente Techniken für die Lokalisierung und Planung entwickeln. Unsere Ansätze zur probabilistischen Zustandsschätzung im Partikelfilter maximieren die aus fehlerbehafteten Sensordaten und aus ungenau ausgeführten Kontrollsignalen extrahierte Information und liefern effiziente Approximationen, die eine Online-Zustandsschätzung und -Planung bei einer hohen Approximationsgüte ermöglichen.

Wir stellen ein neues probabilistisches Sensormodell für Sonarsensoren vor. Im Gegensatz zu anderen Modellen berücksichtigt unser Ansatz explizit die Eigenschaften von sehr kleinen Sonarsensoren mit großem Öffnungswinkel. Unser Modell basiert dabei auf dem physikalischen Messprozess und berücksichtigt die Mehrdeutigkeit von Messungen, die durch die Reflexion des Ultraschallsignals an Objekten verschiedener Größe in unterschiedlichen Distanzen entsteht. Wir entwickeln ein neues probabilistisches Sensormodell für Luftstromsensoren, das sowohl für die Berechnung der Odometrie als auch für die probabilistische Zustandsschätzung geeignet ist. Im Gegensatz zu anderen Ansätzen modellieren wir die Unsicherheit des Messprozesses von Luftstromsensoren explizit und berücksichtigen die heterogene Verteilung des Messrauschens. Außerdem stellen wir ein effektives probabilistisches Modell zur Integration von Orientierungsschätzungen einer IMU in die probabilistische Zustandsschätzung vor. Im Vergleich zu einem Standardmodell ermöglicht unser Modell eine signifikant genauere Lokalisierung.

Je weniger Sensordaten zur Lokalisierung zur Verfügung stehen, desto wichtiger ist die genaue Vorhersage der Bewegung im Bewegungsmodell. Daher stellen wir einen Ansatz vor, der eine höhere Lokalisierungsgenauigkeit ermöglicht, indem anfänglich unbekannte oder variable Parameter des Bewegungsmodells gleichzeitig mit der Pose geschätzt werden. Unser Verfahren verhindert dabei Oszillationen der Parameterschätzung auch dann, wenn die Änderung eines oder mehrerer Parameter verzögert erkannt wird.

Für die autonome Navigation ist bei der Lokalisierung besonders wichtig, dass die Pose des Roboters online, also mindestens so schnell wie die Mess- und Kontrolldaten eingehen, berechnet werden kann. Hierfür entwickeln wir ein effizientes odometriebasiertes Bewegungsmodell für fliegende Roboter. Unser neues Bewegungsmodell berechnet die Odometrie linear aus den Messdaten von Luftstromsensoren und der IMU und erfordert daher wesentlich weniger Rechenleistung als das Standardbewegungsmodell, das auf physikalischer Simulation der Bewegung des Luftschiffs basiert. Des Weiteren stellen wir einen Ansatz vor, der die zeitliche Korrelation von Messfehlern in der Vor-

hersage des Partikelfilters berücksichtigt. Unser odometriebasiertes Bewegungsmodell verringert die Dimensionalität des Zustandsraums im Partikelfilter und ermöglicht daher eine genaue und effiziente Online-Lokalisierung für Miniaturluftschiffe.

Neben der Lokalisierung sind die Planung und die Regelung wichtige Bestandteile der autonomen Navigation mobiler Roboter. Für Miniaturluftschiffe ist dies eine besondere Herausforderung, da deren komplexe und nichtlineare Dynamik in Verbindung mit den schwachen Aktuatoren die Lösung des kinodynamischen Planungsproblems in einem hochdimensionalen Zustandsraum erfordert. In dieser Arbeit stellen wir einen mehrstufigen Algorithmus zur Bewegungsplanung und die Implementierung eines linear-quadratischen Reglers (LQR) vor und lösen damit das hochdimensionale kinodynamische Planungsproblem auf effiziente Weise. Unser Planungsalgorithmus basiert auf einer pfadgestützten, zielgerichteten Wahl von randomisierten Teilzielen und berechnet schnell kinematisch korrekte Teiltrajektorien, die in den folgenden Planungszyklen erweitert werden.

Alle in dieser Arbeit vorgestellten Ansätze wurden implementiert, gründlich getestet und umfassend evaluiert. Die Ergebnisse der Experimente zeigen, dass unser Lokalisierungsverfahren in der Lage ist, Miniaturluftschiffe online und genau in komplexen Innenräumen zu lokalisieren, und dass unser System zur Bewegungsplanung und Regelung die effiziente und zuverlässige Steuerung von Miniaturluftschiffen in realistischen Szenarien ermöglicht. Des Weiteren zeigen wir, dass die in dieser Arbeit vorgestellten Techniken besser als vergleichbare aktuelle Methoden zur autonomen Navigation sind.

Zusammenfassend stellen wir in dieser Arbeit Ansätze vor, die es einem autonom agierenden mobilen Roboter erlauben,

- die ungenauen und mehrdeutigen Messungen kleiner Sensoren zur Lokalisierung zu interpretieren und die aus den Messungen extrahierte Information zu maximieren,
- bei der Zustandsschätzung mit sich ändernden Parametern der Bewegungsdynamik umzugehen,
- das hochdimensionale Problem der Zustandsschätzung effizient zu approximieren und dabei eine robuste und genaue Online-Lokalisierung zu ermöglichen und
- trotz hoher Trägheit und schwacher Aktuatoren schnell und effektiv geeignete Aktionen zu einem Ziel zu planen.

Wir demonstrieren, dass autonome Roboter in ihrer Umgebung durch die in dieser Arbeit vorgestellten Methoden zuverlässig und effektiv navigieren können und wir sind überzeugt, dass unsere Lösungen relevant für zukünftige kleine, kostengünstige und ressourcenbeschränkte Systeme in Industrie und Alltag sind.

Abstract

In recent years, embedded systems have become popular, and, for example, consumer electronics, transportation, and robotics are hard to imagine without them. As mobile devices with embedded systems are often supposed to act intelligently, they are usually equipped with sensors and actuators for interaction with humans or for autonomous operation. However, cost pressure and miniaturization impose several challenges on embedded devices. They have to be designed in an energy-efficient way and therefore have limited computational power, which is often paired with weak actuators and limited sensing capabilities.

In this thesis, we show how to cope with these challenges using the example of autonomous navigation for miniature indoor airships. Such airships have become popular, because they can navigate safely through three-dimensional environments and operate in long-term navigation tasks. We present the design and the implementation of a miniature indoor blimp and several novel techniques for effective autonomous navigation.

In particular, we introduce techniques for robust online localization of miniature airships in known, complex indoor environments. We present a particle filter implementation for probabilistic state estimation of airships equipped with lightweight sonar and air flow sensors as well as an IMU. We introduce probabilistic models dedicated to the miniature and lightweight sensors applied on our blimp. In contrast to other approaches, our models explicitly consider the uncertainty of the measurement process and therefore specify appropriate measurement likelihood functions that enable a robust localization of the blimp. Furthermore, we show that the simultaneous estimation of motion model parameters is beneficial to the localization accuracy and enables to adapt to changing parameters of the system dynamics during operation. We achieve a robust and efficient online localization by introducing an efficient probabilistic odometry motion model based on the measurements of air flow sensors and an IMU. Particularly, our linear odometry model is substantially less computationally demanding than the standard physical simulation-based motion model and decreases the dimensionality of the state space in the particle filter.

In addition to our solutions to online self-localization, we introduce an effective approach to planning and closed-loop control for autonomous navigation. Our method efficiently solves the high-dimensional, kinodynamic planning and control problem, which is imposed by airships with weak actuators, through a multi-stage planner and an LQR

controller. In contrast to other approaches, our planning algorithm performs path-guided sampling and selects optimal actions towards subgoals and therefore can quickly provide a partial trajectory, which is extended during operation.

We implemented and thoroughly tested our novel methods in extensive experiments in simulation and with real robots. We validated the properties of our algorithms and demonstrated the advantages of our approaches compared to state-of-the-art methods. While the work as a whole aims at the autonomous navigation for miniature airships, the individual algorithms and techniques presented in this thesis are often applicable to a variety of problems. Therefore, we believe that the proposed methods are relevant for future low-cost, small, and resource-constrained embedded systems with applications in industrial settings and everyday life.

Acknowledgment

This thesis would not have been possible without the support of several people and I would like to thank everyone who contributed to this work.

First of all, I would like to thank Wolfram Burgard. He was a great advisor and provided me an exceptional degree of freedom in developing my own ideas while at the same time having an open door for discussions. He encouraged me and guided me in critically dealing with theory and practice of research as well as in scientific writing. He provided excellent opportunities for academic cooperations and for working with state-of-the-art equipment in practical experiments.

I would also like to thank Maren Bennewitz for helpful comments and for reviewing this thesis.

Many thanks to Axel Rottmann for providing his experience and assistance with the robotic blimp platform. Many thanks also to Matthias Sippel and Fabian Höflinger for their cooperation in building and adapting hardware modules and Thorsten Zitterell for his support on embedded operating systems. Furthermore, I would like to thank my co-authors for fruitful discussions and collaborations. In particular, I would like to thank Maximilian Beinhofer, Fabian Höflinger, Matthias Sauer, Andreas Riefert, Matthew Lewis, Ilia Polian, Victor Tomashevich, Markus Kuderer, Johannes Wendeberg, Oliver Paul, Bernd Becker, Christian Schindelhauer, and Leonhard Reindl.

My special thanks for being good friends and great colleagues and for their courteous assistance in many theoretical and practical questions go to Kai Wurm, Christoph Sprunk, and Maximilian Beinhofer. Many thanks for his contributions to our software framework Robular go to Boris Lau. I thank Oliver Paul, Jan Peters, and Cyrill Stachniss for fruitful discussions and valuable suggestions. Furthermore, I thank Sensirion AG for providing flow sensor chips and Josef Joos for his help in adapting these flow sensors. Many thanks to the students and research assistants I co-supervised in Freiburg. Especially Christoph Gonsior, Norman Kohler, Andreas Spilla, Johannes Meyer, and Florian Sittel valuably contributed to our research projects.

I thank Christoph Sprunk, Maximilian Beinhofer, Henrik Kretzschmar, Markus Kuderer, Rainer Kümmerle, Nichola Abdo, Bastian Steder, Axel Rottmann, Stefan Oßwald, Barbara Frank, Michael Ruhnke, and Jürgen Hess for reviewing and proof-reading earlier versions of this document. For their help in technical and administrative matters, my thanks go to Michael Keser, Günter Martin, Manuela Kniß, and Susanne Bourjaillat.

Furthermore, I would like to thank the University of Freiburg and the members of the Autonomous Intelligent Systems lab and the PhD program “Embedded Microsystems” for providing me a professional, interdisciplinary, and enjoyable work environment.

Finally, my deepest gratitude goes to my wife Sabrina and my family for their unconditional support in every period of my life.

This work has been partly supported by the German Research Foundation (DFG) within the Research Training Group 1103 and by the Gottfried Wilhelm Leibniz Program of the DFG.

Contents

Chapter 1

Introduction

Nowadays, embedded systems are omnipresent in everyday life. For example, consumer electronics such as video game consoles or digital cameras, home automation devices, driver assistance systems in modern cars, and of course robots are equipped with sensors and embedded computers. Many of these devices are supposed to act intelligently by perceiving their environment and dynamically reacting to situations or interacting with humans. However, there are several challenges with respect to these aims. Many embedded systems are used in mobile applications and therefore are designed in a miniaturized way. Furthermore, usual applications and markets induce a certain cost pressure on such systems. As a consequence, most embedded devices need to be energy-efficient and are restricted with respect to computational power. In addition, their actuators are often weak and imperfect and they typically have limited sensing capabilities so that they have to rely on imprecise measurements.

In the context of mobile robots, the ability to navigate autonomously is commonly regarded as a core prerequisite to flexibly provide a wide range of services. For example, vacuum cleaning robots and autonomous industrial transportation systems are required to navigate systematically and reliably in their environment. For effective autonomous navigation, a mobile robot must be able to localize itself and to plan suitable actions to reach a predefined goal and to accomplish its task.

During self-localization, the robot estimates its pose in a given map of the environment. In this process, the robot usually predicts its motion given the applied control commands and perceives its environment through one or multiple sensors. The sensors can be of various types and their data has to be fused to obtain accurate state estimates during operation. Especially in the context of small and low-cost systems, imperfect actuators as well as sparse and imprecise sensor data often impose a challenging problem. To enable an accurate localization, one has to extract the maximum amount of information from the available noisy data. Probabilistic sensor data fusion and state estimation techniques, e.g. particle filters, have proven to be suitable means for such problems. However, they are computationally demanding as they require to represent the full probability density function of the state of the system. Although there are efficient approxi-

mations like the popular Kalman filter [79], these approximations are usually restricted to a subset of problems, for example to linear, Gaussian systems.

In the planning task, the robot determines a path from the estimated pose to a predefined goal. In many robotic tasks, fast movements are required or small and low-cost systems with weak actuators need to operate close to their velocity and acceleration limits. Then, the kinematics and dynamics of the system have to be taken into account during planning, as the possible actions are restricted in many situations. This requires to solve the kinodynamic motion planning problem in the high-dimensional state space including the velocities. Furthermore, the commonly applied reduction of dimensionality through the decoupled planning of the trajectory shape and the velocities is not applicable.

In this thesis, we show how to cope with the challenges imposed by small and low-cost embedded systems, namely limited computational power, imperfect actuators, and imprecise sensors, using the example of autonomous navigation for miniature indoor airships. Indoor airships have become popular in the robotics community because of their substantial advantages. They can navigate safely and with low noise in three-dimensional environments and their low power consumption makes them well-suited for long-term operation tasks. These features facilitate a wide range of applications including environmental monitoring, surveillance, disaster scenarios, communication, and advertising even in the presence of people, e.g. in public spaces.

However, these favorable properties of indoor airships come at the cost of some challenges with respect to autonomous navigation. Since the payload of small airships is restricted according to their volume, the available battery power, the computational resources, the sensing capabilities of the employed sensors, and also the rotor power are limited. Especially the small number of lightweight and tiny sensors, which can be carried by miniature airships, typically provide only imprecise and often ambiguous measurements. Additionally, the dynamics of airships are dominated by their air drag, which is hard to predict efficiently during operation. Furthermore, the limited acceleration capabilities together with the serious under-actuation require kinodynamic motion planning in a high-dimensional state space.

In this thesis, we present the design of a miniature robotic blimp platform for autonomous indoor navigation. As opposed to zeppelins, which have a rigid framework supporting the hull, blimps are the kind of airships that keep the form of the hull through an overpressure of the lifting gas. Our blimp is equipped with tiny sonar sensors and microelectromechanical system (MEMS)-based air flow sensors and an inertial measurement unit (IMU). In contrast to cameras, these sensors provide a low-dimensional measurement output, which does not require a complex feature extraction and can be processed on the resource-limited CPU of a blimp. Our weight-saving design can be flexibly adapted to the requirements of the navigation task and is an excellent example

for lightweight, low-cost, and power-saving concepts in robotics.

As a contribution to the general field of autonomous navigation for resource-constrained embedded systems, we present techniques to probabilistically fuse measurements obtained from noisy sensor data and controls executed by imprecise actuators. Our corresponding models maximize the amount of information extracted from the available data for robust probabilistic state estimation in a particle filter framework. Furthermore, we introduce efficient approximations that enable a mobile robot to solve high-dimensional state estimation and planning problems online during autonomous navigation. While the individual algorithms and techniques included in this thesis are applicable to a variety of problems, the work as a whole aims at autonomous navigation for miniature indoor airships.

1.1 Key Contributions

With this work, we contribute to the field of robotics research by developing solutions to autonomous navigation in the context of small, low-cost, and resource-constrained systems. We address the core tasks in autonomous navigation, namely state estimation, motion planning, and closed-loop control for mobile robots. Using the example of a robotic blimp, this thesis shows the following contributions:

- We present the design and the implementation of a flexible, low-cost, small, and weight-saving robotic blimp platform (Chapter 2). Our hardware and software architecture is designed in a modular way so that it can be easily extended and adapted to various navigation tasks.

- We introduce probabilistic sensor models for tiny sonar sensors (Chapter 4), miniature air flow sensors (Chapter 5), and IMUs (Chapter 6). In contrast to other approaches, our sensor models are developed by systematically considering the physical measurement process and taking into account the measurement uncertainty of tiny, imprecise sensors. This enables a robot to robustly and accurately localize itself for reliable autonomous navigation.

- We propose the general formulation of probabilistic localization with simultaneous estimation of motion model parameters (Chapter 7). Our approach enables to estimate initially unknown or changing parameters of the system dynamics during localization. This enables a robot to accurately localize itself even if only sparse and imprecise sensor information is available. Furthermore, this approach adaptively copes with changing parameters of the motion model.

- We introduce a novel odometry motion model for accurate online localization of flying vehicles (Chapter 8). Our odometry motion model is based on the mea-

surements of air flow sensors and an IMU and is less computationally demanding compared to the standard physical simulation-based motion model. We explicitly model and propagate the uncertainty in the measurements and introduce a general approach to take into account the temporal correlations of odometry measurement errors for efficient and accurate online localization.

- We present an approach to online motion planning and closed-loop control for miniature airships (Chapter 10). Our multi-stage planning algorithm effectively solves the kinodynamic motion planning problem. It performs path-guided tree planning in the high-dimensional state space and thus can quickly provide a partial trajectory, which is extended and refined in the consecutive planning steps during operation. Furthermore, we present the implementation of a linear quadratic regulator (LQR) for efficient closed-loop control in the high-dimensional state space.

Overall, we introduce probabilistic approaches to accurate and robust self-localization as well as an approach to kinodynamic motion planning for miniature airships. To cope with these high-dimensional problems, we propose efficient approximations that have only small approximation errors and that enable online state estimation and planning in autonomous navigation. We trained our probabilistic models from collected data of a real robotic blimp and validated the theoretical results presented throughout this thesis in numerous extensive experiments in simulation and with real robots.

1.2 Publications

Parts of the thesis have been published in international journals, conference proceedings, and technical reports, and have been presented at conferences and workshops.

- J. Müller, A. Rottmann, L.M. Reindl, and W. Burgard. A probabilistic sonar sensor model for robust localization of a small-size blimp in indoor environments using a particle filter. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

- J. Müller, C. Gonsior, and W. Burgard. Improved Monte Carlo localization of autonomous robots through simultaneous estimation of motion model parameters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

- J. Müller, N. Kohler, and W. Burgard. Autonomous miniature blimp navigation with online motion planning and re-planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

- J. Müller, O. Paul, and W. Burgard. Probabilistic velocity estimation for autonomous miniature airships using thermal air flow sensors. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

- J. Müller and W. Burgard. Efficient probabilistic localization for autonomous indoor airships using sonar, air flow, and IMU sensors. *Advanced Robotics*, 27(9): 711–724, 2013. doi:10.1080/01691864.2013.779005.

- F. Sittel, J. Müller, and W. Burgard. Computing velocities and accelerations from a pose time sequence in three-dimensional space. Technical Report 272, University of Freiburg, Department of Computer Science, 2013.

Parts of Chapter 4 have been published in the diploma thesis

- J. Müller. Techniken für die Navigation autonomer Luftschiffe. Master's thesis, University of Freiburg, Department of Computer Science, 2008. In German.

This thesis does not report on the following publications, which were written during the time as a stipend of the PhD program 1103 "Embedded Microsystems" and as a research assistant. They are given in chronological order, grouped by their main subject.

- J. Müller, C. Stachniss, K.O. Arras, and W. Burgard. Socially inspired motion planning for mobile robots in populated environments. In *Proc. of the Int. Conf. on Cognitive Systems (CogSys)*, 2008.

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

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

- M. Beinhofer, J. Müller, and W. Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics & Autonomous Systems*, 2013. doi:10.1016/j.robot.2012.08.009.

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

- A. Spilla, I. Polian, J. Müller, M. Lewis, V. Tomashevich, B. Becker, and W. Burgard. Run-time soft error injection and testing of a microprocessor using FPGAs. In *Proc. of the Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TUZ)*, 2011.

- M. Sauer, V. Tomashevich, J. Müller, M. Lewis, A. Spilla, I. Polian, B. Becker, and W. Burgard. An FPGA-based framework for run-time injection and analysis of soft errors in microprocessors. In *Proc. of the IEEE Int. On-Line Testing Symposium (IOLTS)*, 2011.

- A. Riefert, J. Müller, M. Sauer, W. Burgard, and B. Becker. Identification of critical variables using an FPGA-based fault injection framework. In *Proc. of the IEEE VLSI Test Symposium (VTS)*, 2013.

- A. Riefert, J. Müller, M. Sauer, W. Burgard, and B. Becker. Identification of critical variables using an FPGA-based fault injection framework. In *Proc. of the Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TUZ)*, 2013.

- F. Höflinger, J. Müller, M. Törk, L.M. Reindl, and W. Burgard. A wireless micro inertial measurement unit (IMU). In *Proc. of the IEEE Int. Instrumentation and Measurement Technology Conf. (I2MTC)*, 2012.

- F. Höflinger, J. Müller, R. Zhang, W. Burgard, and L.M. Reindl. A wireless micro inertial measurement unit (IMU). *IEEE Transactions on Instrumentation & Measurement*, 2013. Accepted for publication.

- J. Wendeberg, J. Müller, C. Schindelhauer, and W. Burgard. Robust tracking of a mobile beacon using time differences of arrival with simultaneous calibration of receiver positions. In *Proc. of the Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*, 2012.

- J. Meyer, M. Kuderer, J. Müller, and W. Burgard. Online marker labeling for automatic skeleton tracking in optical motion capture. In *Proc. of the ICRA Workshop on Computational Techniques in Natural Motion Analysis and Reconstruction*, 2013.

1.3 Collaborations

Parts of this thesis haven been developed in collaboration with others. During the design and the implementation of the robotic blimp platform (Chapter 2), I closely cooperated with Axel Rottmann, Matthias Sippel, and Thorsten Zitterell, who created the basic version of our first prototype of the robotic blimp. Furthermore, the modular software framework for robotics (Section 2.2.3) was designed and implemented together with Boris Lau. The calculation of reference velocities and accelerations (Section 2.5.2) was developed in collaboration with Florian Sittel during his work as a student research assistant under my supervision. The probabilistic sonar sensor model described in Chapter 4

was jointly developed with Axel Rottmann and benefited from comments of Leonhard Reindl. The development of the air flow sensor model introduced in Chapter 5 had a substantial benefit from fruitful discussions with Oliver Paul.

During my time as a PhD student, I supervised several master and bachelor projects. The localization with simultaneous parameter estimation was originally focused in the bachelor thesis of Christoph Gonsior [56] and parts of the insight gained during that project influenced Chapter 7. Furthermore, the planning for miniature airships was first addressed by Norman Kohler in his diploma thesis [89]. In contrast to Chapter 10, this diploma project only addressed the offline planning problem for simulated blimps.

1.4 Notation

The following table summarizes the notation used throughout this work.

Notation	Meaning
$a, b, \ldots$	Scalar values
$\mathbf{a}, \mathbf{b}, \ldots$	(Column) vectors
$A, B, \ldots$	Matrices
$\mathbf{a}^T, A^T$	The transpose of a vector and a matrix
$\lvert a \rvert$	The absolute value of a scalar
$\lVert \mathbf{a} \rVert$	The L^2 norm of a vector, also called Euclidean length
$[\ldots]$	A (row) vector composed of the given values
$\{\ldots\}$	A set of the given values
$(\ldots)$	A tuple of the given values
$\mathbf{x}_{1:t}$	A sequence of the values $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$
$\{(x_i, y_i)\}_{i \in [1,n]}$	A set of tuples, here $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
$p(a)$	The probability density function of the specific value a
$p(a \mid b)$	The conditional probability density function of a given b
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$	The (multivariate) normal distribution (also called Gaussian) of the random variable $\mathbf{x}$ with the mean $\boldsymbol{\mu}$ and the covariance Σ
$\mathrm{diag}(\mathbf{a})$	The diagonal matrix with the vector $\mathbf{a}$ on the main diagonal
$\mathrm{Cov}(\mathbf{a}_1, \mathbf{a}_2, \ldots)$	The covariance of a random variable estimated from the given samples $\mathbf{a}_1, \mathbf{a}_2, \ldots$
$\mathbf{q}_1, \mathbf{q}_2, \ldots$	Unit quaternions according to Diebel [37]
$\mathbf{q}^{-1}$	The inverse of the unit quaternion $\mathbf{q}$, which is equal to its adjoint $\bar{\mathbf{q}}$ [37]
$\mathbf{q}_1 \odot \mathbf{q}_2$	The quaternion product of $\mathbf{q}_1$ and $\mathbf{q}_2$ [37]
$\tilde{\mathbf{q}}(\boldsymbol{\varphi})$	The quaternion corresponding to the incremental three-dimensional rotation $\boldsymbol{\varphi} = [\varphi_x, \varphi_y, \varphi_z]^T$ around all three axes [37]
$I_{3\times3}$	The 3×3 identity matrix

The following table summarizes the important terms and symbols used throughout this work.

Symbol	Meaning
$\mathbf{x}$	The state vector of the robot
$\mathbf{z}$	The measurement vector of the robot
$\mathbf{u}$	The control vector of the robot
$\tilde{\mathbf{u}}$	The odometry vector of the robot
$\mathcal{F}_{\mathrm{g}}$	The global frame of reference
$\mathcal{F}_{\mathrm{b}}$	The body-fixed frame of reference
$\mathbf{p}, \mathbf{q}$	The position and orientation of the robot (with respect to $\mathcal{F}_{\mathrm{g}}$)
$\mathbf{v}, \boldsymbol{\omega}$	The translational and rotational velocity of the robot (with respect to $\mathcal{F}_{\mathrm{b}}$)
$\mathbf{a}, \boldsymbol{\alpha}$	The translational and rotational acceleration of the robot (with respect to $\mathcal{F}_{\mathrm{b}}$)
$\mathbf{x}^{\star}$	The ground truth state of the robot
$\mathbf{x}^{\blacktriangledown}, \mathbf{u}^{\blacktriangledown}$	The desired state and control of the robot
t	The (current) time step
$\mathcal{M}$	The set of particles in the particle filter (see Section 3.2)
N	The number of particles in the particle filter
L	The number of air flow sensors our blimp is equipped with
K	The number of sonar sensors our blimp is equipped with
g	The gravity of Earth ($g \approx 9.81\ \frac{\mathrm{m}}{\mathrm{s}^2}$)

The following table summarizes the acronyms used throughout this work.

Acronym	Meaning
IMU	Inertial measurement unit
MEMS	Microelectromechanical system
EKF	Extended Kalman filter
UKF	Unscented Kalman filter
SIR	Sampling importance resampling
IPC	Inter process communication
UAV	Unmanned aerial vehicle
RRT	Rapidly-exploring random tree
PRM	Probabilistic road map
LQR	Linear quadratic regulator

1.5 Outline

This thesis is organized as follows. In Chapter 2, we present the design and the implementation of our robotic blimp platform, which is used throughout the experiments presented in this thesis, and introduce the state and environment models as well as a physical motion model for miniature airships.

In Part I, which includes Chapter 3 to 8 of this thesis, we present our innovative approaches to probabilistic online localization for miniature airships. Chapter 3 summarizes the recursive state estimation technique for online self-localization of mobile robots. We give a description of the particle filter and provide implementation details in the context of miniature airship localization. The particle filter localization relies on accurate probabilistic models of the motion and of the sensors of the robot. Consequently, we introduce novel sensor models for the small sensors our blimp is equipped with in Chapter 4 to 6. In particular, we present our novel sensor model for tiny sonar sensors with a wide opening angle in Chapter 4, we introduce our sensor model for air flow sensors in Chapter 5, and we propose a sensor model for the orientation estimates of an IMU in Chapter 6. For a robust localization in the particle filter, even if only sparse sensor information is available, we present an approach to simultaneous localization and estimation of the parameters of the motion model in Chapter 7 and develop a technique to adapt to changing parameters during operation. In Chapter 8, we introduce an efficient odometry motion model based on air flow and IMU measurements that enables a robust and accurate online localization for miniature airships.

In Part II, which includes Chapter 9 and 10 of this thesis, we present our effective approach to online motion planning and control. Chapter 9 gives a summary of state-of-the-art motion planning and control algorithms on which our autonomous navigation system is based. Subsequently, in Chapter 10, we introduce our efficient, path-guided multi-stage planning approach and present our implementation of the LQR closed-loop controller for online motion planning and autonomous navigation of miniature airships.

Finally, we recapitulate the contribution of this thesis in Chapter 11 and discuss possible future research directions in the field of autonomous navigation for miniature airships and embedded systems.

Chapter 2

The Robotic Blimp

Recently, the robotics community has shown an increasing interest in small-sized and low-cost unmanned aerial vehicles (UAVs) such as helicopters, quadrotors, or blimps. Especially their low power-consumption and safe navigation capabilities make blimps ideally suited for long-term indoor operation tasks. In this chapter, we present the design of a robotic blimp for autonomous indoor navigation. The hardware as well as the software system of our blimp is designed in a modular way so that it can be easily adapted for various autonomous navigation experiments. Furthermore, we introduce environment models, the formal state and control definition for miniature blimps, and methods to obtain accurate ground truth state information. We finally present a motion model for miniature indoor airships, which physically simulates the motion based on forces and torques and which enables a probabilistic motion prediction.

• • • • •

In this chapter, we describe our robotic blimp system for autonomous navigation in complex indoor environments. During the development of the hardware and the software system of our blimp, the design was focused on a modular system that can be easily adapted to various navigation tasks. Consequently, the robotic blimp platform enables an autonomous navigation and is suitable to experimentally evaluate the algorithms for state estimation and planning presented throughout this thesis. Furthermore, we introduce the environment model as well as the state and control model we apply for autonomous navigation. Additionally, we describe the physical simulation-based control motion model that is used for simulation experiments and for predicting the motion of the blimp during localization and planning.

Figure 2.1: Our first prototype of the robotic blimp. It is equipped with three sonar sensors at the front, left, and right side of the hull as well as a sonar sensor integrated in the gondola pointing downwards. The IMU is placed on top of the hull.

2.1 Hardware System

For autonomous navigation and the experimental evaluation of the algorithms introduced in this thesis, we developed a robotic blimp platform. The basic version of the hardware and the drivers of the blimp was developed by Rottmann et al. [143] within the PhD Program "Embedded Microsystems" of the University of Freiburg. During our work on autonomous navigation for robotic airships, we substantially extended the basic blimp platform by new sensor concepts, a new lightweight hull, and a completely redesigned gondola. A more detailed description of the basic blimp system and parts of our extensions can be found in the work of Rottmann [141]. In the following, we describe our extensions of the robotic blimp platform and present three stages of development, which were used in the experimental evaluations of our approaches to autonomous navigation.

2.1.1 The First Prototype

Compared to the basic blimp system [141, 143], our first prototype is equipped with three additional sonar sensors and the IMU is mounted in a more suitable way. Our first prototype, which is shown in Figure 2.1, was used in the evaluation of the probabilistic sonar model introduced in Chapter 4 and in the evaluation of our approach to simultaneous localization and parameter estimation (see Chapter 7). It has a length of approx. $1.8\,\mathrm{m}$ and is equipped with four tiny Devantech SRF10 sonar sensors with a measurement range of up to $6\,\mathrm{m}$. Each sensor has a membrane diameter $D \approx 8.5\,\mathrm{mm}$ and

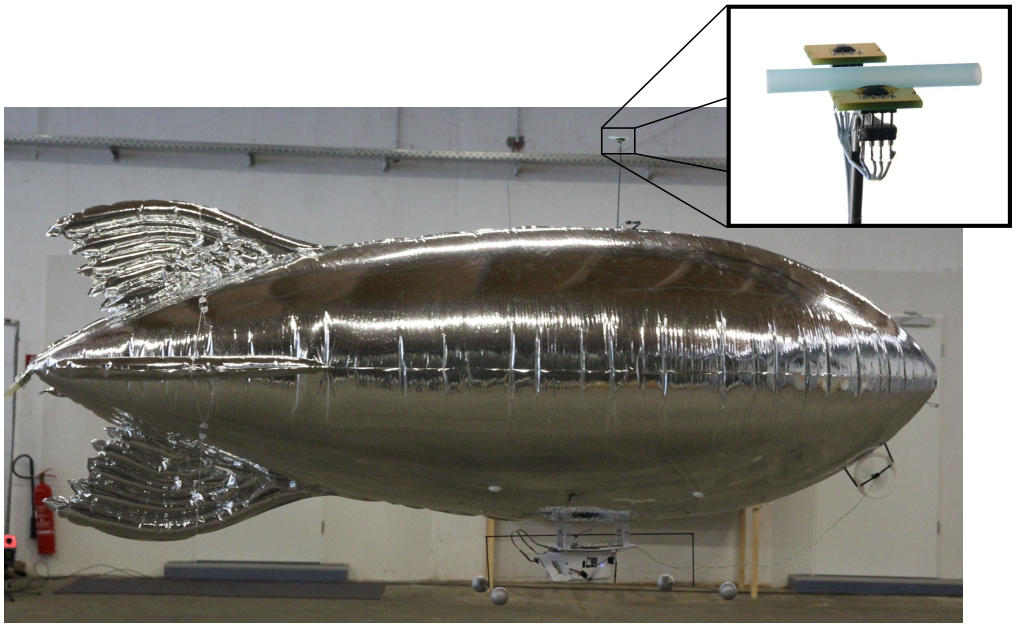


Figure 2.2: Our second prototype of the robotic blimp. It is based on a lightweight hull with a front rotor for yaw rotations. The prototype is equipped with two air flow sensors, a plain sensor and one with a short piece of a tube, an IMU, and four retroreflective markers for obtaining ground truth poses.

a wavelength $\lambda = 8.5\,\text{mm}$. Three of them are mounted on the front, left, and right side of the hull for horizontal distance measurements. The fourth sonar sensor is integrated in the gondola pointing downwards for height measurements. The blimp is actuated by two main propellers that pivot together and provide thrust in the forward/backward and upward/downward directions. A third propeller is mounted laterally at the rear of the blimp for yaw rotation. Additionally, our first prototype is equipped with an IMU [156]. As opposed to the basic version of the blimp in which the IMU was integrated in the gondola, we placed the IMU on top of the hull. This reduces the influence of magnetic disturbances of the power lines and the motors and avoids that the main rotors, which are mounted at the gondola, affect the vibration-sensitive MEMS sensors of the IMU.

2.1.2 The Second Prototype

Our second prototype provides a new design of the hull and is equipped with two air flow sensors as shown in Figure 2.2. It was used in the evaluation of the air flow sensor model (see Chapter 5) and in the evaluation of our planning and control algorithms (see Chapter 10). The air flow model training and the autonomous navigation were based on the accurate state estimates of an optical motion capture system (described in detail in Section 2.5). The retroreflective motion capture markers are exposed on lightweight

carbon tubes on the bottom of the blimp to achieve an optimal visibility of the markers when tracking the blimp with cameras standing on the floor. The hull of our second prototype is custom-built by Effekt-Technik GmbH, Schlaitdorf, Germany. It is longer (approx. 2.1 m long) and slimmer than the basic one. This changes the navigation properties towards a more stable forward motion as well as a reduced swinging during operation. Additionally, the hull is lighter and coated with aluminum so that it is less permeable to the filling gas helium. As a result of the reinforcement learning controller introduced by Rottmann [141], we mounted the third rotor at the front, which substantially simplifies manual as well as autonomous control of the blimp. Furthermore, we equipped our second prototype with two SDP600 differential pressure sensors [153] from Sensirion AG, Stäfa, Switzerland, operated here as thermal flow sensors. We mounted a short piece of a tube on one of the sensors in order to reduce the turbulences and achieve a higher measurement quality. As a result of our evaluation in Chapter 5, we mounted these two air flow sensors on a pole on top of the blimp in a distance of approx. 20 cm to the hull.

2.1.3 The Third Prototype

Our third prototype is the latest stage of development and is shown in Figure 2.3. It facilitates a weight-saving design for an ample sensor configuration for efficient and accurate localization based on on-board sensors. Our third prototype was used in the evaluation of the IMU sensor model (see Chapter 6) and in the evaluation of the odometry motion model (see Chapter 8). The completely redesigned gondola (see Figure 2.4) provides a lightweight frame built of carbon tubes so that all devices and their connectors are easily accessible during setup, maintenance, and debugging. Furthermore, we achieve a high modularity in the gondola by applying a Lego® push-fit system so that components can be easily added and removed depending on the individual navigation task. The hull of our third prototype is custom-built by Effekt-Technik GmbH, Schlaitdorf, Germany and almost transparent to avoid reflections, which can disturb the optical tracking by the motion capture system. Apart from that, it has exactly the same shape and rotor setup as the hull of our second prototype. The retroreflective markers for optical tracking are placed on the top of the blimp, which has two desirable properties. First, it enables an ideal tracking of the blimp with cameras mounted on the ceiling of a structured or cluttered indoor environment. Second, it is weight-saving and provides a more rigid design than the frame of thin carbon tubes so that we can obtain more accurate orientation estimates. As a sensor setup, we chose five tiny sonar sensors, three air flow sensors and an IMU. Four of the Devantech SRF10 sonar sensors are mounted at the front, back, left, and right side of the hull for horizontal distance measurements. The fifth sensor is attached to the gondola pointing downwards for height measurements. Additionally, we equipped the blimp with three air flow sensors. As a result of our sensor setup evaluation described in Chapter 5, we mounted a short piece of a tube on each sensor. Two of these sensors were

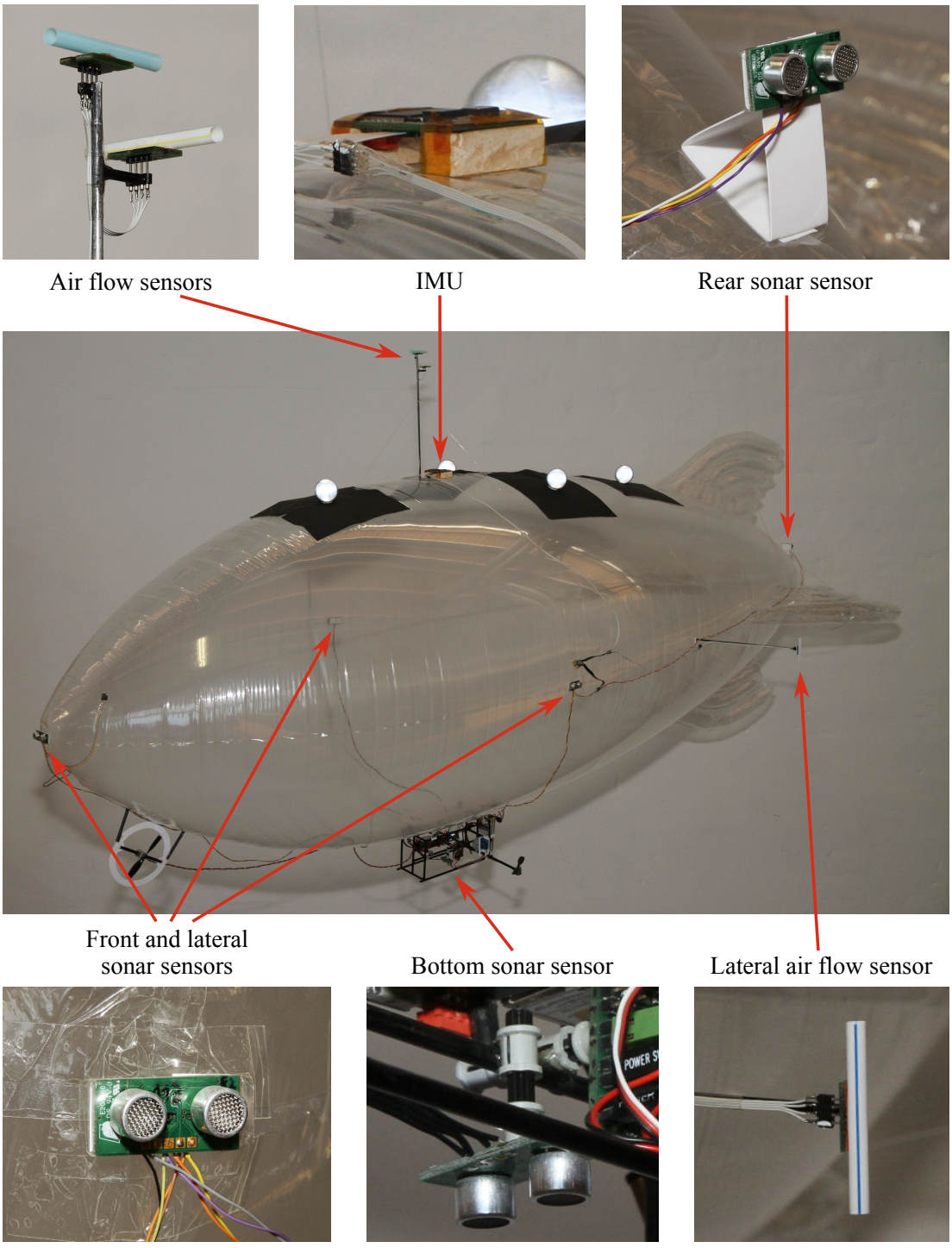


Figure 2.3: Our third prototype of the robotic blimp. It is equipped with four sonar sensors at the front, back, left, and right side of the hull. The fifth sonar sensor is pointing downwards and integrated in the new gondola, which provides a lightweight, modular, and easily accessible frame for the electric devices. The IMU is placed on the top of the hull. Three air flow sensors are mounted on poles for velocity measurements along all three axes.

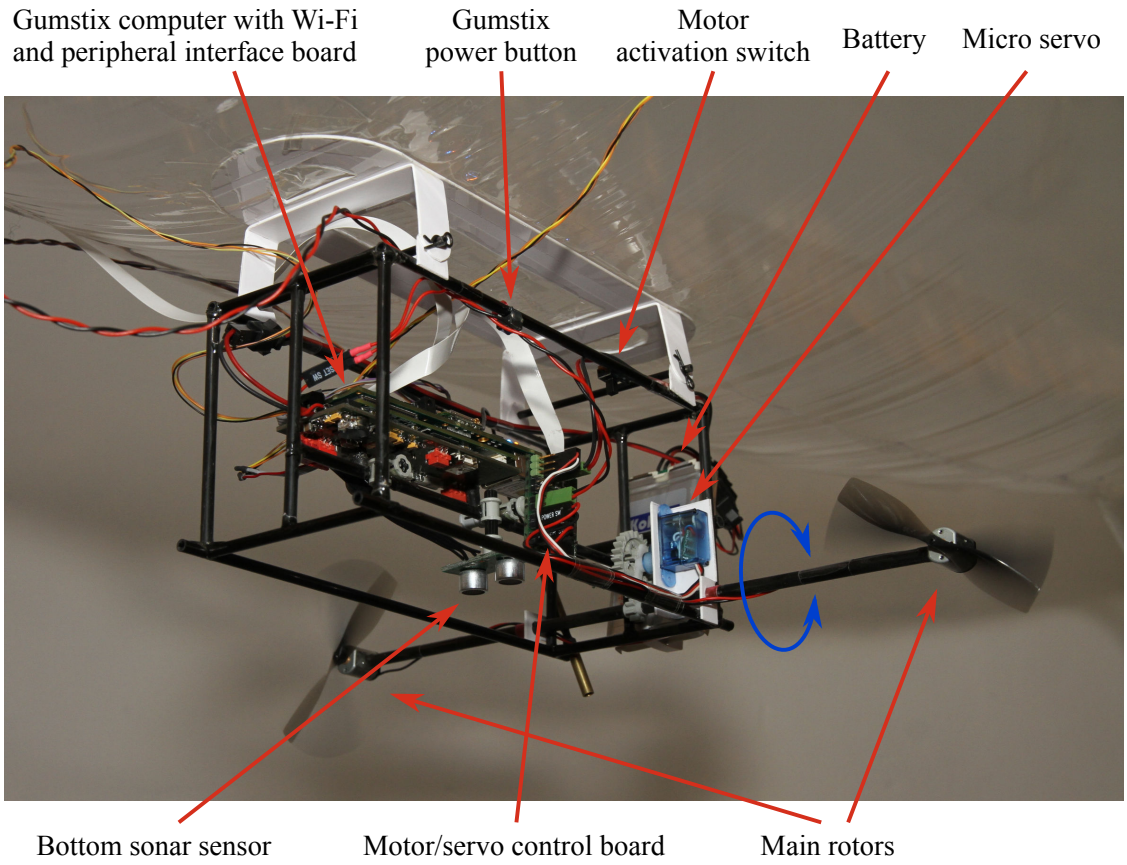


Figure 2.4: The gondola of our third prototype of the robotic blimp provides a lightweight and accessible frame for the electric devices. The components are mounted using a Lego® push-fit system so that the configuration can be easily changed depending on the navigation task.

Quantity	Component	Weight (each)
1	Gumstix Verdex pro XL6P COM	7 g
1	Gumstix Ethernet Board with Wi-Fi and microSD-card	18 g
1	Wi-Fi antenna	4 g
1	Gumstix Peripheral Interface Card	20 g
1	Motor/Servo Control Board	7 g
1-2	1-cell LiPo Battery	33 g
5	Devantech SRF10 sonar sensors	3 g
3	Sensirion SDP600 thermal air flow sensors	1 g
1	Custom-made IMU	9 g
4	Custom-made retroreflective markers	3 g

Table 2.1: The main components of our robotic blimp.

mounted on top of the blimp for velocity measurements in the forward and sideward direction. The third sensor was mounted laterally to measure the vertical velocity. All air flow sensors were placed on a pole in a distance of approx. 20 cm to the hull. This distance turned out to be a good trade-off between a low influence of the air accompanying the blimp and good navigation capabilities in narrow passages. For the calibration of our models, we determined the geometry of the hull and the gondola as well as the sensor positions in an accurate way using our optical motion capture system. The main components of our third prototype are listed in Table 2.1.

Figure 2.5 gives an overview of the electric hardware modules of our third prototype. Our blimp is operated by a Gumstix Verdex pro^{TM}XL6P Computer-on-module (COM) [62] with a Marvell PXA270 CPU @ 600 MHz implementing the ARMv5 architecture, 128 MB memory, and 32 MB non-volatile flash memory. It is extended by the Gumstix Ethernet board with a Wi-Fi module and a microSD-card slot for additional non-volatile memory. The custom-made Gumstix Peripheral Interface Card (GPIC) provides regulated power with 3.3 V, 5 V, and 6 V to the Gumstix and all sensors [143]. Furthermore, it monitors the voltage of the battery and provides convenient connectors to the UART, USB, and I^2C interfaces of the Gumstix. The custom-made Motor/Servo Control Board [143] is operated via UART by the Gumstix and controls the three rotors of the blimp as well as the servo pivoting the main rotors attached to the gondola. The sensors described above are also connected to the GPIC. We minimized the weight of the I^2C cables by adding a custom-made, lightweight voltage converter board to the I^2C bus between the sonars and the air flow sensors. Optionally, a USB camera can be connected to the GPIC. However, as the computational power of the Gumstix is severely limited, only a low-resolution camera with on-board JPEG compression turned out to be usable on the blimp.

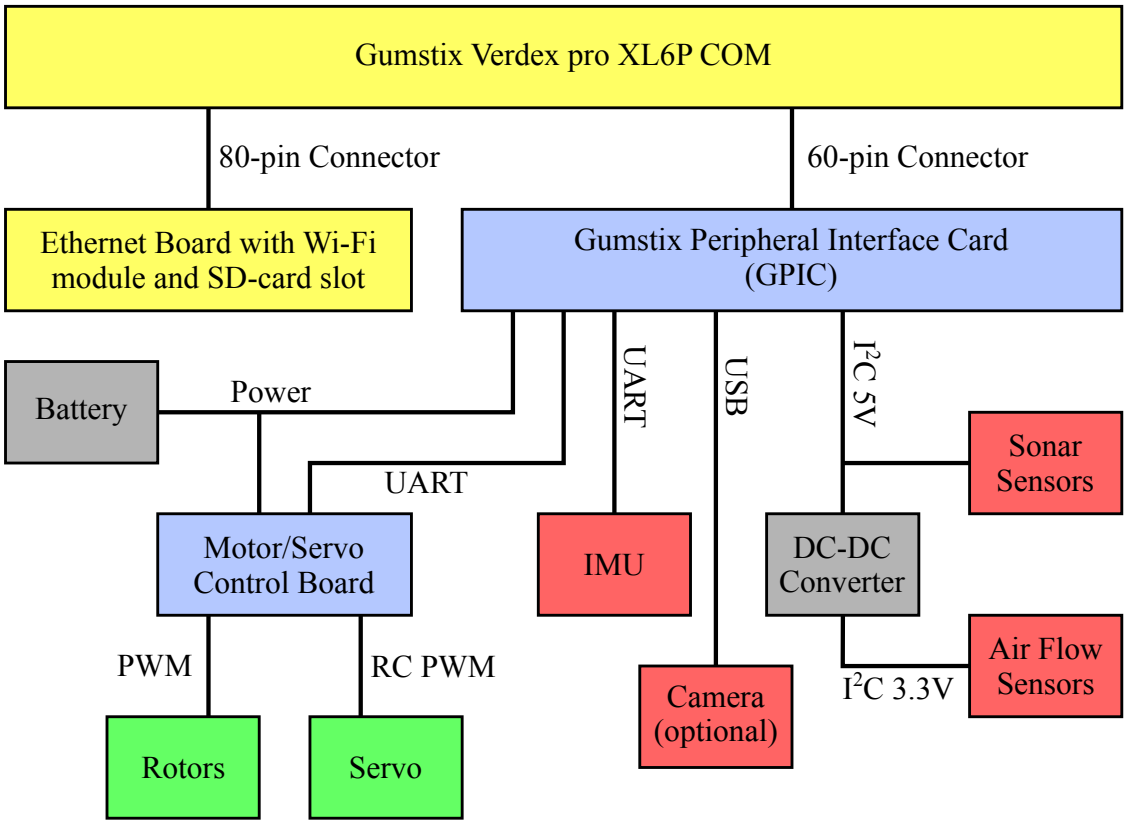


Figure 2.5: The electric hardware modules of our blimp and their connections. The components purchased from Gumstix Inc. are shown in yellow, the custom-made interface and control boards are shown in light blue, the pure power supply devices are shown in gray, the sensors are shown in red, and the actuators are shown in green.

In addition to the electric components, our second and third prototypes are equipped with four retroreflective motion capture markers. To obtain 3D ground truth poses, at least three markers are required as described in Section 2.5. We applied four markers to obtain a trade-off between a low weight and a redundant marker setup for robust optical tracking.

Overall, our robotic blimp platform provides a flexible sensor configuration, a lightweight embedded system with wireless communication capabilities, and a weight-saving design of the hull and the gondola for autonomous navigation in various applications.

2.2 Software Architecture

The software architecture of our robotic blimp was designed to be flexible and reusable in diverse navigation tasks. We achieve these design goals through a modular software framework that provides a generic interface of modules and allows different communication modes between modules for both, local as well as distributed computations. For the inter process communication (IPC) in distributed computations, we apply the IPC framework of Simmons and James [155], as shown in Figure 2.6.

2.2.1 Operating System

Despite the limited computational power and the severe memory constraints of the Gumstix, we run a complete Linux operating system on the embedded computer. In particular, we use the OpenEmbedded [127] build framework for embedded Linux, which provides a cross-compilation environment and numerous BitBake [12] recipes accounting for the dependencies between the individual software packages. Based on this framework, we have built a compact Linux operating system with kernel 2.6.31 providing all required kernel modules as well as TCP/IP network support via the Wi-Fi module and all important Linux system tools such as the OpenSSH server and the common file system tools. Figure 2.6 gives an overview of the operating system forming the basis of the blimp software architecture.

2.2.2 Hardware Drivers

The hardware drivers provide a convenient high-level interface to the devices connected to the Gumstix. Specifically, the hardware drivers of the sensor devices run in a loop, poll the sensors for new measurement data, and publish available data to the IPC central server. Correspondingly, the hardware driver of the actuator devices, i.e. the motor/servo control board, listens to the IPC central server and applies incoming control commands to the actuators of the blimp. The hardware drivers of our current blimp system are

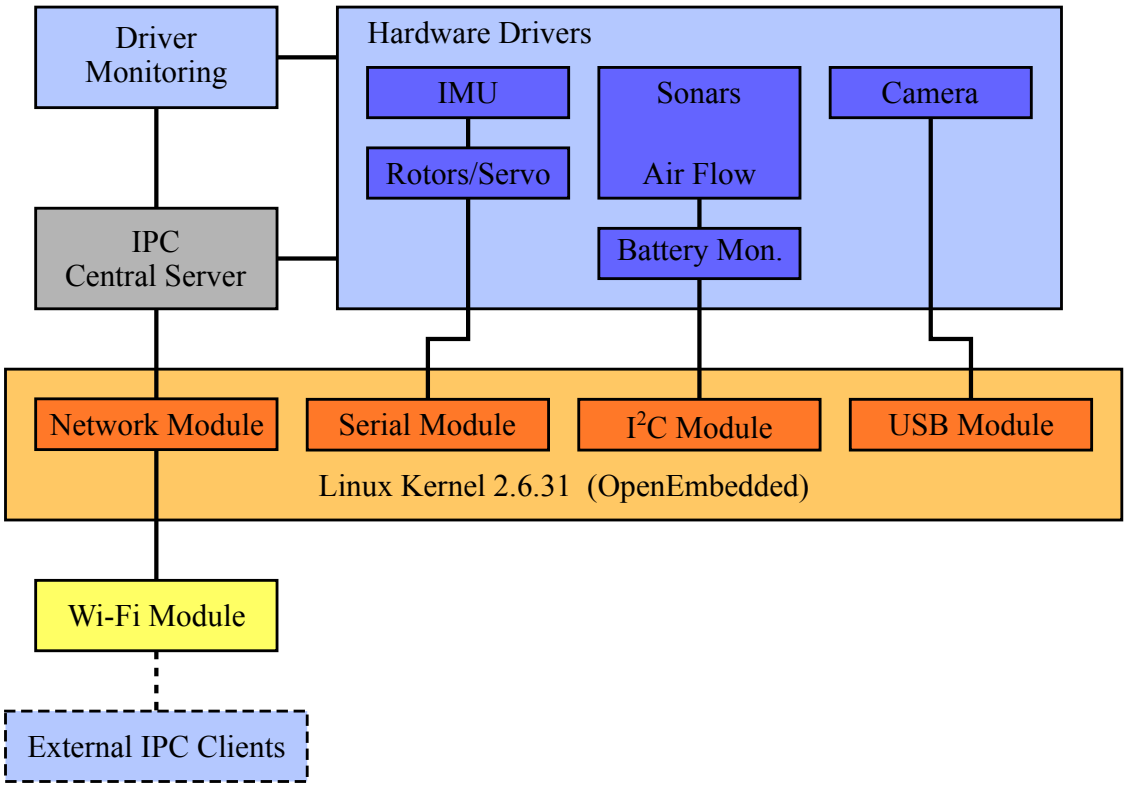


Figure 2.6: The software architecture of our blimp when it is operated by external IPC clients. The operating system with the important kernel modules is shown in orange and the custom-built modules are shown in light blue. The dark blue boxes indicate the individual threads in the hardware driver executable running on the Gumstix.

based on the work of Rottmann et al. [141, 143]. We implemented a single hardware driver binary for an easy software handling during experimental setup and maintenance. Within the hardware driver binary, the individual drivers are running in separate threads with an alternating access to the corresponding kernel modules, which is coordinated via mutual exclusions where needed. As an exception, the sonar and air flow sensors are implemented in a common thread, because a manual coordination of their measurement process (consisting of triggering a measurement, waiting, and receiving the result) significantly improves the measurement frequency of the air flow sensors. Additionally, all driver components are continuously monitored. Thus, the measurement frequencies of all drivers together with the Gumstix temperature and the battery voltage can be graphically presented to the user, e.g. in an external IPC client module on a base station. The average measurement frequencies are 51 Hz for the air flow sensors, 84 Hz for the IMU when providing temperature calibrated measurement data of all integrated sensors, and 8.5 Hz for the sonar sensors. Since the blimp is equipped with three air flow and five sonar sensors, these sensors are triggered in sequence so that the measurement frequency of each individual sensor is 17 Hz for air flow and 1.7 Hz for sonar sensors.

2.2.3 Flexible Modular High-level Software Framework

Especially in complex navigation tasks like the tasks presented throughout this thesis, it is beneficial to develop and test the individual parts of the software system independently. Furthermore, for distributed computing in online operation, a way of data transport between different processes or different machines is desirable. Additionally, an easy data recording in log files and an efficient offline processing of recorded log files is important.

We created a flexible software framework, named Robular, that enables us to develop all parts of the system as individual modules and to couple them in different ways depending on the navigation task. The key idea of Robular is the specification of a generic software interface of modules without imposing any constraints on the format of the data exchanged between the modules. In contrast to many other software frameworks in robotics, for example CARMEN [113], ROS [135], MOOS [125], and YARP [47], Robular strictly separates the interface of modules from the data transport. In particular, Robular specifies inputs and outputs, which are parameterized by the message type. According to the interface, such inputs and outputs can be integrated into modules and provide generic connection points to other modules, as each output can be connected to one or multiple inputs with the corresponding message type. This has the substantial advantage that also the transport modules are implementing the interface of modules, which allows to flexibly choose the data transportation mode between modules by simply connecting them in different ways. Modules can be directly connected to each other (without serializing data for transportation in between) or arbitrary transport modules can be plugged between two or more modules to exchange data between different processes.

Hence, multiple modules can be connected in three different ways: via triggering connections for synchronous processing in a single thread, via non-triggering connections for asynchronous processing in multiple threads in a single process, and via triggering connections to an arbitrary transport module. The transport module usually is a logging module for recording and playback of log files or an inter process communication module (we use IPC [155] in our implementation) for asynchronous processing in multiple processes, optionally on one or multiple machines. All three connection methods can be arbitrarily mixed and each module can be used with all three methods without any change in the implementation of the module itself.

This modular software framework allows us to apply a distributed computing approach in the evaluation of the algorithms for autonomous navigation. In particular, we utilize three computers, namely the Gumstix our blimp is equipped with, a Linux laptop computer, and a Windows laptop computer. As shown in Figure 2.6, the Gumstix provides access to the hardware devices of the blimp. The blimp is controlled by one or several external IPC client modules running on the Linux laptop computer, whereas the Windows laptop computer is responsible for operating the motion capture cameras and providing accurate 3D positions of the reflective markers to the Linux laptop computer with high frequency.

In addition to the closed-loop control scheme for autonomous navigation described above, we can replace the real blimp and the motion capture system by the simulator described in Section 2.6.4 for an uncomplicated evaluation of planning and control algorithms. When evaluating our localization approaches, we applied the distributed computing scheme described above. We thereby manually controlled the blimp via a joystick connected to the Linux laptop computer. We recorded all sensor and control data together with the accurate reference poses from the motion capture system. This data set is later used as input for multiple offline evaluations of the localization approaches in an efficient synchronous way.

2.3 Environment Models

The model of the environment, which we call *map* throughout this thesis, plays a twofold role in autonomous navigation. First, the map relates the measurements of sensors to the pose of the robot in the localization task. Second, in the planning and control task, the map encodes the traversability of the environment.

To relate the measurements of sensors to the state of the robot, the representation of the environment has to be suitable for the particular sensor device. Accordingly, there are various types of environment representations. For sensors measuring the distance to objects in the environment, such as the sonar sensors our blimp is equipped with, the most popular type of map is the occupancy map. For three-dimensional mapping,

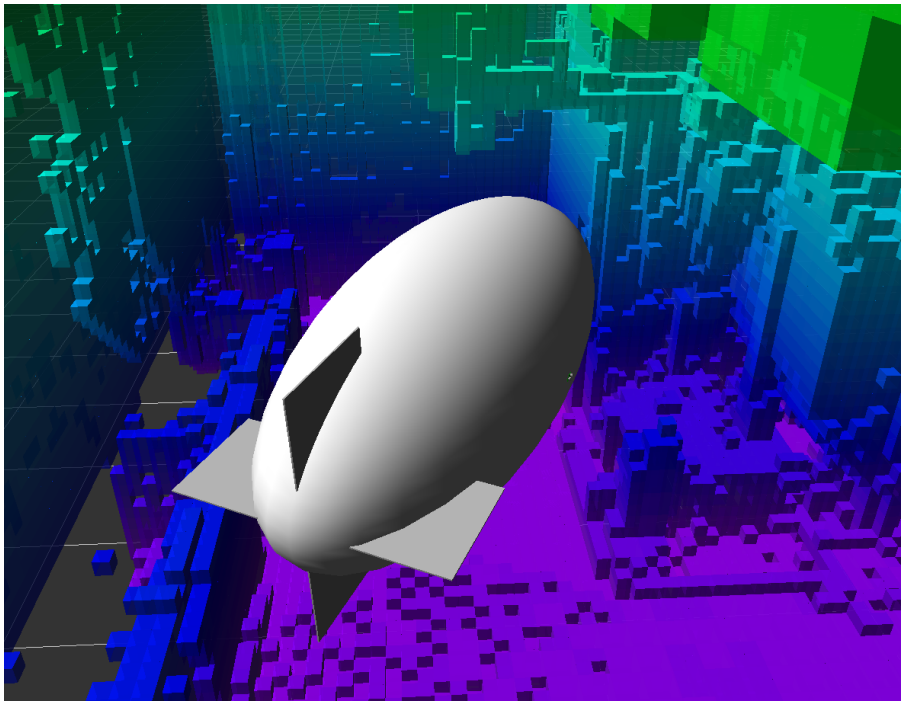


Figure 2.7: The visualization of our first blimp prototype in an OctoMap with 10 cm resolution. Each cell of the map is colored depending on its z-coordinate.

an occupancy map represents the environment through discrete volume elements, often cubes on a regular grid where each element stores whether the represented volume element is occupied by an object of the environment. Additionally, an occupancy map provides information about the traversability of the environment as volume elements that are non-occupied are typically traversable.

In this work, we apply 3D occupancy grid maps to represent the environment. In the implementation of our first prototype, we employ multi-level surface maps [172], which are based on a two-dimensional regular grid. Each of the grid cells contains a list of vertical ranges in which the cell is occupied. For our second and third prototype, we modeled the environment using the OctoMap framework [180]. This framework provides a tree-based map structure representing the occupancy of 3D volume elements in a hierarchical, memory-efficient fashion.

We created the map of an environment by collecting distance measurements using a laser range finder. In particular, the environment was three-dimensionally scanned using a wheeled platform with a panning or tilting 2D laser range finder. We obtained the trajectory of the laser range finder device either by scan matching and optimizing all collected data [59, 172] or from the optical motion capture system tracking markers attached to the device. The resulting distance measurements together with the trajectory of the laser range finder are then fused to the maximum likelihood estimate of the map

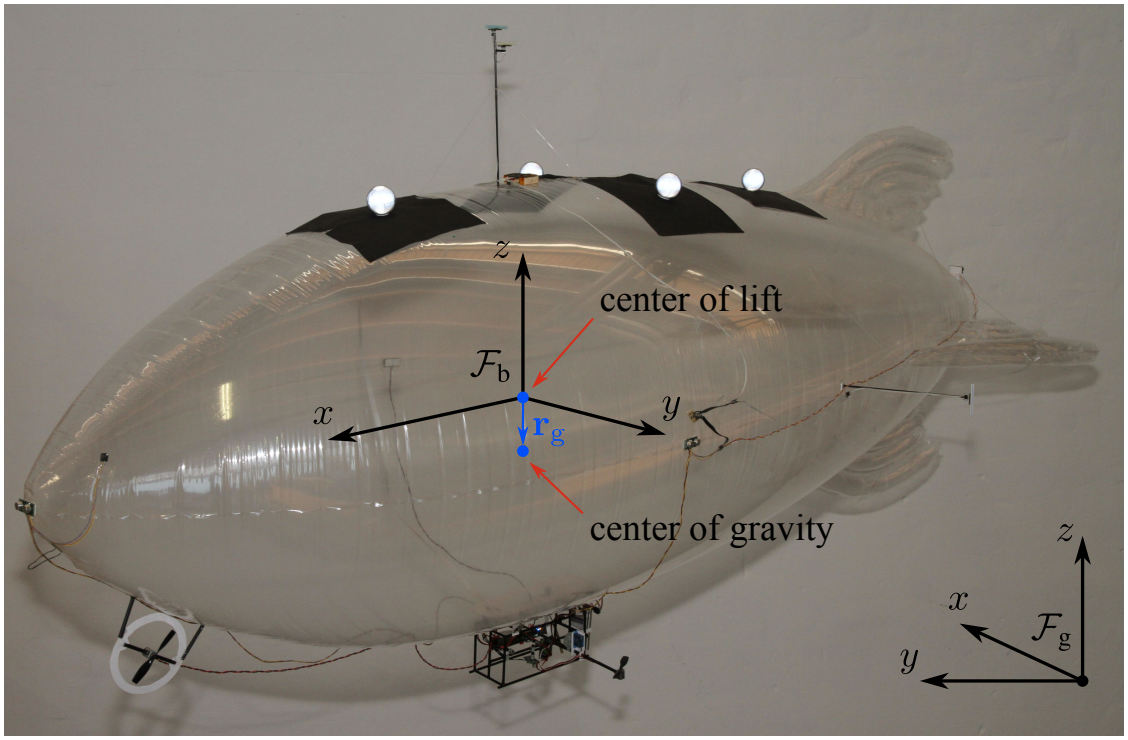


Figure 2.8: The body-fixed frame of reference $\mathcal{F}_\mathrm{b}$ of the blimp and the global frame of reference $\mathcal{F}_\mathrm{g}$. The center of gravity of the blimp is located at the position $\mathbf{r}_\mathrm{g}$ with respect to $\mathcal{F}_\mathrm{b}$.

in a 3D occupancy mapping algorithm [71]. A resulting OctoMap with a resolution of 10 cm is shown in Figure 2.7.

2.4 State and Control

For autonomous navigation, we model our blimp as a floating rigid body in a three-dimensional environment. To describe the state of the blimp relative to its environment, we define two frames of reference as shown in Figure 2.8:

- The *body-fixed frame of reference* $\mathcal{F}_\mathrm{b}$ is located in the center of lift of the blimp. Its x-axis is pointing forwards, its y-axis is pointing to the left, and its z-axis is pointing upwards.

- The *global frame of reference* $\mathcal{F}_\mathrm{g}$ (with its z-axis pointing upwards) is defined by the map of the environment.

Throughout this thesis, we express the state of the blimp by its pose, its velocity, and/or its acceleration, depending on the particular task. We define the *pose* of the blimp by the position and the orientation of the body-fixed frame $\mathcal{F}_\mathrm{b}$ relative to the global frame

$\mathcal{F}_\mathrm{g}$. The position is represented by a three-dimensional vector $\mathbf{p} = [x, y, z]^T$ and the orientation is represented by a unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$ defining the orientation in three-dimensional space. A detailed description of three-dimensional orientations and unit quaternions is provided by Diebel [37]. Thus, the position and the orientation are defined with respect to the global frame $\mathcal{F}_\mathrm{g}$.

The velocity and the acceleration of the blimp are both defined by the motion of $\mathcal{F}_\mathrm{b}$ relative to $\mathcal{F}_\mathrm{g}$. We express both, the velocity and the acceleration, as a six-dimensional vector containing a three-dimensional translational and a three-dimensional rotational part. We express all four components, the translational velocity $\mathbf{v} = [v_x, v_y, v_z]^T$, the rotational velocity $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$, the translational acceleration $\mathbf{a} = \dot{\mathbf{v}} = [a_x, a_y, a_z]^T$, and the rotational acceleration $\boldsymbol{\alpha} = \dot{\boldsymbol{\omega}} = [\alpha_x, \alpha_y, \alpha_z]^T$ in the body-fixed frame $\mathcal{F}_\mathrm{b}$.

The blimp is actuated by three propellers. Two of them are mounted next to the gondola and are pivoted together to provide thrust along the forward- and the up-axis. The third propeller is mounted laterally for yaw rotation. The blimp can be controlled by a three-dimensional vector $\mathbf{u} \in \mathcal{U} = [-1, 1]^3$ defining the relative translational thrust in the forward direction, the relative translational thrust in the upward direction, and the relative rotational thrust about the vertical axis – all of them with respect to the body-fixed frame $\mathcal{F}_\mathrm{b}$.

2.5 Ground Truth States

For evaluation and model learning purposes, we define the ground truth state

$$\mathbf{x}^\star = \left[\mathbf{p}^{\star T}, \mathbf{q}^{\star T}, \mathbf{v}^{\star T}, \boldsymbol{\omega}^{\star T}, \mathbf{a}^{\star T}, \boldsymbol{\alpha}^{\star T}\right]^T \ , \tag{2.1}$$

which includes the pose, the velocity, and the acceleration of the robot.

In our practical experiments, we used two methods to obtain ground truth information. In the experiments with our first prototype, we placed visual AR Toolkit Plus [15] markers on the floor (see Figure 2.9), which allow us to accurately determine the pose of the vehicle using the camera integrated in the gondola of the blimp [15]. However, this method only allows us to determine the ground truth pose as long as enough markers are in the field of view of the camera. Since the partial ground truth trajectories obtained with this method suffer from high uncertainty, they cannot be used to extract ground truth velocities and accelerations.

2.5.1 Reference Poses from Optical Motion Capture

In the experiments with our second and third prototype, we attached retroreflective markers to the blimp (see Figures 2.2 and 2.3). The three-dimensional positions of these

Figure 2.9: Our first prototype of the blimp in the experimental environment. The reference poses are obtained using the camera integrated in the gondola pointing downwards and observing the visual markers placed on the floor.

markers were accurately tracked by an optical motion capture system of Motion Analysis Corporation with eight to ten digital Raptor-E cameras, each of them having a high power infrared ring light around the lens. The cameras synchronously take images at up to 300 Hz and accurately extract the markers, which are seen as bright spots, from their images. The three-dimensional positions of the markers are obtained through triangulation using the calibration parameters of the camera setup. The calculated positions are associated to the markers based on their rigid geometry, and the labeled data is provided via network.

In an initial calibration procedure, we determine the positions $\widetilde{\mathbf{m}}_1, \ldots, \widetilde{\mathbf{m}}_\ell$ of the ℓ markers relative to the body-fixed frame $\mathcal{F}_\mathrm{b}$ by temporarily placing additional markers on the coordinate axes of $\mathcal{F}_\mathrm{b}$.

In each discrete time step, we receive the marker position data $\mathbf{m}_1, \ldots, \mathbf{m}_\ell$ from the motion capture system in a module of our software framework and estimate the rigid body pose of the blimp. We identify the pose as the transformation parameters, the rotation matrix R and the translation vector $\mathbf{t}$, between the marker positions $\widetilde{\mathbf{m}}_1, \ldots, \widetilde{\mathbf{m}}_\ell$ and $\mathbf{m}_1, \ldots, \mathbf{m}_\ell$ by minimizing the mean squared error

$$e^2(R, \mathbf{t}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \|\mathbf{m}_i - (R\widetilde{\mathbf{m}}_i + \mathbf{t})\|^2 \tag{2.2}$$

between the marker position in $\mathcal{F}_\mathrm{b}$ transformed by R and $\mathbf{t}$ and the observed marker positions. To this end, we follow the least squares approach of Umeyama [174]. In a first step, we calculate the means

$$\boldsymbol{\mu}_\mathbf{m} = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{m}_i \qquad \text{and} \qquad \boldsymbol{\mu}_{\widetilde{\mathbf{m}}} = \frac{1}{\ell} \sum_{i=1}^{\ell} \widetilde{\mathbf{m}}_i \tag{2.3}$$

of both sets of marker positions. We then estimate the joint covariance of both sets of marker positions as

$$\Sigma_{\mathbf{m}\widetilde{\mathbf{m}}} = \frac{1}{\ell} \sum_{i=1}^{\ell} (\mathbf{m}_i - \boldsymbol{\mu}_\mathbf{m}) \, (\widetilde{\mathbf{m}}_i - \boldsymbol{\mu}_{\widetilde{\mathbf{m}}})^T \, . \tag{2.4}$$

The optimal rotation matrix with respect to Equation (2.2) is

$$R = USV^T \tag{2.5}$$

where UDV^T is the singular value decomposition of $\Sigma_{\widetilde{\mathbf{m}}\mathbf{m}}$ and

$$S = \begin{cases} I_{3\times 3} & \text{if } \det(U)\det(V) = 1 \\ \mathrm{diag}(1, 1, -1) & \text{if } \det(U)\det(V) = -1 \, . \end{cases} \tag{2.6}$$

Algorithm 1 MOTIONREGRESSION1D

Input: A sequence of values $\big((x_1, t_1), \ldots, (x_n, t_n)\big)$ with corresponding time stamps and a target time t.

Output: The first derivative $v := \dot{x}(t)$ and second derivative $a := \ddot{x}(t)$ at the target time t.

// compute various sums
$(s_x, s_{tx}, s_{t^2x}, s_t, s_{t^2}, s_{t^3}, s_{t^4}) \leftarrow (0, \ldots, 0)$
for $i = 1$ to n **do**
  $t_i \leftarrow t_i - t$
  $s_x \leftarrow s_x + x_i$
  $s_{tx} \leftarrow s_{tx} + x_i \cdot t_i$
  $s_{t^2x} \leftarrow s_{t^2x} + x_i \cdot t_i^2$
  $s_t \leftarrow s_t + t_i$
  $s_{t^2} \leftarrow s_{t^2} + t_i^2$
  $s_{t^3} \leftarrow s_{t^3} + t_i^3$
  $s_{t^4} \leftarrow s_{t^4} + t_i^4$
end for
// matrix inversion pre-factor
$A \leftarrow n\,(s_{t^3}\, s_{t^3} - s_{t^2}\, s_{t^4}) + s_t\,(s_t\, s_{t^4} - s_{t^2}\, s_{t^3}) + s_{t^2}\,(s_{t^2}\, s_{t^2} - s_t\, s_{t^3})$
// velocity at time t
$v \leftarrow A^{-1}\,(s_x\,(s_t\, s_{t^4} - s_{t^2}\, s_{t^3}) + s_{tx}\,(s_{t^2}\, s_{t^2} - n\, s_{t^4}) + s_{t^2x}\,(n\, s_{t^3} - s_t\, s_{t^2}))$
// (constant) acceleration
$a \leftarrow 2A^{-1}\,(s_x\,(s_{t^2}\, s_{t^2} - s_t\, s_{t^3}) + s_{tx}\,(n\, s_{t^3} - s_t\, s_{t^2}) + s_{t^2x}\,(s_t\, s_t - n\, s_{t^2}))$
return (v, a)

The optimal translation can be expressed as

$$\mathbf{t} = \boldsymbol{\mu}_{\mathbf{m}} - R\boldsymbol{\mu}_{\widetilde{\mathbf{m}}}\ , \tag{2.7}$$

and finally, the ground truth pose is $\mathbf{p}^\star = \mathbf{t}$, and $\mathbf{q}^\star$ is the quaternion obtained from the rotation matrix R in a straightforward way [37].

2.5.2 Reference Velocities and Accelerations from a Pose Time Sequence

Due to the high precision of the motion capture system, the obtained sequence of poses is afflicted with a comparably low noise. Therefore, in an offline post processing step, we can accurately compute the ground truth velocities and accelerations through a regression in a shifting window of size Δt on the sequence of poses. In a nutshell,

Algorithm 2 MOTIONREGRESSION6D

Input: A sequence of positions and orientations $T = \big((\mathbf{p}_1, \mathbf{q}_1, t_1), \ldots, (\mathbf{p}_n, \mathbf{q}_n, t_n)\big)$ with corresponding time stamps and a target time t as well as the orientation $\mathbf{q}_t$ at time t.

Output: The translational and rotational velocities $\mathbf{v}, \boldsymbol{\omega}$ and accelerations $\mathbf{a}, \boldsymbol{\alpha}$ in the body-fixed frame of reference at the target time t.

$(\mathbf{v}, \boldsymbol{\omega}, \mathbf{a}, \boldsymbol{\alpha}) \leftarrow (\mathbf{0}, \ldots, \mathbf{0})$
$(\dot{\mathbf{q}}, \ddot{\mathbf{q}}) \leftarrow (\mathbf{0}, \mathbf{0})$ *// initialize the quaternion rates with zero in all dimensions*
// run motion regressions independently in every dimension
for all $i \in \{x, y, z\}$ **do**
  $(\mathbf{v}_i, \mathbf{a}_i) \leftarrow$ MOTIONREGRESSION1D$\big(\big((\mathbf{p}_{i,1}, t_1), \ldots, (\mathbf{p}_{i,n}, t_n)\big), t\big)$
end for
for all $i = 0$ to 3 **do**
  $(\dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i) \leftarrow$ MOTIONREGRESSION1D$\big(\big((\mathbf{q}_{i,1}, t_1), \ldots, (\mathbf{q}_{i,n}, t_n)\big), t\big)$
end for
// transform the derivatives to the body-fixed frame of reference
$\left[\begin{smallmatrix} 0 \\ \mathbf{v} \end{smallmatrix}\right] \leftarrow \mathbf{q}_t^{-1} \odot \left[\begin{smallmatrix} 0 \\ \mathbf{v} \end{smallmatrix}\right] \odot \mathbf{q}_t$
$\left[\begin{smallmatrix} 0 \\ \mathbf{a} \end{smallmatrix}\right] \leftarrow \mathbf{q}_t^{-1} \odot \left[\begin{smallmatrix} 0 \\ \mathbf{a} \end{smallmatrix}\right] \odot \mathbf{q}_t$
$\left[\begin{smallmatrix} 0 \\ \boldsymbol{\omega} \end{smallmatrix}\right] \leftarrow 2\,\mathbf{q}_t^{-1} \odot \dot{\mathbf{q}}$
$\left[\begin{smallmatrix} 0 \\ \boldsymbol{\alpha} \end{smallmatrix}\right] \leftarrow 2\,\mathbf{q}_t^{-1} \odot \ddot{\mathbf{q}}$
// account for fictitious forces
$\mathbf{a} \leftarrow \mathbf{a} - \boldsymbol{\omega} \times \mathbf{v}$
return $(\mathbf{v}, \boldsymbol{\omega}, \mathbf{a}, \boldsymbol{\alpha})$

Algorithm 3 POSETIMESEQUENCEREGRESSION

Input: The pose time sequence (trajectory) $T = \big((\mathbf{p}_1, \mathbf{q}_1, t_1), \ldots, (\mathbf{p}_n, \mathbf{q}_n, t_n)\big)$ given as positions and orientations with corresponding time stamps and the regression window Δt.

Output: The sequence of translational and rotational velocities and accelerations $\big((\mathbf{v}_1, \boldsymbol{\omega}_1, \mathbf{a}_1, \boldsymbol{\alpha}_1), \ldots, (\mathbf{v}_n, \boldsymbol{\omega}_n, \mathbf{a}_n, \boldsymbol{\alpha}_n)\big)$ corresponding to the input trajectory.

$\big((\mathbf{v}_1, \boldsymbol{\omega}_1, \mathbf{a}_1, \boldsymbol{\alpha}_1), \ldots, (\mathbf{v}_n, \boldsymbol{\omega}_n, \mathbf{a}_n, \boldsymbol{\alpha}_n)\big) \leftarrow \big((\mathbf{0}, \ldots, \mathbf{0}), \ldots, (\mathbf{0}, \ldots, \mathbf{0})\big)$
for $i = 1$ to n **do**
  // extract the regression window of t_i and apply regression to it
  $T' \leftarrow \{(\mathbf{p}_j, \mathbf{q}_j, t_j) \mid t_j \in [t_i - \Delta t/2, t_i + \Delta t/2]\}$
  $(\mathbf{v}_i, \boldsymbol{\omega}_i, \mathbf{a}_i, \boldsymbol{\alpha}_i) \leftarrow$ MOTIONREGRESSION6D(T', t_i)
end for
return $\big((\mathbf{v}_1, \boldsymbol{\omega}_1, \mathbf{a}_1, \boldsymbol{\alpha}_1), \ldots, (\mathbf{v}_n, \boldsymbol{\omega}_n, \mathbf{a}_n, \boldsymbol{\alpha}_n)\big)$

this regression approach assumes a constant acceleration in the regression window and computes the velocity and acceleration through regression in each dimension. The resulting translational velocities and accelerations are transformed to the body-fixed frame $\mathcal{F}_\mathrm{b}$ according to our definition of $\mathbf{v}^\star$ and $\mathbf{a}^\star$ where we account for fictitious forces in the non-inertial body-fixed frame $\mathcal{F}_\mathrm{b}$. The corresponding rotational velocity $\boldsymbol{\omega}^\star$ and acceleration $\boldsymbol{\alpha}^\star$ can be obtained from the quaternion rates resulting from the regression [37]. While Algorithm 1 and 2 give the pseudo code of auxiliary functions, namely the one-dimensional and the six-dimensional regression, Algorithm 3 shows the offline post processing of a recorded trajectory by computing $\mathbf{v}_i^\star$, $\boldsymbol{\omega}_i^\star$, $\mathbf{a}_i^\star$, and $\boldsymbol{\alpha}_i^\star$ from the pose time sequence given by the motion capture system. The derivation of the algorithms and additional explanations can be found in Sittel et al. [157].

2.6 Physical Simulation-based Control Motion Model

The model of the motion of the robot plays an important role in autonomous navigation. It relates the chosen control commands to the motion of the robot and is mainly used in the following four tasks within autonomous navigation:

- In the path planning task, the planning algorithm needs to select suitable control commands to navigate the robot to its goal state. In this task, the planning algorithm usually considers various sequences of control commands and evaluates them depending on the predicted motion computed by the motion model.

- In the control tasks for complex robots such as our blimp, the motion model is often applied as a core component of a model predictive control approach. These approaches utilize the relation between controls and the motion of the robot to optimize the control commands in order to minimize the expected future deviation of the robot from its desired trajectory.

- In the localization task, the motion model is used to predict the motion of the robot given the applied control commands. In probabilistic localization, as described in Part I of this thesis, the motion model also needs to specify the uncertainty of the predicted motion.

- The evaluation of the autonomous navigation system is often extended by simulation experiments in which the motion model is used as a core component of the simulator of the physical robotic system.

The proper design of the motion model is essential for planning feasible trajectories, defining suitable closed-loop control laws, and accurately and efficiently estimating the state of the robot. An inaccurate motion model would result in a high deviation of the

robot from its desired trajectory in the planning and control task and in a wide-spread proposal distribution in the particle filter localization task, which increases the number of wasted particles and therefore decreases the efficiency of the filter.

In the following, we first derive a deterministic model by considering the underlying physics of the motion of miniature airships. In particular, our model is based on the work of Zufferey et al. [183] and adapted to our type of airship. We present an approach to identify the parameters of the physical motion model from data of flight experiments with accurate ground truth trajectories. In a subsequent step, we extend the deterministic model by a statistical identification of the sources of uncertainty and define the probabilistic motion model suitable for recursive Bayes filtering. Finally, we give an overview of the architecture of our simulator for miniature indoor airships.

2.6.1 Deterministic Physical Motion Model

Miniature airships are typically not equipped with sensors directly measuring their motion such as wheel encoders found on most ground vehicles. Instead, their motion has to be estimated based on forces and torques acting on them.

In the physical model of airships, the Newton-Euler equation of motion

$$M \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} = \mathfrak{F}_{\text{external}}(\mathbf{q}, \mathbf{v}, \boldsymbol{\omega}, \mathbf{u}) + \mathfrak{F}_{\text{fictitious}}(\mathbf{v}, \boldsymbol{\omega}) \tag{2.8}$$

couples the acceleration to the force and torque $\mathfrak{F} = \left[\mathbf{F}^T, \boldsymbol{\tau}^T\right]^T$. Here, $\mathbf{u}$ is the vector containing the applied control commands. Note that all velocities, accelerations, forces, and torques are defined with respect to the body-fixed frame of reference $\mathcal{F}_\text{b}$.

With respect to $\mathcal{F}_\text{b}$, the inertia matrix

$$\begin{aligned} M &= \begin{bmatrix} m\, I_{3\times 3} & -m\, S(\mathbf{r}_\text{g}) \\ m\, S(\mathbf{r}_\text{g}) & J \end{bmatrix} + \operatorname{diag}(k_1\, m_\text{air}, k_2\, m_\text{air}, k_2\, m_\text{air}, 0, k'\, J_{\text{air},y}, k'\, J_{\text{air},z}) \\ &=: \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad \text{with} \quad M_{ij} \in \mathbb{R}^{3\times 3} \end{aligned} \tag{2.9}$$

is composed of the mass of the airship m and its moment of inertia J, the skew symmetric matrix operator

$$S(\mathbf{r}) = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \tag{2.10}$$

and the position of the center of gravity $\mathbf{r}_\text{g}$ of the blimp. The air accompanying the airship is taken into account by Lamb's virtual mass coefficients k_1, k_2, and k' [98] where m_air

and J_{air} are the mass and the moment of inertia of the air displaced by the blimp. Here, we exploit rotation symmetry of the hull of the blimp around the x-axis of $\mathcal{F}_{\text{b}}$.

The fictitious forces and torques are caused by Coriolis and centripetal effects in the moving frame of reference $\mathcal{F}_{\text{b}}$. They can be efficiently calculated from the inertia matrix as

$$\mathfrak{F}_{\text{fictitious}}(\mathbf{v}, \boldsymbol{\omega}) = \begin{bmatrix} O_{3\times 3} & S(M_{11}\mathbf{v} + M_{12}\boldsymbol{\omega}) \\ S(M_{11}\mathbf{v} + M_{12}\boldsymbol{\omega}) & S(M_{21}\mathbf{v} + M_{22}\boldsymbol{\omega}) \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.11}$$

where $O_{3\times 3}$ is the zero block matrix [183].

The external forces and torques

$$\mathfrak{F}_{\text{external}}(\mathbf{q}, \mathbf{v}, \boldsymbol{\omega}, \mathbf{u}) = \mathfrak{F}_{\text{lg}}(\mathbf{q}) + \mathfrak{F}_{\text{D,h}}(\mathbf{v}, \boldsymbol{\omega}) + \mathfrak{F}_{\text{D,f}}(\mathbf{v}, \boldsymbol{\omega}) + \sum_{i=1}^{3} \mathfrak{F}_{\text{r},i}(\mathbf{u}) \tag{2.12}$$

consist of the lift and gravity $\mathfrak{F}_{\text{lg}}$, the drag of the hull $\mathfrak{F}_{\text{D,h}}$ and the fins $\mathfrak{F}_{\text{D,f}}$, and the propulsion of the three rotors $\mathfrak{F}_{\text{r},i}$.

The force and torque of lift and gravity can be calculated jointly as

$$\mathfrak{F}_{\text{lg}}(\mathbf{q}) = \begin{bmatrix} \mathbf{F}_{\text{l}}(\mathbf{q}) + \mathbf{F}_{\text{g}}(\mathbf{q}) \\ \mathbf{r}_{\text{g}} \times \mathbf{F}_{\text{g}}(\mathbf{q}) \end{bmatrix} \tag{2.13}$$

where $\mathbf{r}_{\text{g}}$ is the center of gravity with respect to $\mathcal{F}_{\text{b}}$ (see Figure 2.8). Since the center of lift is in the origin of $\mathcal{F}_{\text{b}}$ by definition, the torque resulting from the lifting force is zero. In particular, the lift and gravity in the body-fixed frame can be obtained by transforming the global lift $[0, 0, m_{\text{air}}\, g]^T$ and gravity $[0, 0, -m\, g]^T$ into the body-fixed frame through a rotation by the inverse of the orientation $\mathbf{q}$:

$$\begin{bmatrix} 0 \\ \mathbf{F}_{\text{l}}(\mathbf{q}) \end{bmatrix} = \mathbf{q}^{-1} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ m_{\text{air}}\, g \end{bmatrix} \odot \mathbf{q} \qquad \text{and} \qquad \begin{bmatrix} 0 \\ \mathbf{F}_{\text{g}}(\mathbf{q}) \end{bmatrix} = \mathbf{q}^{-1} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -m\, g \end{bmatrix} \odot \mathbf{q}\,.$$

Here, $\odot$ is the quaternion product, $\mathbf{q}^{-1}$ is the inverse of the unit quaternion $\mathbf{q}$ [37], m_{air} is the mass of the air displaced by the airship, and g is the gravitational constant.

In general, two different types of air drag forces can be distinguished: viscous resistance and quadratic drag. In the typical range of operation, our blimp has a Reynolds number $R_e \approx 30{,}000$ so that we can safely drop the viscous resistance term and specify the air drag by the quadratic term. We approximate the drag force and torque of the hull in an uncoupled way as

$$\mathfrak{F}_{\text{D,h}}(\mathbf{v}, \boldsymbol{\omega}) = [-D_1 v_x |v_x|, -D_2 v_y |v_y|, -D_2 v_z |v_z|, 0, -D' \omega_y |\omega_y|, -D' \omega_z |\omega_z|]^T\,. \tag{2.14}$$

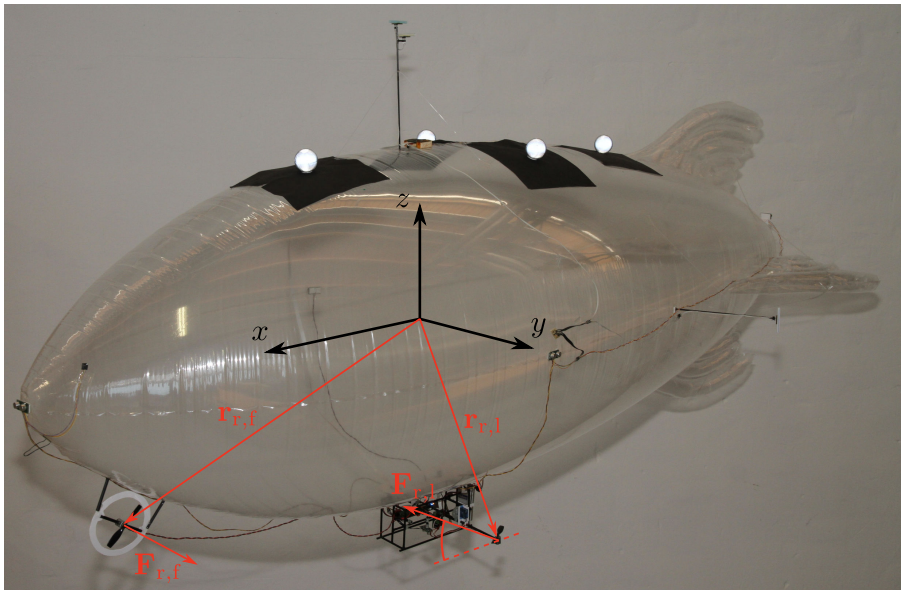


Figure 2.10: The forces of the rotors depend on their current direction. When calculating the corresponding torques, the positions the forces are acting on, i.e. the rotor positions relative to $\mathcal{F}_{\mathrm{b}}$, have to be taken into account.

Here, D_1, D_2, and D' are the drag coefficients, and we exploit rotation symmetry along the x-axis and neglect the ω_x-component, which is dominated by the drag of the fins. Analogously, the drag force of each fin acts at its center $\mathbf{r}_{\mathrm{f}}$ parallel to its normal $\mathbf{n}_{\mathrm{f}}$ and scales with the fin drag coefficient D_{f}, its area a_{f}, and the velocity component along the normal:

$$\mathbf{F}_{\mathrm{f}}(\mathbf{v}, \boldsymbol{\omega}) = -D_{\mathrm{f}}\, a_{\mathrm{f}}\ (\mathbf{n}_{\mathrm{f}} \cdot (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_{\mathrm{f}}))\ |\mathbf{n}_{\mathrm{f}} \cdot (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_{\mathrm{f}})|\ \mathbf{n}_{\mathrm{f}}\ . \tag{2.15}$$

Consequently, the forces and torques resulting from each fin are

$$\boldsymbol{\mathfrak{F}}_{\mathrm{D,f}}(\mathbf{v}, \boldsymbol{\omega}) = \begin{bmatrix} \mathbf{F}_{\mathrm{f}}(\mathbf{v}, \boldsymbol{\omega}) \\ \mathbf{r}_{\mathrm{f}} \times \mathbf{F}_{\mathrm{f}}(\mathbf{v}, \boldsymbol{\omega}) \end{bmatrix}\ . \tag{2.16}$$

The propulsion force and torque of each rotor is

$$\boldsymbol{\mathfrak{F}}_{\mathrm{r},i}(\mathbf{u}) = \begin{bmatrix} \mathbf{F}_{\mathrm{r},i}(\mathbf{u}) \\ \mathbf{r}_{\mathrm{r},i} \times \mathbf{F}_{\mathrm{r},i}(\mathbf{u}) \end{bmatrix} \tag{2.17}$$

and depends on the applied control signal where $\mathbf{F}_{\mathrm{r},i}(\mathbf{u})$ is the rotor force and $\mathbf{r}_{\mathrm{r},i}$ is the corresponding rotor position. Each rotor force depends on the applied control signal and the current direction of the rotor, which is illustrated in Figure 2.10.

Finally, we compute the overall acceleration

$$\begin{bmatrix} \mathbf{a}_t \\ \boldsymbol{\alpha}_t \end{bmatrix} = M^{-1}\ (\boldsymbol{\mathfrak{F}}_{\text{external}}(\mathbf{q}_t, \mathbf{v}_t, \boldsymbol{\omega}_t, \mathbf{u}_t) + \boldsymbol{\mathfrak{F}}_{\text{fictitious}}(\mathbf{v}_t, \boldsymbol{\omega}_t)) \tag{2.18}$$

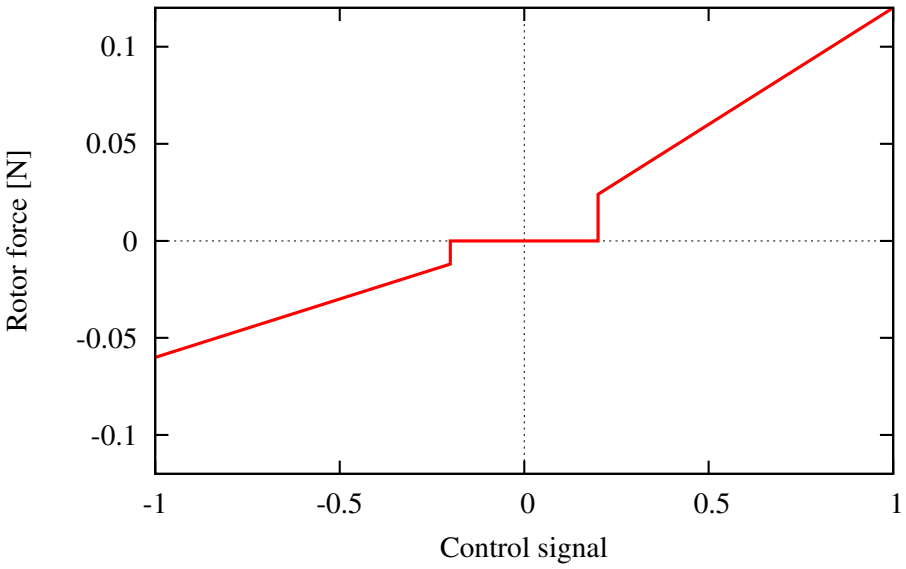


Figure 2.11: The force of a rotor depending on the applied control signal was measured with a force gauge. Due to the airfoil-shaped asymmetric propeller blades, the force in backwards rotation is substantially smaller.

and solve Equation (2.8), which is a second-order differential equation, through the numerical integration

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t \, \Delta t \tag{2.19}$$

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \boldsymbol{\alpha}_t \, \Delta t \tag{2.20}$$

$$\begin{bmatrix} 0 \\ \mathbf{p}_{t+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{p}_t \end{bmatrix} + \mathbf{q}_t \odot \begin{bmatrix} 0 \\ \mathbf{v}_t \, \Delta t + \frac{1}{2} \mathbf{a}_t \, \Delta t^2 \end{bmatrix} \odot \mathbf{q}_t^{-1} \tag{2.21}$$

$$\mathbf{q}_{t+1} = \mathbf{q}_t \odot \tilde{\mathbf{q}} \left(\boldsymbol{\omega}_t \, \Delta t + \tfrac{1}{2} \boldsymbol{\alpha}_t \, \Delta t^2 \right) \tag{2.22}$$

assuming constant acceleration during the time step of duration Δt. The function $\tilde{\mathbf{q}}(\boldsymbol{\varphi})$ represents the quaternion from the incremental rotation $\boldsymbol{\varphi} = [\varphi_x, \varphi_y, \varphi_z]^T$ around all three axes [37].

2.6.2 Parameter Learning

The physical motion model described in the previous section depends on several parameters, such as the positions of the fins and of the rotors and the mass as well as the volume of the airship. Most of the parameters can be determined in a straightforward way [118, 183]. For example, the propulsion force of a rotor depending on the applied control signal can be measured using a force gauge as shown in Figure 2.11. However, especially the air drag parameters of the hull and the fin cannot be determined without sophisticated or expensive methods such as gas simulations or measurements in a big

wind tunnel. For a useful motion model, the parameters that are difficult to identify in a straightforward way can be learned from data recorded during flight experiments with the blimp.

For our first prototype, we learned these parameters from recorded data of the blimp based on the reference trajectory $\mathbf{x}^{\star}_{1:T}$, which was acquired using the camera integrated in the gondola of the blimp [161]. To find the parameters that minimize the incremental prediction error of the motion model, we evaluated $30{,}000$ uniformly sampled values and started a gradient descent minimization routine from the ten best random parameter vectors. As all of them converged to the same parameters, we expect this to be the global minimum.

For our second and third prototype, we determined these parameters from data recorded during operation of the real airship by minimizing the difference

$$\Delta\mathfrak{a} = M^{-1}\ \left(\mathfrak{F}_{\text{external}}(\mathbf{q}^{\star}, \mathbf{v}^{\star}, \boldsymbol{\omega}^{\star}, \mathbf{u}) + \mathfrak{F}_{\text{fictitious}}(\mathbf{v}^{\star}, \boldsymbol{\omega}^{\star})\right) - \begin{bmatrix}\mathbf{a}^{\star}\\ \boldsymbol{\alpha}^{\star}\end{bmatrix} \tag{2.23}$$

between the accelerations estimated by the motion model and the ground truth accelerations $\mathbf{a}^{\star}$ and $\boldsymbol{\alpha}^{\star}$.

2.6.3 Probabilistic Motion Model

The probabilistic motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ is the core component of the prediction step of the recursive Bayes filter for probabilistic state estimation. We extend the deterministic model described above by a statistical identification of the sources of uncertainty and define the probabilistic motion model suitable for recursive Bayes filtering.

The uncertainty of the motion model described above has basically three sources. First, the applied physical model is only an approximation, and second, the estimated parameters are not guaranteed to be the true parameters due to imperfection of the reference trajectory and the minimization routine. Third, the most important errors are statistical errors such as imperfect motor responses, wind, or numerical errors. In our probabilistic localization system, we combine all three sources of error and approximate them as a white (zero-mean) Gaussian noise of the acceleration estimated by the motion model. From a sequence of differences defined in Equation (2.23) we can estimate the covariance of the acceleration errors calculated by the learned model. This covariance implicitly defines the probabilistic model needed for sampling in the prediction step of the particle filter. In particular, we sample the acceleration errors from the corresponding Gaussian and obtain samples of subsequent states by error propagation through the numerical integration of Equations (2.19) to (2.22).

2.6.4 The Blimp Simulator

Especially for the evaluation of approaches to autonomous navigation, a simulator can be a great advantage. It enables the evaluation under controlled conditions and can save the time of the elaborate experimental setups of the robotic blimp. Furthermore, a training of pilot candidates in the blimp simulator has proven to substantially improve the safety when these pilots manually control the real blimp through complex environments with narrow passages in later experiments.

We implemented a blimp simulation module in our modular software framework. This module simulates the behavior of the blimp as well as that of the sensors the blimp is equipped with. To obtain a realistic simulation, our simulation module samples from the probabilistic models of both, the motion of the blimp (described in the previous section) and the sensors (described in the following chapters). As an additional option, our simulation module allows to adjust the amount of noise when sampling from the probabilistic models.

Consequently, the simulation system needs the static map of the environment for simulation and processes incoming blimp control commands. As an output, the simulator provides the state of the blimp in terms of poses, velocities, and accelerations as well as the measurements of the sensors enabled for simulation.

2.7 Related Work

In the past, several authors have considered autonomous aerial blimps. For example, Kantor et al. [80], Elfes et al. [42], Hada et al. [65], Saiki et al. [149] and Hygounenc et al. [73] developed airships with several kilograms of payload and utilized them for surveillance, data collection, or rescue mission coordination tasks. The relatively high payload of these systems allows the blimp to carry more powerful sensors and also facilitates more extensive on-board computations than our miniature blimp system.

Additionally, there has been work on navigation with small-scale blimps that utilize cameras for localization or even SLAM [4, 88, 161]. While cameras provide rich information, the processing of the images can typically not be carried out on the embedded computers installed on such miniature airships.

Kirchner and Furukawa [86] present a localization system for indoor UAVs, which utilizes an infrared emitter on the vehicle and three external infrared sensors to localize the robot via triangulation. Although this approach does not have high computational demands, it requires external devices that perceive the infrared signals.

In addition, many small-scale blimps have been designed for remote controlled operation or rather simple control tasks such as corridor or line following [57]. Consequently, these airships are usually equipped with severely limited sensing and navigation capa-

bilities. For example, Zufferey et al. [183] developed a miniature blimp, which was controlled on a circular trajectory based on the image of a camera with only a single horizontal line of pixels. Ishida [74] developed a blimp for mapping gas distributions during random flights through the environment. Several approaches were proposed for blimp systems equipped with altitude stabilization and collision avoidance. Al-Jarrah and Roth [2] controlled their blimp using an on-board microcontroller processing sonar measurements in fuzzy logic without considering their measurement uncertainty. Bermúdez i Badia et al. [13] applied a wireless camera and operated the blimp from a base station using neural network controllers.

In contrast to all small-scale blimp concepts described above, our blimp is equipped with various sensors and a full-fledged computer for autonomous navigation in complex environments, and its sensor setup can be easily adapted to different navigation tasks.

Part I

Localization

Chapter 3

Recursive State Estimation

Estimating the state of a system from noisy motion and sensor data is a fundamental problem in the domain of autonomous robots. In this chapter, we give an overview of the concept of the Bayes filter, which is a recursive algorithm for probabilistic sensor data fusion. We discuss several implementations of the Bayes filter with regard to the localization of miniature airships. One exceptionally powerful implementation is the particle filter for which we give a detailed description and implementation details in the context of miniature airship localization.

• • • • •

In this chapter, we describe fundamental techniques for estimating the state of the world when a robot interacts with its environment. The state of the world can include that of the robot, for example, its position or configuration, as well as that of its environment, e.g. the location of objects surrounding the robot. We distinguish between two possible modes of interaction of a robot with its environment:

- **Action**. When acting, the robot executes random or specific *control* commands to alter its state or that of the environment. Typical actions include accelerating the wheels of a mobile robot and manipulating the environment with the gripper of a robotic arm.

- **Perception**. When perceiving, the robot retrieves information about its own state or the state of the environment by the use of its sensors. The perception results in a *measurement*, also called *observation*, which can, for example, be the image of a camera or the distance measurement of a sonar sensor.

Throughout this part of the thesis, we consider the *localization* of a mobile robot, which means to estimate its state including its position and orientation. In the localization scenario, we assume that the state of the robot is dynamic, i.e. possibly changing over

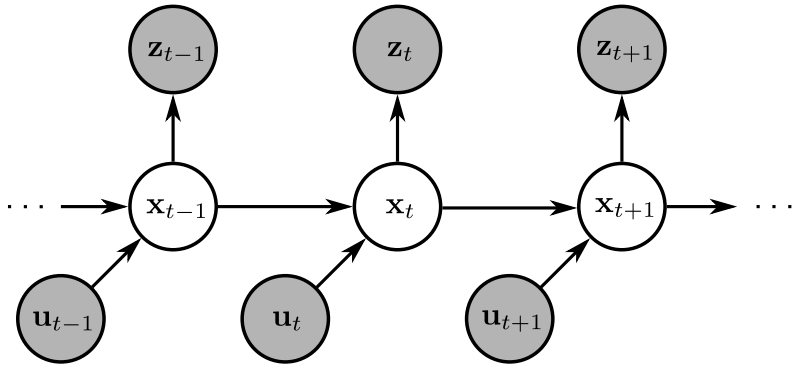


Figure 3.1: The dynamic Bayes network for the localization of a mobile robot under the complete state (Markov) assumption. It characterizes the evolution of the hidden state variables $\mathbf{x}$ given the observable control variables $\mathbf{u}$ and measurement variables $\mathbf{z}$.

time, and the state of the environment is static and known a-priori. We represent the known environment by a *map*, whose model usually depends on the type of the applied sensor devices.

In this chapter, we describe our approach to online state estimation for autonomous navigation, which requires a computationally efficient incremental integration of control commands and sensor measurements as soon as they are available during operation of the robot. Although optimization-based maximum likelihood estimators have become popular recently [78, 94, 162], they usually require a time-consuming optimization on a large fraction of all available data in certain intervals or even every time new information from actuators or sensors comes in. In contrast to this, we describe in the following a recursive filtering approach that enables an efficient incremental online integration of control and measurement data and therefore is suitable for online localization as a part of an autonomous navigation system.

3.1 The Bayes Filter

For solving the online localization problem, we follow the probabilistic approach and apply the recursive Bayes filtering scheme, which is summarized in the following according to Thrun et al. [169].

Throughout most of this thesis, we assume that the dynamic process is discretized into discrete time steps and that the robot executes a control $\mathbf{u}_t$ and takes a measurement $\mathbf{z}_t$ at time steps $t \in \mathbb{N}$. The controls and the measurements are possibly multidimensional vectors, e.g. the acceleration commands for each motor or the range values of all individual beams of a laser range finder. The state $\mathbf{x}_t$ at time t can, for example, include the pose and velocity of the robot. Furthermore, we use the notation $\mathbf{x}_{1:t}$ to denote the (time) sequence $\mathbf{x}_1, \ldots, \mathbf{x}_t$ of variables.

In addition, we assume a *complete state*, which is also known as the *Markov assumption*, throughout this thesis. This means that the knowledge of the current state contains all information that is currently available about the future of the system. In particular, the additional knowledge of past states or current or past measurements adds no extra information about the future. Under the complete state assumption, the evolution and causal dependencies of the random variables described above can be expressed by the dynamic Bayes network shown in Figure 3.1. Since the state variables are not observable, this in particular is a hidden Markov model, which is a special case of a dynamic Bayes network [147].

The key concept of the Bayes filter algorithm is to explicitly account for the uncertainty in the process by considering probability distributions instead of maximum likelihood estimates. Although the Bayes filter algorithm can deal with discrete and continuous probability distributions, the notation in this thesis assumes continuous probability distributions, as robots usually operate in continuous state spaces. Let Y be a random variable. Then, $p(Y)$ is the probability distribution of Y. Furthermore, $p(Y = y)$ denotes the value of the probability density function of the random variable Y at the specific value y. For a detailed introduction to probability theory, the reader is referred to Bishop [16]. In the following, we omit the random variable for brevity whenever possible and use the common short notation $p(y)$ instead of $p(Y = y)$.

In the Bayes filter algorithm, we recursively estimate the a-posteriori probability distribution, also called posterior distribution,

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.1}$$

of the state $\mathbf{x}_t$ conditioned on all sensor data $\mathbf{z}_{1:t}$ and control commands $\mathbf{u}_{1:t}$ up to time t. Using Bayes rule [169] and the rules of d-separation [16] on the dynamic Bayes network (Figure 3.1), one can derive the recursive update step [169]

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \, p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \, \mathrm{d}\mathbf{x}_{t-1} \; . \tag{3.2}$$

Here, η_t is a normalizing constant that ensures that $\int p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, \mathrm{d}\mathbf{x}_t = 1$. Furthermore, the algorithm requires the belief $p(\mathbf{x}_0)$ as an initial condition, which is usually given as a Gaussian or a point mass distribution in practice.

The resulting Bayes filter algorithm is shown in Algorithm 4. Basically, it has two separate steps:

- The **prediction** step, also called motion update, is implemented in line 1. It updates the belief of the filter based on the so-called motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$. The motion model specifies the state transition probability given the applied control command.

Algorithm 4 BAYESFILTERUPDATE

Input: The belief bel_{t-1} at time $t-1$ with $bel_{t-1}(\mathbf{x}_{t-1}) = p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})\ \forall \mathbf{x}_{t-1}$, the current measurement $\mathbf{z}_t$, and the applied control command $\mathbf{u}_t$.

Output: The belief bel_t at time t with $bel_t(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\ \forall \mathbf{x}_t$.

1: $\forall \mathbf{x}_t :\ \overline{bel}_t(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)\ bel_{t-1}(\mathbf{x}_{t-1})\ \mathrm{d}\mathbf{x}_{t-1}$
2: $\eta_t = \left(\int p(\mathbf{z}_t \mid \mathbf{x}_t)\ \overline{bel}_t(\mathbf{x}_t)\ \mathrm{d}\mathbf{x}_t\right)^{-1}$
3: $\forall \mathbf{x}_t :\ bel_t(\mathbf{x}_t) = \eta_t\ p(\mathbf{z}_t \mid \mathbf{x}_t)\ \overline{bel}_t(\mathbf{x}_t)$
4: **return** bel_t

- The **correction** step, known as measurement update, is implemented in line 2 and 3. The probability distribution of the belief is corrected by the sensor model $p(\mathbf{z}_t \mid \mathbf{x}_t)$. The sensor model specifies the likelihood of the measurement $\mathbf{z}_t$ given the corresponding state $\mathbf{x}_t$. To obtain a probability distribution, the result is normalized.

For practical applications with continuous state spaces, the rather abstract concept of probability distributions in the Bayes filter algorithm has to be approximated by a representation that can be implemented efficiently. In the following, we discuss popular implementations of the Bayes filter algorithm. We present their complexity depending on the dimensionality n of the state space and the dimensionality k of the measurement space. Furthermore, we discuss their suitability for the localization of miniature indoor airships equipped with tiny sonar and air flow sensors as well as an IMU.

Kalman filters [79] (and their counterparts, the information filters [6]) efficiently approximate each probability distribution appearing in the Bayes filter by a normal distribution. The normal distribution is represented in a parametric way by its mean and its covariance. Additionally, the Kalman filter requires linear system dynamics, i.e. a linear measurement and motion function. Under these conditions, the Kalman filter is the optimal estimator and each update step has (approximately) complexity $O(k^{2.4}+n^2)$. To relax the prerequisite of linearity, there are several extensions. The *extended Kalman filter (EKF)* [6] linearizes the system using a first-order Taylor approximation, and the *unscented Kalman filter (UKF)* [75] as well as the *square-root UKF* [175] propagate the Gaussians using deterministic sigma-points. Although they are robust to global nonlinearities in the process, they suffer from substantial nonlinearities in the measurement and the motion function in the region of high probability of the normal distribution. These nonlinearities are often an issue if only little sensor information is available and the uncertainty of the estimator and therefore the region of high probability is large [169].

Discrete filters work on discrete state spaces, which, in the context of robotics, are usually obtained by a discretization of the continuous state space into a grid [21, 169]. While these filters are precise as long as the resolution of the discretization is sufficiently

high, their drawback is the high memory consumption of the grid and the computationally demanding update of all grid cells in each filter step. For a fixed grid resolution and a fixed size of the bounding box of the state space, the number of grid cells scales exponentially in n and therefore the complexity is exponential in n.

Particle filters [40] approximate the posterior distribution by a set of weighted random samples, called particles. This allows to represent arbitrary distributions, while the accuracy of the approximation increases with an increasing number of samples. The complexity of the particle filter algorithm is in $O(nN)$ where N is the number of particles. For a fixed particle density, in general the number of particles scales exponentially with the dimensionality of the state so that the overall complexity is exponential in n [165]. However, since the resampling step ensures that only the area of high probability is represented densely by particles, for tracking scenarios, the particle filter has proven to be substantially more efficient in practice than the discrete filters [35].

Besides the three popular Bayes filter implementations described above, there are a couple of combinations and extensions of them. For example, the *Gaussian particle filter* [91] is similar to the sigma-point Kalman filters. It relies on the Gaussian approximation of the posterior distribution but is more tolerant to nonlinear process dynamics, as it is based on a possibly large number of random samples (like the particle filter) instead of the deterministic sigma-points. The only advantage of this approach over particle filters is that it does not require the resampling step. However, this advantage gives no improvement in terms of complexity of the algorithm. Other methods that allow to represent a non-Gaussian posterior distribution are, for example, the *multi-hypothesis Kalman filters* [136] using the sum of Gaussians approximation [3] and the recently introduced *antiparticle filter* [48, 49].

For the localization of a miniature indoor airship equipped with tiny sonar and air flow sensors as well as an IMU we choose the particle filter approach. The particle filter has been successfully applied to localize mobile robots in the past [35, 60, 70, 95, 159, 165, 168] and can cope with the challenging properties of an airship, which are a nonlinear motion model with high motion uncertainty combined with sparse and ambiguous position information from tiny sonar sensors with a huge opening angle.

3.2 The Particle Filter

The particle filter [40, 169] is a nonparametric implementation of the Bayes filter for recursive state estimation. It approximates the probability distribution of the state $\mathbf{x}_t$ by a set of weighted particles

$$\mathcal{M}_t = \left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right) \right\}_{i \in [1,N]} \tag{3.3}$$

where each particle consists of a state hypothesis $\mathbf{x}_t^{[i]}$ weighted by a so-called importance weight $w_t^{[i]}$. The importance weight of each particle is a positive value and the sum of the weights over all particles is 1.

For recursive state estimation of a dynamic system, the most common particle filtering algorithm is the sampling importance resampling (SIR) filter. In general, the actual probability distribution $p(\mathbf{x}_t)$ to be estimated and represented by the filter, also called the target distribution, is not suitable for sampling. The SIR filter accounts for this fact by sampling from a proposal distribution π with $p(\mathbf{x}_t) > 0 \Rightarrow \pi(\mathbf{x}_t) > 0$ for all possible states. The recursive belief update of the SIR filter is performed according to the following three steps [35, 40, 58]:

1. **Sampling**: In the *prediction step*, for each particle of $\mathcal{M}_{t-1}$, a successor state is drawn from the proposal distribution π. The resulting set of propagated particles is $\mathcal{M}_t$ where the weight of each particle in $\mathcal{M}_t$ is equal to the weight of the corresponding particle in $\mathcal{M}_{t-1}$.

2. **Importance Weighting**: In the *correction step*, the importance weight of each particle is updated according to the importance sampling principle, which takes into account the difference between the target distribution p and the proposal distribution π:
$$w_t^{[i]} = \frac{p(\mathbf{x}_t^{[i]})}{\pi(\mathbf{x}_t^{[i]})} \ . \tag{3.4}$$

3. **Resampling**: In the *resampling step*, a new generation of particles is drawn from $\mathcal{M}_t$ (with replacement) so that each sample in $\mathcal{M}_t$ is selected with a probability that is proportional to its weight. This step accounts for having only a finite number of particles to approximate the continuous distribution. After resampling, the equal weight $\frac{1}{N}$ is assigned to all particles.

In the following, we describe the implementation of the particle filter in the context of mobile robot localization. In this context, the particle filter is also called Monte Carlo localization, which was first introduced by Dellaert et al. [35]. In mobile robot localization, we recursively estimate the full trajectory

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.5}$$

of the robot as the target distribution in the particle filter. Hence, each particle contains a hypothesis of a sequence of states $\mathbf{x}_{1:t}^{[i]} = \mathbf{x}_1^{[i]}, \ldots, \mathbf{x}_t^{[i]}$. To finally obtain the posterior distribution of the current state from the state of the particle filter, we marginalize out $\mathbf{x}_{1:t-1}$, which can be straightforwardly accomplished by ignoring the corresponding part of the state hypothesis of each sample of the particle filter.

For the implementation of the particle filtering scheme described above, the optimal proposal distribution would be the target distribution $p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$. However, one can usually not directly sample from this distribution. Instead, we factorize the target distribution into

$$
\begin{aligned}
&p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \\
&\qquad = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \, p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) && (3.6)\\
&\qquad = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t) \, p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) && (3.7)\\
&\qquad = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t) \, p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \, p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) && (3.8)\\
&\qquad = \eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t) \, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \, p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) && (3.9)
\end{aligned}
$$

where η_t is a normalizing constant. In this factorization, we apply Bayes rule in Equation (3.6) and exploit the rules of d-separation in the dynamic Bayes network (Figure 3.1) under the Markov assumption in Equation (3.7) and Equation (3.9). Furthermore, in Equation (3.8), we apply a factorization based on the rule of conditional probability. In Equation (3.9), $p(\mathbf{z}_t \mid \mathbf{x}_t)$ is the sensor model, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ is the motion model, and $p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$ is the recursive term. As proposed by Dellaert et al. [35], we use the probabilistic motion model together with the recursive term as the proposal distribution, i.e.

$$
\pi(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \, p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \, . \tag{3.10}
$$

As a consequence of this choice of the proposal distribution, the weight of the i-th particle in the correction step of the particle filter is

$$
\begin{aligned}
w_t^{[i]} &= \frac{p(\mathbf{x}_{1:t}^{[i]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{\pi(\mathbf{x}_{1:t}^{[i]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})} && (3.11)\\
&= \frac{\eta_t \, p(\mathbf{z}_t \mid \mathbf{x}_t^{[i]}) \, p(\mathbf{x}_t^{[i]} \mid \mathbf{x}_{t-1}^{[i]}, \mathbf{u}_t) \, p(\mathbf{x}_{1:t-1}^{[i]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})}{p(\mathbf{x}_t^{[i]} \mid \mathbf{x}_{t-1}^{[i]}, \mathbf{u}_t) \, p(\mathbf{x}_{1:t-1}^{[i]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})} && (3.12)\\
&\propto p(\mathbf{z}_t \mid \mathbf{x}_t^{[i]}) \, . && (3.13)
\end{aligned}
$$

Thus, in the correction step, we weight each particle by the measurement likelihood specified by the probabilistic sensor model and afterwards account for η_t by normalizing the particles so that their weights sum up to 1.

In a nutshell, the resampling step typically replaces particles with a low weight with particles that have a higher weight. On the one hand, this ensures that the set of particles represents the target distribution and that its region of high probability is densely covered by particles. On the other hand, the standard implementation of independent resampling has complexity $O(N \log N)$ and additionally can problematically reduce the diversity of the particles, which is known as the particle deprivation problem [169]. To cope with

these challenges, we apply an efficient implementation of the particle filter: First, we use low-variance resampling [169], also known as systematic resampling. Its complexity is in $O(N)$ and it furthermore reduces the particle deprivation problem through its systematic selection of particles. Second, we apply selective resampling, which means that we omit the resampling step until the variance in the weights of the particles has reached a specific threshold. In particular, we omit the resampling step until the effective number of particles [106]

$$N_{\text{eff}} = \left(\sum_{i=1}^{N} \left(w^{[i]} \right)^2 \right)^{-1} , \tag{3.14}$$

which is a measure of the equality of the importance weights, drops below $N/2$ where N is the number of particles. A detailed description of these techniques as well as of the particle filter in general and in particular for mobile robot localization is provided by Doucet et al. [40] and Thrun et al. [169].

3.3 Particle Filter Localization for Indoor Airships

Throughout this part of the thesis, we consider the localization of an indoor airship using a particle filter. In the prediction step of the particle filter, we sample from the probabilistic motion model described in Section 2.6.3. Thereby, we first sample the six-dimensional acceleration from a Gaussian. The mean of this Gaussian is determined from the physical motion model based on the state and the controls (see Equation (2.18)). The noisy acceleration value is then propagated through numerical integration of the Newton-Euler equation of motion (see Equations (2.19) to (2.22)) to obtain a random sample of the probabilistic motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$.

For pose and velocity measurements, our blimp is equipped with a couple of sonar and air flow sensors as well as with an IMU. The measurements of the individual sensors are considered together as one high-dimensional measurement vector $\mathbf{z}$ in the particle filter described above. However, in practice, the individual sensors operate at different frequencies so that their measurements cannot be considered as being available at each discrete time step t.

Fortunately, under the Markov assumption, the measurements of the individual sensors are conditionally independent given the state of the system [169]. Hence, we can factorize the measurement likelihood

$$p(\mathbf{z} \mid \mathbf{x}) = \left(\prod_{i=1}^{K} p(z_{\text{S},i} \mid \mathbf{x}) \right) \left(\prod_{j=1}^{L} p(z_{\text{F},j} \mid \mathbf{x}) \right) p(\mathbf{z}_{\text{I}} \mid \mathbf{x}) \, . \tag{3.15}$$

into the product of the measurement likelihoods of the individual sensors. For our miniature blimp, these are the scalar distance measurements $z_{\text{S},i}$ of the K sonar sensors, the

scalar measurement values $z_{\mathrm{F},j}$ of the L air flow sensors, and the orientation estimate $\mathbf{z}_{\mathrm{I}}$ of the IMU. Due to their conditional independence given the state of the system, the measurements of the individual sensors can be integrated in separate weighting steps into the belief of the particle filter as shown in Figure 3.2.

Throughout the remaining chapters of this part, we describe the probabilistic models for the sensors our airship is equipped with. Furthermore, we propose the simultaneous localization and estimation of the parameters of the motion model and introduce our novel air flow odometry motion model for computing accurate and efficient proposal distributions for the particle filter.

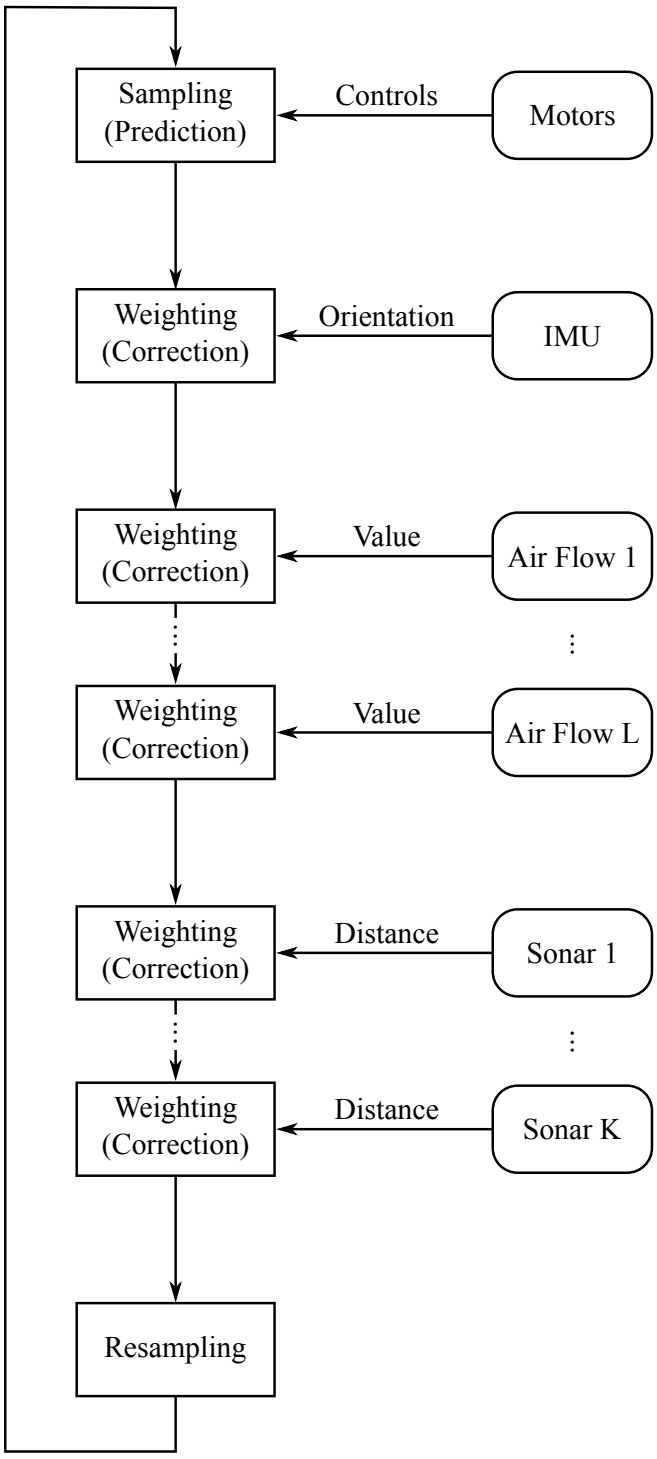


Figure 3.2: The process of the particle filter localization for a mobile robot equipped with an IMU, L air flow sensors, and K sonar sensors. The left side shows the individual processing steps of the set of particles. The right side shows the actuator and sensor devices providing control and measurement data.

Chapter 4

Sonar Sensor Model

The major constraints of miniature airships stem from their limited payload, which introduces substantial constraints on their perceptional capabilities. In this chapter, we consider the problem of localizing a miniature blimp with lightweight ultrasound sensors. Since the opening angle of the sound cone emitted by a sonar sensor depends on the diameter of the membrane, small-sized sonar devices introduce a high uncertainty regarding which object has been perceived. We present a novel sensor model for ultrasound sensors with large opening angles that enables an autonomous blimp to robustly localize itself in a known environment using Monte Carlo localization. As we demonstrate in experiments with a real blimp, our novel sensor model outperforms a popular sensor model that has in the past been shown to work reliably on wheeled platforms.

• • • • •

Sonar sensors have been a popular sensing device in mobile robot navigation, as they are lightweight and cheap compared to laser range finders. In nature, for example, bats navigate based on ultrasonic echolocation whereby they localize, avoid obstacles, and hunt prey with great success. Their ultrasonic sensing technique is far more sophisticated than the basic time-of-flight distance measurement approach sonar sensors in robotics rely on. Although these sophisticated techniques would be desirable in robotic applications, our work shows that even tiny time-of-flight sonar sensors can be successfully applied for perception in autonomous mobile robot navigation if their challenging properties are adequately taken into account.

In probabilistic state estimation, a crucial aspect is the design of the so-called probabilistic sensor model $p(\mathbf{z} \mid \mathbf{x})$, which defines the likelihood of a measurement $\mathbf{z}$ given the state $\mathbf{x}$ of the system. In case of localization with sonar sensors, the measurement $r \in \mathbb{R}$ is a scalar distance value and its likelihood depends on the pose of the sensor and the

environment. We assume that the state $\mathbf{x}$ of the system contains the pose of the airship at which the sensor is rigidly mounted or the pose of the sonar sensor itself and also tacitly includes the map of the environment.

The probabilistic sensor model needs to be specified properly to provide accurate state estimates and to avoid divergence of the filter. In this context, the miniature Devantech SRF10 ultrasound sensors our blimp is equipped with pose a challenging problem. Their wide opening angle introduces a high uncertainty, which needs to be correctly modeled. We present a novel sensor model for ultrasound sensors that has several desirable features compared to previously developed models. It better reflects the physical properties of ultrasound sensors and it is especially suited to deal with the wide opening angles of small-scale ultrasound sensors. We experimentally evaluate our model on a miniature blimp system in an indoor navigation task. In practical experiments, we demonstrate that our model outperforms an alternative and popular sonar sensor model.

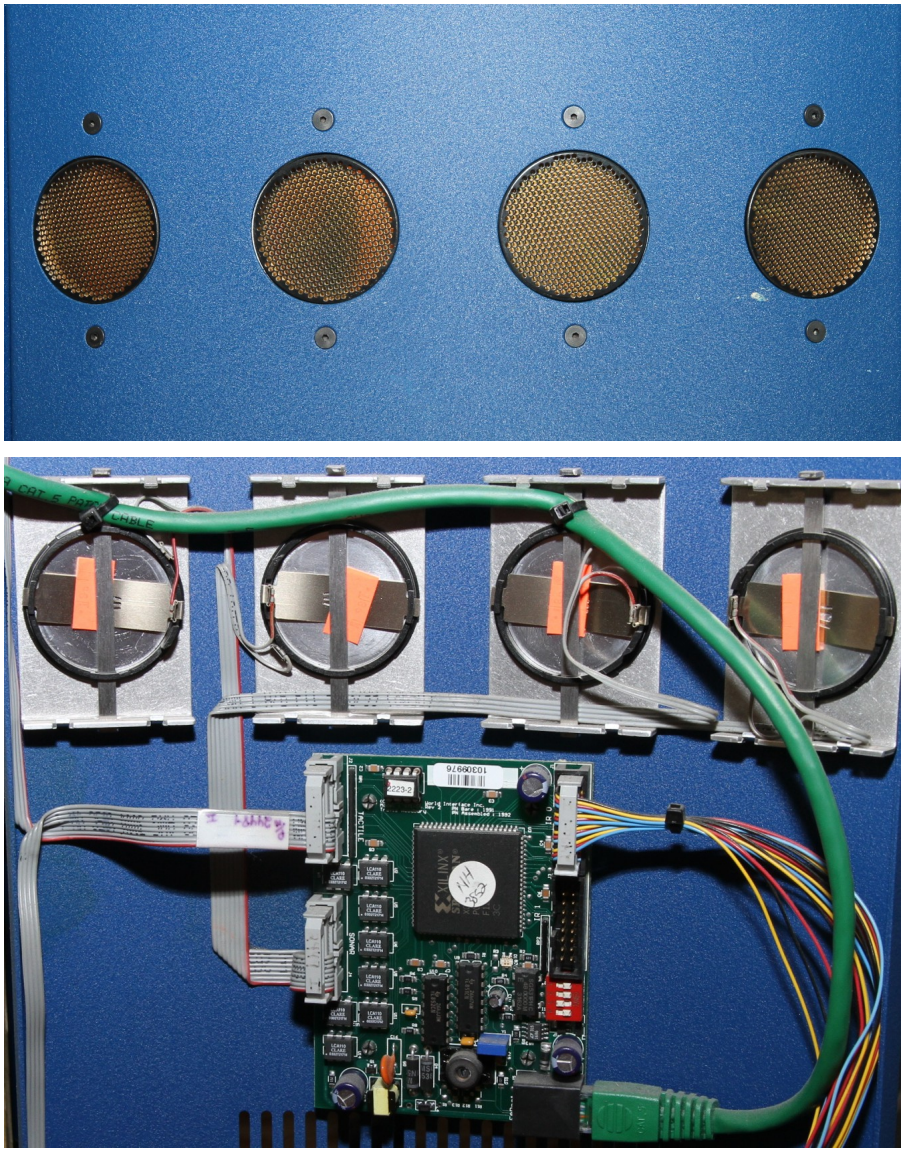
In the following, we first briefly discuss the popular ray-casting sensor model. We will then introduce our novel cone sensor model, which explicitly models the characteristics of small-sized sonar sensors with large opening angles. While the novel cone sensor model and its evaluation was published in the diploma thesis [118], the comparison to the standard ray-casting sensor model and the extended experimental evaluation is a contribution of this thesis.

4.1 The Ray-casting Model

Thrun et al. [169] and Choset et al. [28] describe an approach to model the measurement likelihood for sonar or laser range finders, which in the past has successfully been applied to robustly localize wheeled platforms equipped with standard Polaroid ultrasound sensors (see Figure 4.1) with an opening angle of approximately 15° [64].

Their approach models $p(r \mid d(\mathbf{x}))$ based on the distance $d(\mathbf{x})$ to the closest object along the acoustical or optical axis of the sensor. To determine this likelihood, they perform a ray-casting operation in the map to determine $d(\mathbf{x})$ and calculate $p(r \mid d(\mathbf{x}))$ based on a mixture of four different distributions to capture the noise and error characteristics of range sensors. The major component of this model is a Gaussian $\mathcal{N}(r; d(\mathbf{x}), \sigma^2)$ that characterizes the distribution of measurements in situations in which the closest object along the acoustical or optical axis of the sensor is detected. Additionally, this model includes an exponential distribution $\lambda e^{-\lambda r}$ to properly model measurements reflected by objects not contained in the map. Furthermore, it utilizes a uniform distribution to model random measurements caused, for example by sensor failures. Finally, maximum range measurements are modeled using a constant probability. The measurement likelihood is composed as a weighted average of these four different distributions.

While this model enables a highly accurate localization given typical ultrasound sen-



(a)

(b)

Figure 4.1: Four standard Polaroid 6500 sonar sensors (a) in front and back view. Their membrane has a diameter of 38 mm and the controlling board has a size of 77 mm $\times$ 102 mm. Compared to that, the tiny Devantech SRF10 sonar sensor (b) our blimp is equipped with has a membrane diameter of 8.5 mm and the controlling board has a size of 32 mm $\times$ 15 mm.

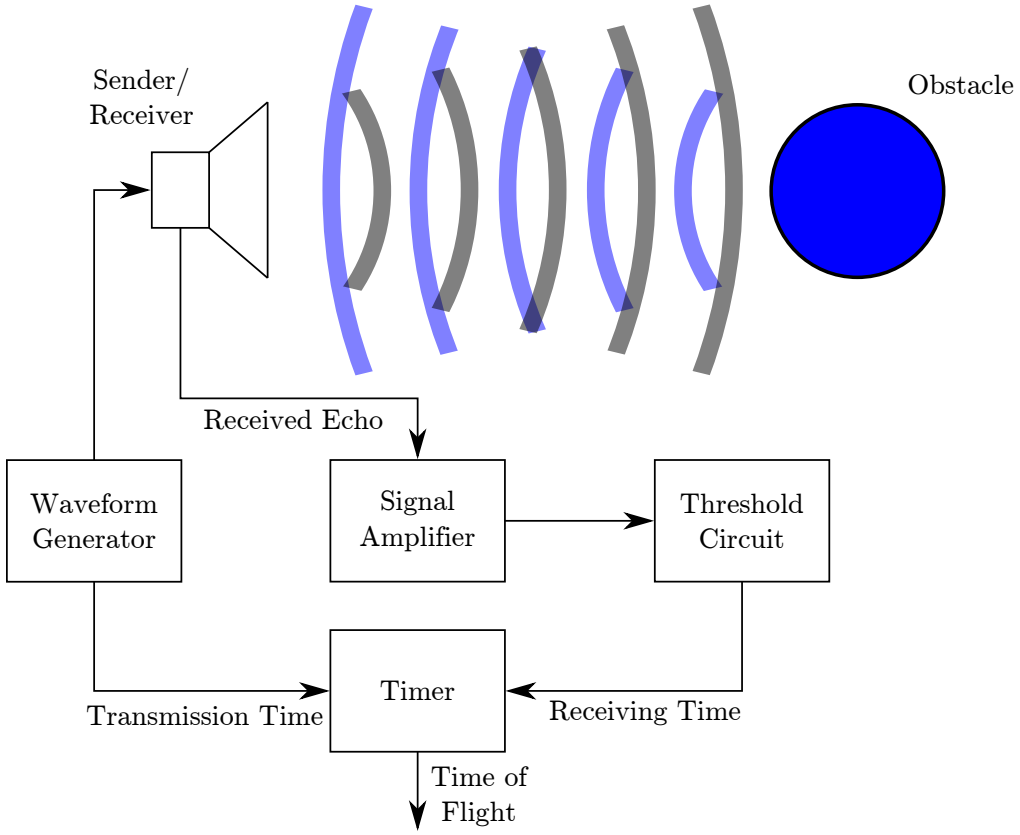


Figure 4.2: The sonar distance measurement procedure according to Leonard and Durrant-Whyte [104].

sors or laser range finders, it yields suboptimal results for small sonar sensors having a large opening angle. The reason is that for wide opening angles, it is no longer sufficient to calculate the measurement likelihood solely based on the distance to the closest object along the acoustical or optical axis of the sensor. In this work, we especially cope with this problem and propose a model that explicitly considers the opening angle $\theta = 1.22\,\frac{\lambda}{d_{\mathrm{m}}}$, which depends on the wavelength λ of the signal and the diameter d_{m} of the membrane (see Brown [17]). Accordingly, the closest object in the entire corresponding cone is considered, which better reflects the wide opening angle.

4.2 The Cone Model

In our novel approach, we seek to model the observation likelihood by systematically considering the underlying physics of ultrasound sensors, which is illustrated in Figure 4.2. A measurement starts with the generation and transmission of an ultrasound signal. The signal propagates spherically through the space and, after it got reflected by

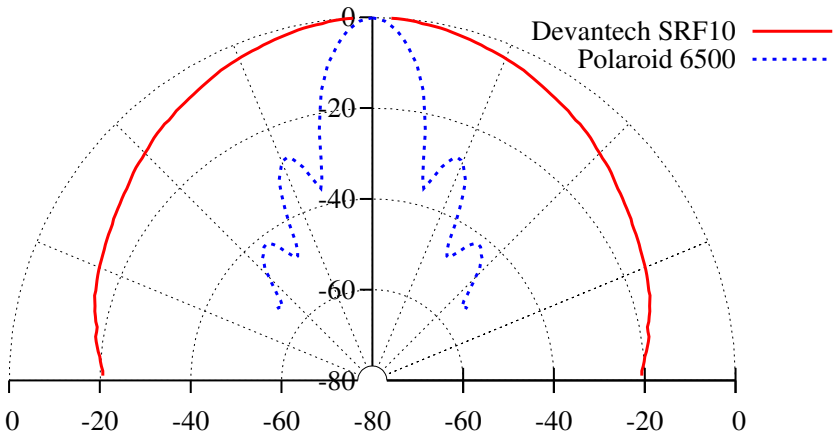


Figure 4.3: The intensity pattern of the Devantech SRF10 miniature sonar sensor compared to the one of the popular Polaroid 6500 sensor [23, 36]. Units are decibel normalized to the maximum intensity.

objects in the environment, the corresponding echo is typically detected by the receiver unit of the sensor. Since the received signal is usually much weaker than the transmitted signal, it gets amplified by a predefined amplification factor f_a. As soon as the received and amplified signal exceeds a given threshold, the measurement procedure is terminated and the distance $r = \frac{c\,\Delta t}{2}$ is calculated based on the time of flight Δt and the constant velocity c of sound in air.

For very small transmitters with a diameter in the same order of magnitude as the wavelength, the signal is hardly focused as shown in Figure 4.3. Thus, it can be considered as a growing hemisphere, which has lower intensity at its boundary area. Consequently, a small sensor tends to detect large, well reflecting objects such as walls that are perpendicular to the heading of the sensor. At the same time, small objects that are further away from the sensor cannot be detected even if they are in the center of the cone. In our approach, we model this behavior by considering the detection of objects depending on their size, angle, distance, and the applied amplification factor. In particular, we calculate a probability distribution of triggering a measurement by modeling the received signal over the elapsed time Δt.

To consider the propagation of the signal in the environment, we define a spherical coordinate system with its origin at the position of the sensor (Figure 4.4). The emitted signal intensity (power per area) I depends on the zenith angle θ, which is depicted in Figure 4.3. Due to the symmetry of ultrasonic membranes, it does not depend on the azimuth angle ϕ. Hence, the whole signal power can be written as

$$P_0 = \int I(\theta)\,\mathrm{d}\Omega \tag{4.1}$$

by integration over the hemisphere in front of the sensor where Ω is the dihedral angle.

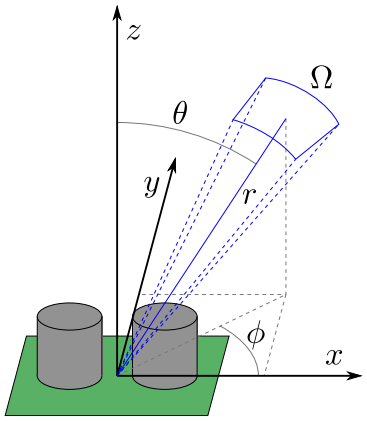


Figure 4.4: The spherical coordinate system used for modeling the sensor behavior. An object is seen by the sensor in distance r, azimuth angle ϕ, and zenith angle θ. In this way, the dihedral angle Ω is covered.

This signal power is damped by a factor $D(r)$ and the intensity is scaled by $\frac{1}{r^2}$ with increasing distance r since the surface area of the hemisphere scales with r^2. In contrast to Moravec [115], we explicitly model these two effects physically.

To determine the objects that potentially reflect the propagating signal, we assume that an occupancy map of the environment specifying the obstacles and the free space is given. We determine the set of relevant objects by a discrete set of ray-casting operations according to a fixed angular resolution so that the entire visible hemisphere is covered. Let H_i be an object that is seen by the sensor in distance r_i and zenith angle θ_i and that corresponds to the dihedral angle Ω_i. Then, the incident signal power is

$$P_i = I(\theta_i)\, D(r_i)\, \Omega_i \,. \tag{4.2}$$

A proportion $P_{R,i} = \kappa_i\, P_i$ of this signal power is reflected back to the sensor. The reflection proportion $\kappa_i \in [0, 1]$ depends on the relative angle of incidence of the signal and the reflection properties of the object. Unfortunately, the latter properties are hard to obtain and would also further increase the storage requirements of the map. Since diffuse reflection just occurs on surfaces that have a roughness in the order of the magnitude of the wavelength, typical uncluttered indoor environments mainly produce specular reflections. Additionally, diffuse reflected signals again propagate on a hemisphere, which causes them to be very weak. Therefore, we only consider specular reflections, whereby the signal power that is reflected towards the receiver can be estimated according to

$$\begin{aligned} p_i(P_{R,i}) &= \alpha\, p(P_{R,i} \mid \text{reflection towards sensor}) \\ &\quad + (1-\alpha)\, p(P_{R,i} \mid \text{reflection to other direction}) \\ &\approx \alpha\, \delta(P_{R,i} - P_i) + (1-\alpha)\, \delta(P_{R,i}) \end{aligned} \tag{4.3}$$

for some $\alpha \in [0,1]$ using the Dirac delta δ. Since there is typically no information about correlations between the reflection properties of objects, we assume them to be independent. Furthermore, we do not consider multiple reflections or interference.

At time Δt, the sensor starts to receive the reflected signal of objects at the distance $r = \frac{c\,\Delta t}{2}$. The emitted ultrasound signal has the length l_s, which usually is a couple of wavelengths. Therefore, at this time, the sensor still receives the reflected signal of objects in distances between $r - \frac{l_\mathrm{s}}{2}$ and r. In the following, we denote the set of objects that reflect a signal that could contribute to trigger the measurement of distance r by $H(r) = \left\{H_i \mid r_i \in [r - \frac{l_\mathrm{s}}{2}, r]\right\}$. Consequently, the total received power corresponding to the distance r can be written as the sum over the reflected powers of all objects of $H(r)$

$$P_R(r, \mathbf{x}) = \sum_{H_i \in H(r)} P_{R,i} \tag{4.4}$$

where each $P_{R,i}$ is distributed according to Equation (4.3).

Furthermore, the probability distribution of $P_R(r, \mathbf{x})$ can be calculated by the convolution

$$p(P_R(r) \mid \mathbf{x}) = \left(\underset{H_i \in H(r)}{*} p_i \right) (P_R(r) \mid \mathbf{x}) \,. \tag{4.5}$$

By choosing an appropriate and variable resolution during the calculation of the objects via ray-casting, which results in an adapted Ω_i, we can achieve equal incident signal powers P_i for all objects $H_i \in H(r)$. Thus, this quantity can be simplified to

$$p(P_R(r) \mid \mathbf{x}) = \sum_{j=0}^{|H(r)|} \left(\frac{\binom{|H(r)|}{j}}{2^{|H(r)|}} \cdot \alpha^j \cdot (1-\alpha)^{|H(r)|-j} \cdot \delta\left(P_R(r, \mathbf{x}) - j \cdot P_i\right) \right) \,. \tag{4.6}$$

Here, we exploit the fact that the Dirac delta is the neutral element of convolution. For large values of $|H(r)|$, this binomial distribution can be approximated by the Gaussian

$$p(P_R(r) \mid \mathbf{x}) \approx \mathcal{N}\big(P_R(r); \alpha\, P_{\max}(r, \mathbf{x}),\ \alpha\,(1-\alpha)\, P_{\max}(r, \mathbf{x})\big) \,. \tag{4.7}$$

The mean $\mu = P_{\max}\ \alpha$ and the variance $\sigma^2 = P_{\max}\ \alpha\,(1-\alpha)$ of this Gaussian depend on the reflection proportion α and the maximum power

$$P_{\max}(r, \mathbf{x}) = \sum_{H_i \in H(r)} P_i \tag{4.8}$$

that would be received if all objects in distance r would reflect the ultrasound signal towards the receiver. An example of $P_{\max}$ and the corresponding measurement likelihood is shown in Figure 4.5.

In the measurement process, the received signal is amplified by some predefined factor f_a and the threshold circuit causes the sensor to measure the shortest distance out of

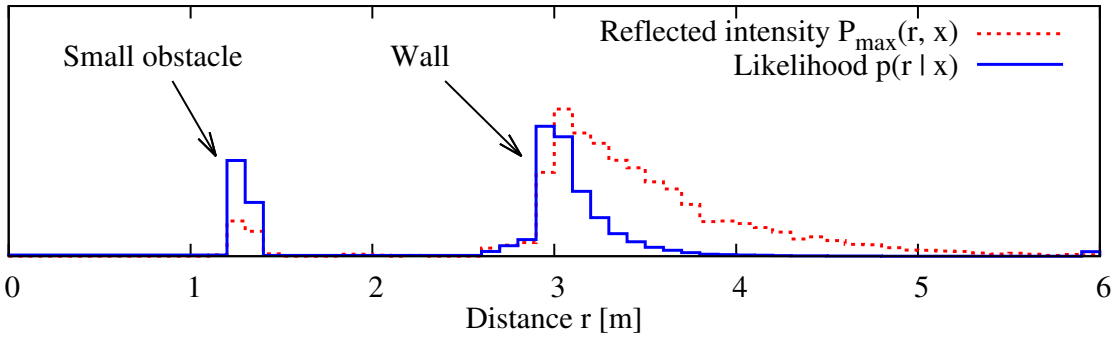


Figure 4.5: An example of the maximum power P_{max} that would be received if all objects in distance r reflect the ultrasound signal towards the receiver and the corresponding measurement likelihood.

which the received and amplified signal exceeds a given threshold P_E. Consequently, the measurement probability

$$p'(r_i \mid \mathbf{x}) = p\left(f_{\text{a}} \cdot P_R(r_i) > P_E \mid \mathbf{x}\right) \cdot \left(1 - \sum_{j<i} p'(r_j \mid \mathbf{x})\right) \tag{4.9}$$

is the product of the probability that the received and amplified signal exceeds the threshold and the probability that the measurement procedure has not already been terminated. In the course of this, we discretize the measured distances into $r_0, \ldots, r_M$ similar to Moravec [115].

Additionally, dynamic, unmapped objects such as people or other robots could influence the measurements. This effect is modeled by a small probability for detecting dynamic objects β, which extends the measurement likelihood to

$$p''(r_i \mid \mathbf{x}) = \left((1-\beta)\, p'\left(f_{\text{a}} \cdot P_R(r_i) > P_E \mid \mathbf{x}\right) + \beta\right) \cdot \left(1 - \sum_{j<i} p''(r_j \mid \mathbf{x})\right) . \tag{4.10}$$

Furthermore, the sensor could fail and generate measurements uniformly distributed over the whole measurement range. This can be modeled by the uniform random measurement probability and leads to the overall likelihood

$$p(r_i \mid \mathbf{x}) = (1-\gamma) \cdot p''(r_i \mid \mathbf{x}) + \gamma \cdot p_{\text{uniform}}(r_i) . \tag{4.11}$$

All in all, there are four parameters to be determined in the model training stage: the reflection proportion α, the probability of dynamic objects β, the probability of random measurements γ, and the receiver threshold P_E. We learn these parameters from data by maximizing the joint log likelihood

$$\log p(r_{1:T} \mid \mathbf{x}^{\star}_{1:T}, \alpha, \beta, \gamma, P_E) = \sum_{t=1}^{T} \log p(r_t \mid \mathbf{x}^{\star}_t, \alpha, \beta, \gamma, P_E) \tag{4.12}$$

Figure 4.6: The environment used to evaluate our localization system with different sensor models.

of the sonar measurement data $r_{1:T}$ given the ground truth trajectory $\mathbf{x}^{\star}_{1:T}$ of the airship. The resulting likelihood is shown in Figure 4.5. In contrast to the ray-casting model, our sensor model specifies a multi-modal likelihood, which explicitly models different object sizes and takes into account multiple objects in different distances. Therefore, our model can better deal with the large opening angle of miniature sonar sensors.

4.3 Experimental Evaluation

The sensor model described above has been implemented and evaluated using real data acquired in a large indoor environment with our first blimp prototype described in Section 2.1.1. The blimp is equipped with four Devantech SRF10 sonar sensors (see Figure 4.1) with a measurement range up to $6\,\mathrm{m}$. Each sensor has two membranes with a diameter of $d_{\mathrm{m}} = 8.5\,\mathrm{mm}$ and a wavelength of $\lambda = 8.5\,\mathrm{mm}$. Three sonar sensors are mounted horizontally at the front, left, and right side of the hull. The fourth sensor is integrated into the gondola pointing downwards to measure the height. Additionally, the blimp is equipped with an IMU [156] that provides accurate attitude and heading estimates.

The indoor environment in which we carried out the experiments is shown in Figure 4.6. It provided an area for flying of about $14\,\mathrm{m} \times 7\,\mathrm{m}$ with a vertical space of $5\,\mathrm{m}$. The multi-level surface map (see Section 2.3) representing the environment had a resolution of $0.1\,\mathrm{m}$ and was created from 3D laser scans. In this map, we precalculated the set of potentially reflecting objects for our sensor model by ray-casting using a fixed angular resolution of 5°.

As described in Chapter 3, we applied particle filter localization using the physical simulation-based control motion model introduced in Section 2.6. We obtained ground truth poses from the images of the camera integrated in the gondola of the blimp by detecting the markers placed on the floor (see Section 2.5).

To compare our novel sensor model to the ray-casting model, we learned the parameters of both models from real data by mounting the sonar sensor on a wheeled robot. We determined the corresponding sensor poses using a laser-based localization approach and calculated the parameters using the given map and $40{,}000$ sonar measurements. There were virtually no dynamic objects and very few wrong measurements while we acquired the data. As a result, the values for the corresponding parameters β and γ of our model were lower than 0.01.

In order to evaluate the improvement in terms of the localization error, we compared the performance of our novel sensor model to the standard ray-casting model. We used the Euclidean distance between the weighted average of all particles and the ground truth pose as a measure of localization error.

In an extensive experiment of about 23 min of manually operated flight, the blimp collected $13{,}430$ sonar measurements. Figure 4.7 shows the path of the blimp as estimated by the localization system using our novel sensor model. As can be seen from Figure 4.8, our novel sensor model resulted in a significantly smaller localization error than the standard ray-casting model. Furthermore, we evaluated the localization success rate, which revealed that the number of particles required to reliably localize the blimp is substantially smaller using our sensor model.

4.4 Related Work

Before laser scanners became available for installation on mobile robots, ultrasound sensors were popular sensors for estimating the distance to objects in the environment of a robot. Typically, robots were equipped with arrays of Polaroid ultrasound sensors, which had, compared to the sensors installed on our blimp, a relatively small opening angle. In the literature, several approaches to model the behavior of such ultrasound sensors can be found.

Some approaches utilize ray-casting operations to estimate the distance to be measured according to a given map. One of the first of such approaches to model ultrasound sensors in the context of localization and mapping is the pioneering work by Moravec and Elfes [115, 116]. The sensor model approach described there is somewhat similar to ours. However, it has originally been designed for two-dimensional occupancy grid maps only and also does not specifically model the intensity decrease of the sound cone while it propagates. A corresponding model has been utilized by Burgard et al. [20] and has been shown to allow a mobile robot to robustly localize itself using Markov

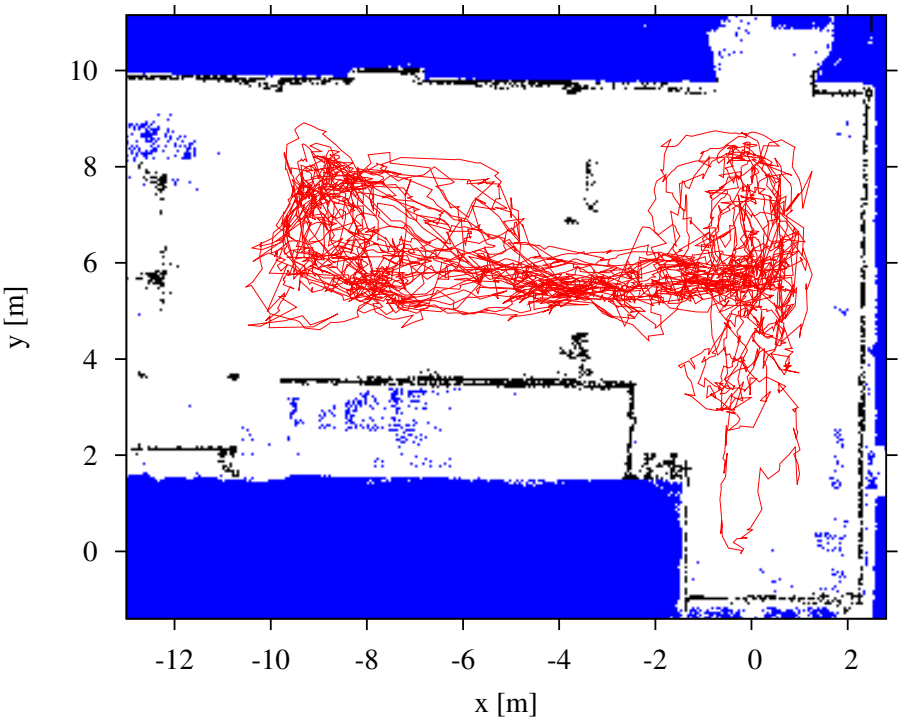


Figure 4.7: A horizontal sectional view of our testing area at 2.0 m height. Obstacles are shown in black, unknown space is blue. The path localized with the novel sensor model is depicted in red.

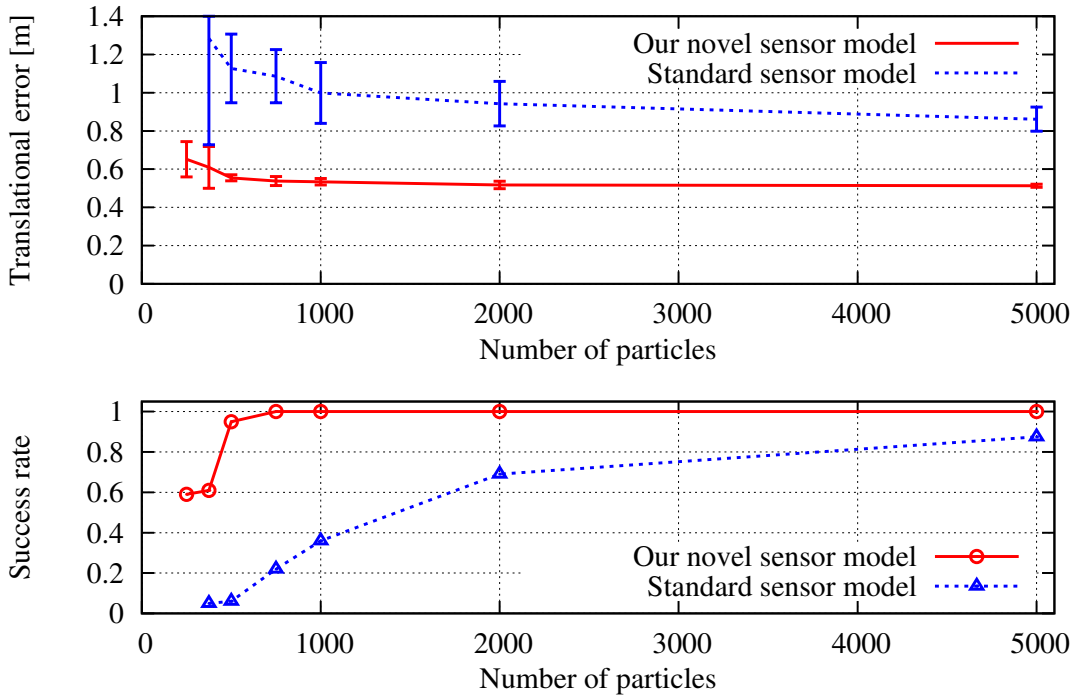


Figure 4.8: The average translational RMS localization error and the success rate of our novel sensor model in comparison to the standard ray-casting model. The error bars indicate the 99.7 % confidence intervals over ten runs.

Localization, a grid-based variant of recursive Bayes filters. Thrun [166] proposed an approach to occupancy grid mapping that considers multiple objects in the sound cone. However, this approach utilizes a simplified sensor model. Fox et al. [50] presented a sensor model for range measurements that has been designed especially for robots operating in dynamic environments. It also does not explicitly model the intensity changes on the surface of the sound cone.

Crowley [30] estimated lines from sonar measurements and applied an extended Kalman filter for state estimation. One of the assumptions in this work is that the arc corresponding to the front of the sound wave can be approximated by a Gaussian, which is only justified for ultrasound sensors with a small opening angle. Additionally, several authors have presented so-called endpoint or correlation models, which are more efficient but ignore the area intercepted by the sound cones [90, 164]. Schroeter et al. [152] directly learned the likelihood function from data collected with a mobile robot, which is an approach similar to the one described by Thrun et al. [169]. Compared to these approaches, our technique seeks to rigorously model the physical measurement process of the sensor and explicitly takes into account the potential reflections of objects.

Physical models have also been considered by Leonard and Durrant-Whyte [104]. Their approach assumes certain types of geometric objects, such as planes, cylinders,

corners, and edges, in the context of a landmark-based SLAM algorithm. In this context, Tardós et al. [163] utilized a similar approach to extract lines and corners to robustly build large-scale maps based on ultrasound data. Compared to these techniques, our approach does not rely on the assumption that the environment consists of certain types of geometric objects. Rather, it can be applied to arbitrary indoor environments. Additionally, these approaches assume relatively accurate odometry, which is typically not available in the context of airships.

4.5 Conclusions

In this chapter, we presented a novel sonar sensor model for probabilistic localization techniques that explicitly considers the characteristics of small sonar sensors with large opening angles. Our approach is rigorously based on the physics of sonar sensors. It explicitly takes the propagation of their hardly focused sound signal into account and models the signal reflection by objects with different sizes and distances. In this way, it specifies a multi-modal likelihood distribution. Practical experiments with a real miniature blimp demonstrate that our novel sensor model allows the blimp to robustly localize itself in a known environment. It also significantly outperforms the popular ray-casting model in terms of the localization accuracy and the number of particles needed.

Chapter 5

Air Flow Sensor Model

While airships are attractive as they can move freely in the three-dimensional space, their high-dimensional state space and the restriction to small and lightweight sensors are demanding constraints with respect to self-localization. Furthermore, their complex second-order kinematics makes the estimation of their pose and velocity through double integration in the motion model difficult and imprecise. In this chapter, we consider the problem of estimating the velocity of a miniature blimp with lightweight air flow sensors. We present a probabilistic sensor model that accurately models the uncertainty of the flow sensors and thus allows for robust state estimation using a particle filter. In experiments carried out with a real airship, we demonstrate that our method precisely estimates the velocity of the blimp and outperforms the standard velocity estimates of the motion model as applied in many existent autonomous blimp navigation systems.

• • • • •

Measuring the airspeed is an important capability of modern airplanes and a failure of the corresponding sensors, the pitot tubes, can cause them to crash [126]. As opposed to large airplanes, indoor airships do not require accurate airspeed measurements for flight stabilization. However, airspeed sensors are often applied for velocity or wind speed estimation on small UAVs [45, 171]. Furthermore, they can also be used for dead reckoning in the context of localization, because the motion prediction of flying vehicles usually suffers from large accumulating errors.

In this chapter, we consider the problem of designing an air flow sensor model that can be applied in particle filter localization as well as in dead reckoning. In particular, we introduce a probabilistic sensor model $p(\mathbf{z} \mid \mathbf{x})$, which defines the likelihood of a measurement $\mathbf{z}$ given the state of the system $\mathbf{x}$. In the context of air flow sensors, the

measurement is a scalar value $z \in \mathbb{R}$ and its likelihood depends on the velocity of the airship the sensor is rigidly attached to.

There exist various techniques for measuring air velocity. While cup, windmill, and sonic anemometers are rather heavy and bulky, hot-wire anemometers and thermal mass flow meters can be built in MEMS technology and therefore are suitable even for employment on miniature flying vehicles. To measure the airspeed, our blimp is equipped with MEMS-based thermal air flow micro-sensors. These smart sensors determine the velocity of media sweeping over them through the detection of on-chip thermal differences [110]. However, the measurement characteristics of these thermal flow sensors are nonlinear and the measurement noise depends on the velocity of the air sweeping over the sensor.

We present a probabilistic air flow sensor model that is based on the measurement characteristics of the sensors. To approximate the characteristics of the air flow sensors including their heteroscedastic measurement noise, we present and compare two regression techniques. Both are suitable for probabilistic sensor data fusion in a particle filter and therefore our sensor model enables robust state estimation through the modeling of all underlying uncertainties.

In the following, we first briefly discuss the placement of the air flow sensors on miniature indoor airships. We introduce our probabilistic air flow sensor model, which is suitable for dead reckoning (see Chapter 8) as well as state estimation in the particle filter. In practical experiments, we demonstrate that our approach to velocity estimation with air flow sensors outperforms the simulation-based control motion models applied in many autonomous blimp navigation systems [55, 88, 183].

5.1 Sensors and Placement

Although various other types of miniature air flow sensors are available [34, 181], we chose the SDP600 differential pressure sensors from Sensirion AG, Stäfa, Switzerland, shown in Figure 5.1. They are operated here as thermal flow sensors and have several desirable properties. They are fully developed, have a weight below $1\,\mathrm{g}$, a very low power consumption, quickly react to changes in the gas flow, and their integrated evaluation circuitry can be controlled via the $\mathrm{I}^2\mathrm{C}$ interface available on the Gumstix computer of the blimp. As an additional sensor setup, we mounted a short tube onto one of the sensors to reduce the air turbulences in the vicinity of the thermal elements of the sensor. This sensor is called *tube sensor* in the following.

When mounting air flow sensors on the airship, one has to take into account various aerodynamic effects of the air flowing around the airship. First, due to the viscosity of the air, there is a boundary layer in the vicinity of flat surfaces as illustrated in Figure 5.2. Second, the airship typically operates at velocities at which the air flow is turbu-

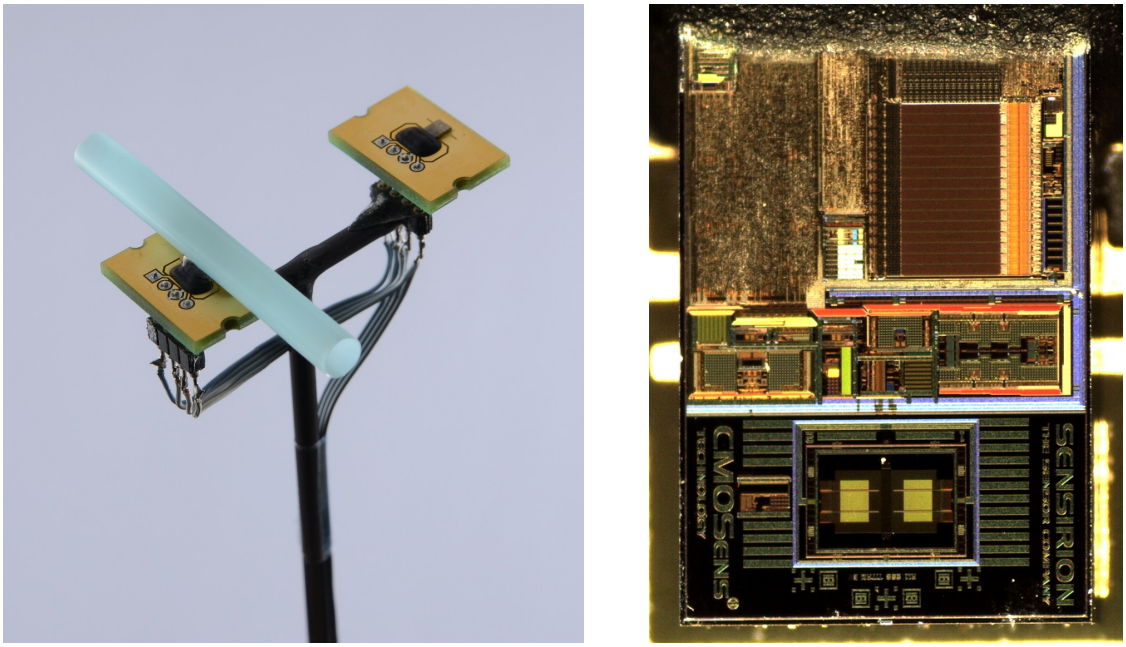


Figure 5.1: Left: Two Sensirion SDP600 air flow sensors are mounted on a pole in parallel. One of them is equipped with a short piece of a tube above the sensing element. Right: A detailed picture of the sensing element.

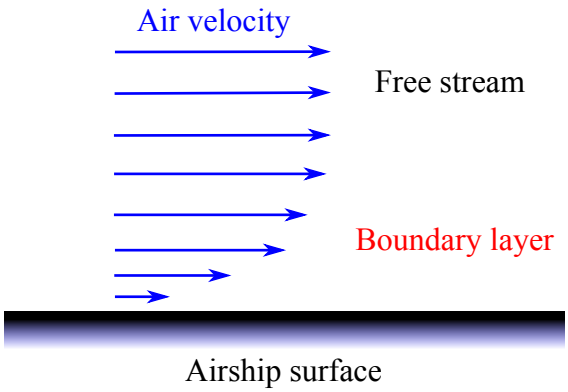


Figure 5.2: The air velocity profile of a laminar flow in the vicinity of an approximately flat surface.

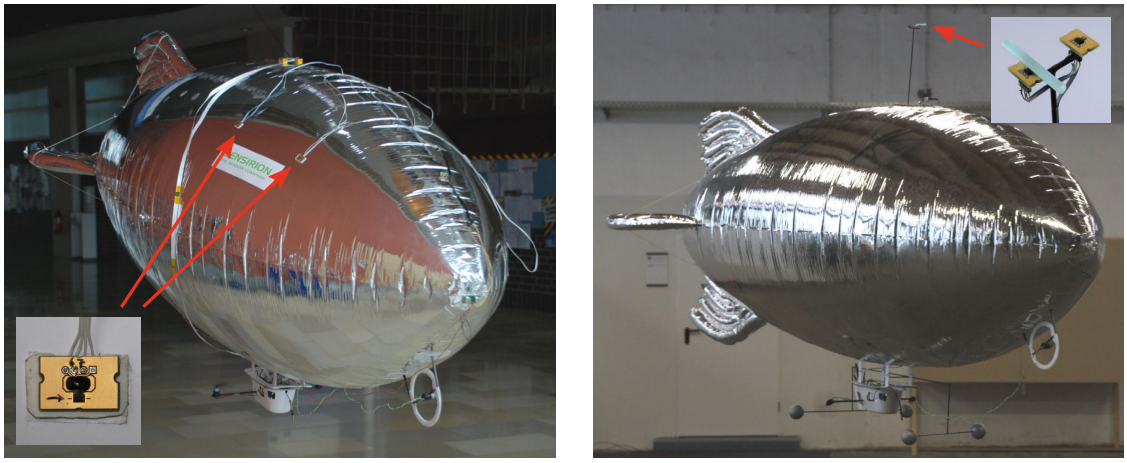


Figure 5.3: Different mounting methods of the air flow sensors on the blimp. The picture on the left shows the blimp with four sensors (two of them are occluded) mounted directly on the surface of the blimp. The picture on the right shows two sensors mounted in parallel on a pole that has a height of 20 cm.

lent (see Section 2.6) and a substantial amount of the surrounding air is accelerated with the airship and accompanies it [98]. While the thickness of the boundary layer can be calculated in closed form for flat surfaces [111], the surface of the blimp is substantially curved and therefore a flat surface approximation would be hard to justify. Furthermore, modeling the turbulences and the accompanying air usually requires extensive simulations of fluid dynamics or costly wind tunnel experiments.

Therefore, we performed a couple of practical experiments in which we evaluated different mounting methods of the air flow sensors. Since we aim to mount the sensors in a way so that the influences of the boundary layer and turbulences are reduced in the measurements, we evaluated the mounting methods by relating the measurements to the expected ones assuming that the sensors are operating in the free stream. In our experiments, we evaluated three mounting positions where two of them are shown in Figure 5.3:

1. **Directly on the surface**. In our first experiments, we placed four sensor chips, similarly to Fei et al. [45] and Tokutake et al. [171], at different positions directly on the hull of the blimp. In this configuration, the sensing element of each sensor sticks up approximately 5 mm from the airship surface.

2. **On a 10 cm pole**. In a second experiment, we mounted a 10 cm high pole on the top of the blimp so that the sensor is on an exposed position in 10 cm distance to the hull.

3. **On a 20 cm pole**. In a third experiment, we extended the pole to 20 cm height so that the sensor is mounted in 20 cm distance to the hull.

The measurements in the experiments with the first two methods showed a non-negligible influence of the surrounding air accompanying the blimp, which is hard to model. Even when placing the sensors on a pole at a distance of 10 cm from the hull, the velocity estimates provided by our particle filter approach were worse than those obtained with the plain motion model. In contrast, the third method turned out to be a good trade-off between a sufficiently large distance to the hull to minimize the influence of aerodynamic effects on the one hand and a low distance to keep good navigation capabilities when navigating close to obstacles on the other hand. Therefore, we decided to mount all flow sensors in a distance of 20 cm to the hull in our further system setups.

5.2 Probabilistic Flow Sensor Model

Like many types of air flow sensors, the thermal flow sensors that we apply on our blimp provide a scalar measurement value $z \in \mathbb{R}$ that depends on the air velocity v_z along the fixed measurement axis of the sensor. In the following, we present our probabilistic sensor model that specifies the measurement likelihood $p(z \mid \mathbf{x})$ of air flow sensors with a fixed measurement axis.

We model the measurement principle by assuming Gaussian noise in the heteroscedastic measurement process

$$z = h(v_z) + \varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(\varepsilon; 0, \sigma(v_z)^2) \tag{5.1}$$

where h is a strictly monotonic increasing function. The noise ε typically depends on the sensor characteristics as well as on the air velocity.

In indoor navigation scenarios, we assume the air to be static (no wind) and the sensor to be placed at a sufficient distance from the hull so that the influence of the surrounding air accompanying the blimp can be neglected. The measurement depends on the translational velocity $\mathbf{v}$ and rotational velocity $\boldsymbol{\omega}$ of the blimp where both are contained in the state vector $\mathbf{x}$. The velocity of the point at the position $\mathbf{r}_z$ in the body-fixed frame of reference where the sensor is rigidly mounted, is $\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_z$. Hence, the velocity component of the air sweeping over the sensor along its measurement axis $\mathbf{n}_z$ is defined through the projection

$$v_z(\mathbf{x}) = (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_z) \cdot \mathbf{n}_z \ . \tag{5.2}$$

According to Equation (5.1) the probabilistic measurement model used for Monte Carlo state estimation is defined by the Gaussian distribution

$$p(z \mid \mathbf{x}) = \mathcal{N}(z; h(v_z(\mathbf{x})), \sigma(v_z(\mathbf{x}))^2) \ . \tag{5.3}$$

For the implementation of the model described above, we learn functions approximating $h : \mathbb{R} \to \mathbb{R}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ from a set of training data $\{(x_i, y_i)\}_{i \in [1,n]}$. Each training

data point contains the velocity v_z of the sensor relative to the air in x_i and the measured flow value z in y_i. According to our model, all points are assumed to be generated from $y_i = h(x_i) + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(\varepsilon_i; 0, \sigma(x_i)^2)$. In the following, we present a non-parametric and a parametric regression approach to approximate h and σ. As shown in Figure 5.4, both are suitable for our regression analysis problem.

5.2.1 Local Linear Regression

In general, local regression [176] computes a weighted average of the training function values y_i giving a higher weight to those points near the requested value x. In our approach, we apply the Gaussian kernel

$$w(x, x') = \frac{1}{\sqrt{2\pi}\, l} \exp\left(-\frac{(x - x')^2}{2\, l^2}\right) \tag{5.4}$$

with bandwidth l for local weighting. For a compact representation, we define

$$W(x) = \frac{1}{\sum_{i=1}^{n} w(x, x_i)} \operatorname{diag}(w(x, x_1), \ldots, w(x, x_n)) \,, \tag{5.5}$$

$$X = \begin{bmatrix} 1 & \ldots & 1 \\ x_1 & \ldots & x_n \end{bmatrix} \,, \tag{5.6}$$

$$\text{and} \quad Y = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix} \,. \tag{5.7}$$

In local linear regression, each function value is computed from a linear regression

$$f(x) = A(x) \begin{bmatrix} 1 \\ x \end{bmatrix} \tag{5.8}$$

where the coefficient matrix A minimizes the locally weighted sum of squared errors

$$(Y - A\,X)^T\, W(x)\, (Y - A\,X) \,. \tag{5.9}$$

In the linear, one-dimensional case it holds that $A = [a_0, a_1]$. Minimizing Equation (5.9) gives the weighted least squares estimator

$$\hat{A}(x) = Y\, W(x)\, X^T\, (X\, W(x)\, X^T)^{-1} \tag{5.10}$$

and finally the estimated function value

$$\hat{f}(x) = \hat{A}(x) \begin{bmatrix} 1 \\ x \end{bmatrix} \,. \tag{5.11}$$

For our flow sensor model, we extend the local linear regression by an estimate of the variance σ^2. In the training stage, we calculate $\varepsilon_i = y_i - \hat{f}(x_i)$ for each training data point. Based on these values, we estimate $\sigma(x)^2$ as the local constant regression [176] on the squared residuals:

$$\hat{\sigma}(x)^2 = \frac{\sum_{i=1}^{n} w(x, x_i)\, \varepsilon_i^2}{\sum_{i=1}^{n} w(x, x_i)} \,. \tag{5.12}$$

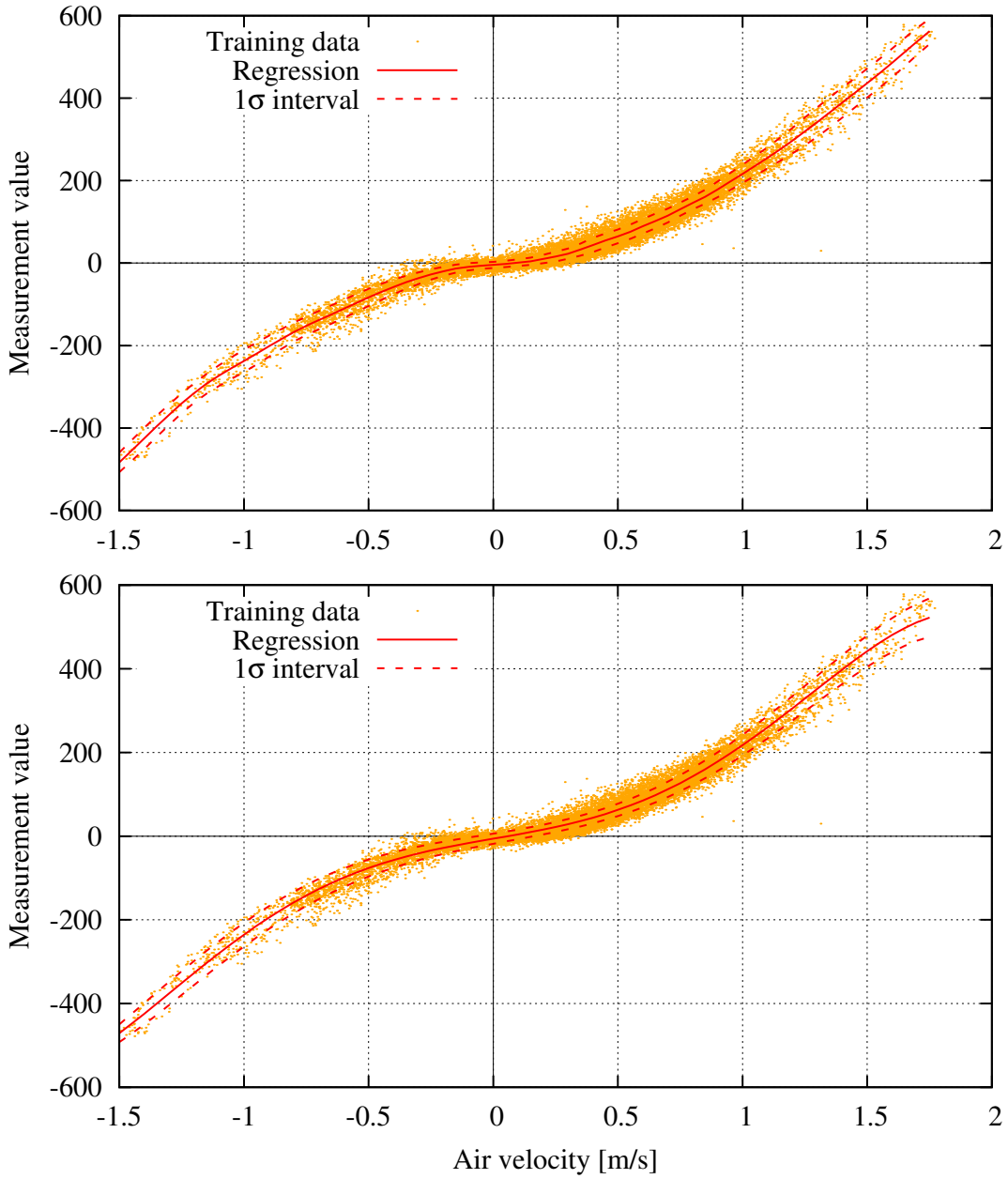


Figure 5.4: The local linear regression (top) and the polynomial regression (bottom) on the tube sensor training data generated from about 20 min of operation. The regression on the measurement noise is represented by the 1σ interval.

5.2.2 Polynomial Regression

An alternative technique is the polynomial regression. It is a parametric representation and therefore less flexible than local regression, but the evaluation of the polynomial is usually more efficient than applying a local linear regression to the training data. For a compact representation of the polynomial function

$$f(x) = \sum_{d=0}^{p} a_d x^d = A_p \, [1, x^1, \ldots, x^p]^T \tag{5.13}$$

of degree p, we define the regression parameter $A_p = [a_0, \ldots, a_p]$ and

$$X_p = \begin{bmatrix} 1 & \ldots & 1 \\ x_1^1 & \ldots & x_n^1 \\ \vdots & & \vdots \\ x_1^p & \ldots & x_n^p \end{bmatrix} . \tag{5.14}$$

Minimizing the squared sum of estimation errors

$$(Y - A_p \, X_p)^T \, (Y - A_p \, X_p) \tag{5.15}$$

on the training data gives the polynomial least squares estimator [176]

$$\hat{A}_p = Y \, X_p^T \, (X_p \, X_p^T)^{-1} \tag{5.16}$$

and finally the estimate $\hat{f}(x) = \hat{A}_p \, [1, x^1, \ldots, x^p]^T$. Here, we estimate the variance $\sigma(x)^2$ by another polynomial regression

$$\hat{\sigma}(x)^2 = \hat{A}'_p \, [1, x^1, \ldots, x^p]^T \tag{5.17}$$

on the squared residual with $\varepsilon_i = y_i - \hat{f}(x_i)$ and

$$\hat{A}'_p = [\varepsilon_1^2, \ldots, \varepsilon_n^2] \, X_p^T \, (X_p \, X_p^T)^{-1} . \tag{5.18}$$

As shown in Figure 5.4, both regression techniques are suitable for our flow sensor model.

5.2.3 Efficient Model Approximation

Although it is computationally more demanding than the polynomial regression, we apply local linear regression, because it has better extrapolation properties, which are required in cases where the test data temporarily leaves the area covered by the training data. Furthermore, local linear regression better models the heteroscedastic uncertainty

compared to the polynomial regression, which does not ensure that the estimated variance is positive, especially when extrapolation is required.

The main drawback of local linear regression, however, is its computational complexity. A single function call is in $O(n)$ where n is the typically large number of training data points. Additionally, the particle filter requires to call the regression function for every particle on every incoming air flow measurement such that an online localization requires a fast approximation of the regression. To obtain a fast approximation, we discretize the function and store a dense grid containing the function and variance values. This discrete approximation can be calculated once and is stored in a table for fast approximative function calls with linear interpolation in $O(1)$.

Furthermore, the fast approximation of the regression allows to efficiently calculate the inverse h^{-1} of the strictly monotonic increasing measurement function, which is required for airship odometry from air flow and IMU sensors (see Chapter 8). We calculate the inverse function value through a binary search in the function values of the model approximation and apply linear interpolation. This results in a complexity of $O(\log m)$ where m is the number of grid points in the discrete approximation.

5.3 Experimental Evaluation

We evaluated our probabilistic air flow sensor model in extensive experiments with our second blimp prototype (see Section 2.1.2) in a large indoor environment of about $20\,\text{m} \times 12\,\text{m}$ with a vertical space of $5\,\text{m}$. The blimp was equipped with two SDP600 differential pressure sensors, which are operating as thermal flow sensors. As a result of our preliminary experiments, we mounted the flow sensors on a pole at a distance of $20\,\text{cm}$ from the hull. Because the estimation of the forward velocity usually suffered from large errors, we placed the sensors on top of the blimp with the measurement axis pointing forward. In particular, we mounted two sensors on the pole in parallel: one sensor with a tube and a plain sensor as shown in the Figures 5.1 and 5.3. During all experiments, we obtained accurate ground truth states from an optical motion capture system (see Section 2.5).

In the model training stage, we recorded training data from about $20\,\text{min}$ of manually operated flight including $24{,}196$ measurements of each air flow sensor at $20\,\text{Hz}$. Using this data, we trained the flow sensor models as shown in Figure 5.4 and also learned the parameters of our motion model (see Section 2.6).

To evaluate the performance of our novel air flow sensor model with respect to state estimation, we applied the model in a particle filter as described in Section 3.2. To reduce the influence of other sensors, we utilized only the air flow sensors together with the airship motion model for velocity estimation in the particle filter. In our implementation, we restricted the considered points in the local linear regression to those having

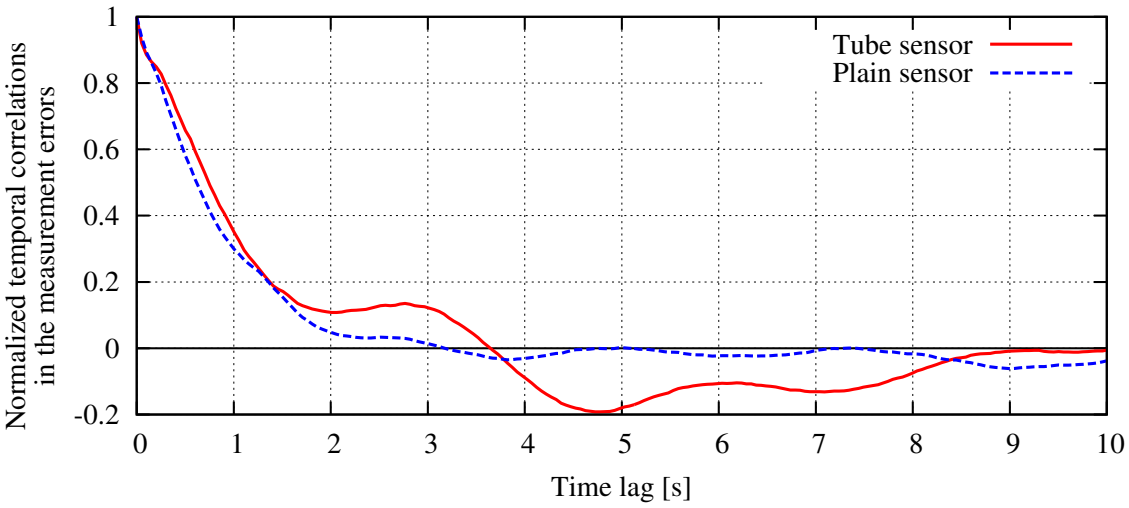


Figure 5.5: The temporal correlations in the measurement errors of the air flow sensors that sense the forward velocity of the blimp. Each measurement error is computed as the deviation between the actual measurement and the prediction of our local linear regression air flow sensor model given the ground truth velocity information.

a significant weight for the reason of efficiency. We chose the bandwidth $l = 0.1$ of the Gaussian kernel and the polynomial degree $p = 5$. We evaluated our approach in terms of the state estimation error in an experiment of about 12 min of manual operation. Throughout this experiment, the operator carried out many different maneuvers including strong accelerations and rotations, which also caused rocking movements of the blimp. We evaluated the quality of the velocity estimates in terms of the root mean square error (RMSE) of the estimated forward velocity v_x with respect to the ground truth value $v_x^\star$ (see Section 2.5).

The Bayes filtering scheme on which our state estimation approach is based relies on the Markov assumption [169] that the measurement noise of consecutive measurements is conditionally independent. To validate this with respect to our flow sensor model we evaluated the autocorrelation of the measurement noise depending on the time lag (see Figure 5.5). Despite the high level of correlation even at a time lag of 1 s, the results of our approach showed its optimal performance when a flow measurement is integrated into the belief of the filter every 0.2 s (see Figure 5.6). This is caused by the fact that the filter benefits from the information of more frequent measurements even if there are a limited correlations that are not modeled. In our experiments, the tube sensor significantly outperforms the plain sensor and the local linear regression model performs slightly better than the polynomial regression model, at least for the tube sensor.

Figure 5.7 shows the robustness of our approach to the Monte Carlo approximation with a limited number of particles. As a trade-off between accuracy and runtime, 500 particles seem to be the best choice. Furthermore, we compared the forward velocity

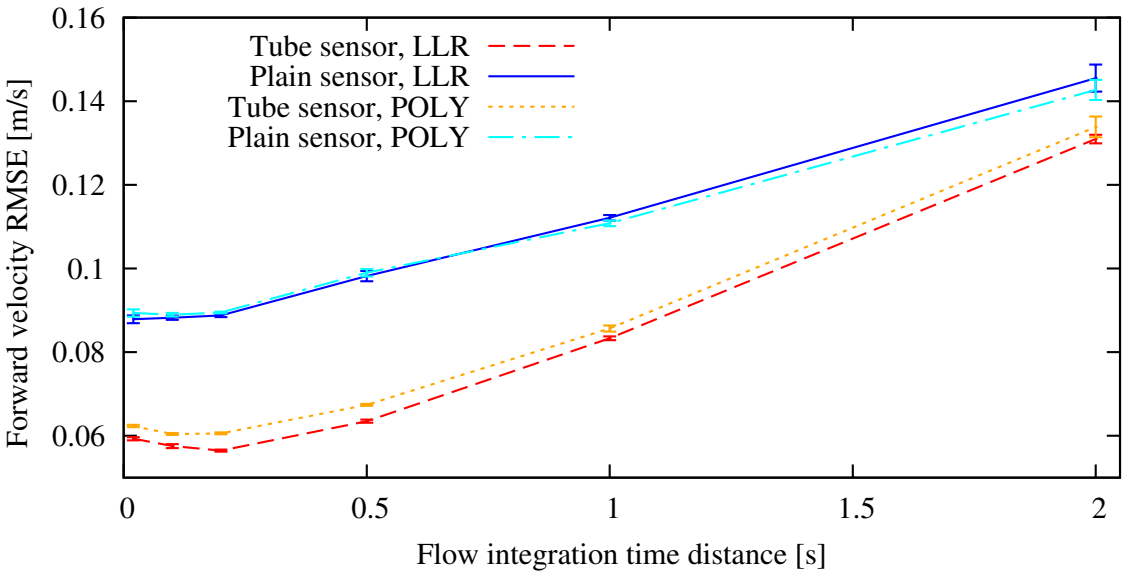


Figure 5.6: The average forward velocity RMS error of the different flow sensors together with the local linear regression (LLR) and the polynomial regression (POLY) compared for different time distances between two integrations of sensor measurements into the particle filter. The error bars indicate the 95 % confidence intervals over ten runs.

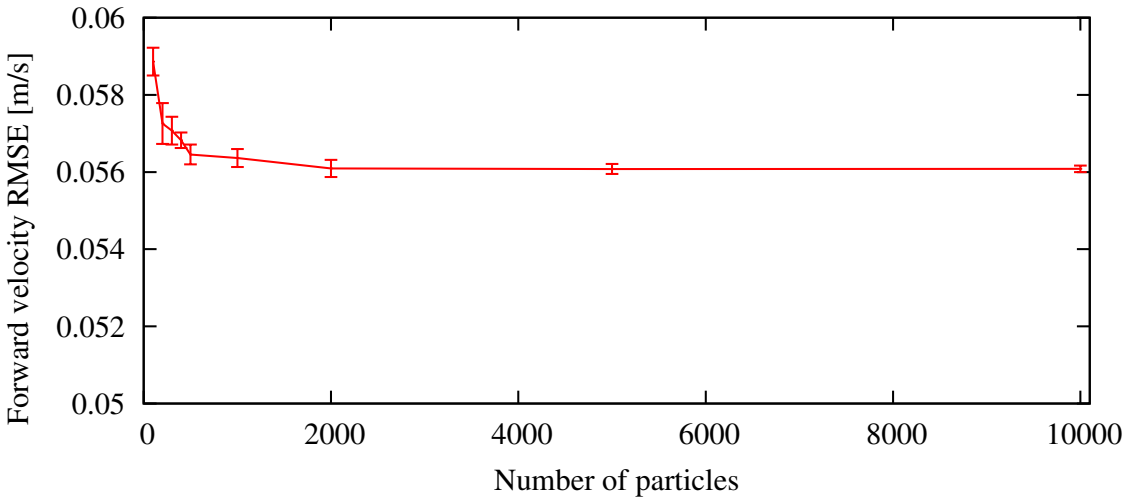


Figure 5.7: The average forward velocity RMS error for different numbers of particles in our filter with the tube flow sensor and the local linear regression model. The error bars indicate the 95 % confidence intervals over ten runs.

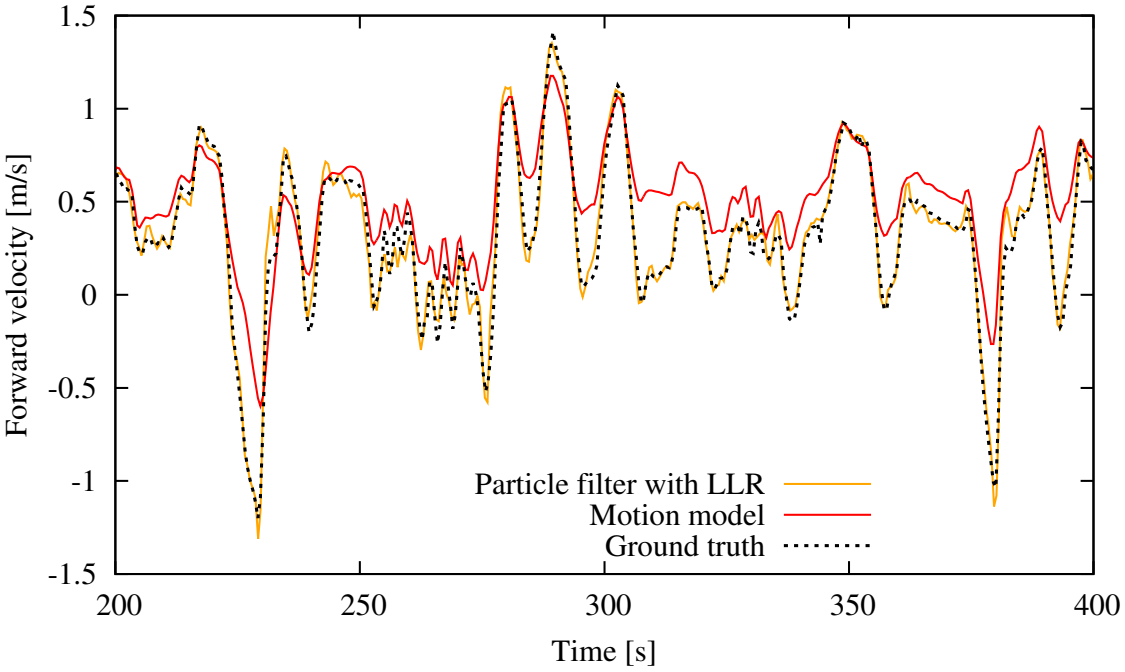


Figure 5.8: An exemplary extract of the forward velocity estimates of our filter with the tube flow sensor with the local linear regression (LLR) model compared to the motion model estimates and the ground truth data.

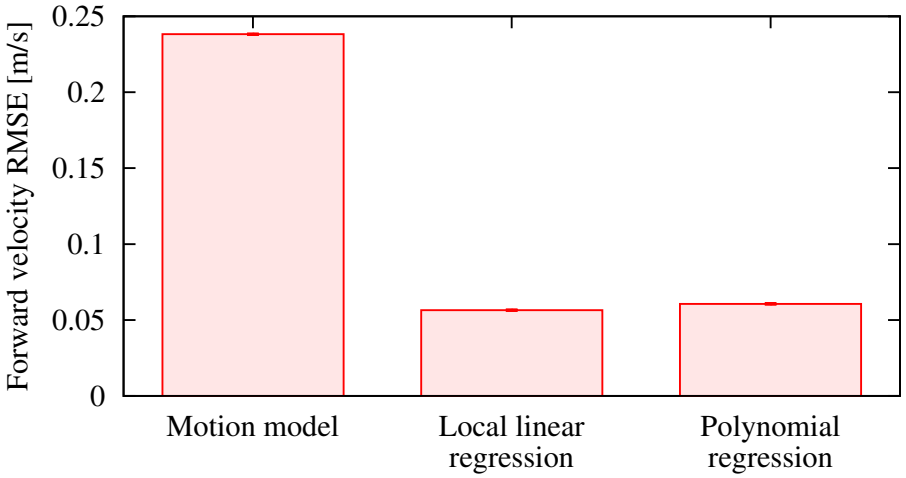


Figure 5.9: The average forward velocity RMS error of the estimates of our filter with the tube flow sensor compared to the motion model estimates. The error bars indicate the 95 % confidence intervals over ten runs.

estimates of our approach with both flow sensor models to those of the motion model without any sensor data fusion. As shown in Figure 5.8 and 5.9, both sensor models significantly outperform the plain motion model and therefore are beneficial to an improved localization of miniature airships.

5.4 Related Work

A popular application of airspeed sensors on UAVs is the combination with GPS for wind estimation [27]. In this context, approaches to calibrate the scaling of an airspeed sensor have also been developed [24].

Furthermore, several authors considered state estimation or control of robots based on flow sensors. For example, Fei et al. [45] and Tokutake et al. [171] utilized thermal air flow sensors on the wings of small unmanned aircrafts for the detection of flight parameters including the airspeed based on a neural network and piecewise quadratic regression, respectively. Kruusmaa et al. [92] determined the optimal position of pressure sensors on an artificial trout to estimate the velocity using a quadratic regression model. For attitude estimation, Euston et al. [44] fused IMU and airspeed measurements of a UAV.

Many researchers used optical flow on image sensors such as the low cost devices employed in optical mice for improved dead-reckoning odometry on ground robots [85]. Dille et al. [38] additionally apply an online re-calibration of the local linear regression calibration for an optical flow sensor. Likewise, Conroy et al. [29] determined the maximum likelihood velocity of a quadrotor using an omnicam-based optical flow sensor. Similar techniques have been applied on micro aerial vehicles such as the palm-sized glider designed by Woods et al. [179] inspired by flying insects. Here, the authors utilized an embedded low resolution optical flow sensor for target detection and obstacle avoidance. In addition to an optical flow sensor, some authors employed airspeed sensors for flight stabilization [83] or even ground speed and wind estimation [148].

However, the approaches mentioned above apply maximum likelihood state estimation or control of robots based on the calibrated output of one or multiple identical flow sensors. In contrast, we explicitly model the uncertainty of the measurements and of the motion of the robot for probabilistic state estimation. Thus, our approach can seamlessly integrate arbitrary sensors and takes into account the control signals sent to the actuators of the robot.

5.5 Conclusions

In this chapter, we presented a novel probabilistic model for air flow sensors, which is suitable for dead reckoning odometry as well as for probabilistic velocity estimation

for miniature indoor airships. In contrast to other approaches, we explicitly consider the measurement uncertainties of the thermal air flow sensors and probabilistically fuse their measurements with the prediction calculated by the airship motion model in a particle filter. Additionally, our approach allows to seamlessly integrate other sensors such as sonars or an IMU for localization. We compared two regression methods for sensor calibration including uncertainties. Both proved to be suitable for probabilistic velocity estimation. Furthermore, we identified suitable air flow sensor setups by investigating a couple of sensor placement methods and different sensor configurations. In experiments with a real blimp operating in a large indoor environment, we demonstrated that our approach accurately estimates the velocity of the blimp and outperforms the velocity estimates of a standard motion model for miniature airships.

Chapter 6

IMU Sensor Model

Today, inertial sensors are a popular means to obtain accurate orientation estimates even in small-sized and low-cost devices. In this chapter, we present a standard probabilistic sensor model for the compensated measurement data for the individual sensor chips applied on an inertial measurement unit (IMU). Furthermore, we propose a probabilistic sensor model that specifies the likelihood of orientation estimates of a recursive filter applied on the IMU. In the context of miniature airship localization, we demonstrate in experiments with a real blimp that the particle filter localization with the sensor model for orientation estimates significantly outperforms the localization with the standard model through its more effective two-stage filter design.

• • • • •

Inertial sensors are a well-studied topic and most commercially available IMUs are equipped with three-axis accelerometers, gyroscopes, and magnetometers. Furthermore, they are usually equipped with a low-power processor, which carries out the compensation for systematic measurement errors resulting in the *compensated measurement data* and the sensor data fusion resulting in the *filtered orientations*, which are accurate orientation estimates using a variant of the Kalman filter. Especially since the sensor chips that are usually applied on IMUs can be built in MEMS technology, lightweight and energy-efficient IMUs are broadly available. Their sensors provide valuable and precise information on the motion of the device so that IMUs have become popular for the localization of robots and mobile devices in general.

In this chapter, we consider two variants of IMU data and introduce corresponding probabilistic sensor models for recursive state estimation. First, we present a standard sensor model for compensated measurement data of the individual sensor chips the IMU is equipped with. Second, we propose a model for the output of the on-board sensor data fusion, namely filtered three-dimensional orientations. Both probabilistic sensor models

specify the likelihood $p(\mathbf{z} \mid \mathbf{x})$ of a measurement $\mathbf{z}$ given the state of the system $\mathbf{x}$, which can be used in the particle filter for localization.

6.1 Sensor Model for Compensated IMU Measurement Data

In this section, we present a standard sensor model for compensated IMU measurement data [170]. We assume that the IMU provides the compensated measurements of the individual sensor chips the IMU is equipped with. In particular, we assume that the measurement vector

$$\mathbf{z}_{\mathrm{I,c}} = \left[\boldsymbol{\omega}_{\mathrm{I,c}}^T, \mathbf{a}_{\mathrm{I,c}}^T, \mathbf{m}_{\mathrm{I,c}}^T\right]^T \tag{6.1}$$

contains the angular velocities $\boldsymbol{\omega}_{\mathrm{I,c}} = [\omega_{\mathrm{I},x}, \omega_{\mathrm{I},y}, \omega_{\mathrm{I},z}]^T$ measured by the gyroscopes, the translational accelerations $\mathbf{a}_{\mathrm{I,c}} = [a_{\mathrm{I},x}, a_{\mathrm{I},y}, a_{\mathrm{I},z}]^T$ measured by the accelerometers, and the magnetic field vector $\mathbf{m}_{\mathrm{I,c}} = [m_{\mathrm{I},x}, m_{\mathrm{I},y}, m_{\mathrm{I},z}]^T$ measured by the magnetometers. Furthermore, we assume that all sensors are aligned with the IMU frame of reference $\mathcal{F}_{\mathrm{I}}$.

As it is common practice, we assume that the individual components of the measurements are statistically independent given the state $\mathbf{x}$ and model their measurement likelihood as the expected values with an additional white Gaussian noise [1, 18]. This results in

$$p(\boldsymbol{\omega}_{\mathrm{I,c}} \mid \mathbf{x}) = \mathcal{N}\left(\boldsymbol{\omega}_{\mathrm{I,c}}; \boldsymbol{\mu}_{\boldsymbol{\omega}}(\mathbf{x}), \sigma_{\mathrm{I,c},\boldsymbol{\omega}}^2 I_{3\times3}\right) \tag{6.2}$$

$$p(\mathbf{a}_{\mathrm{I,c}} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{a}_{\mathrm{I,c}}; \boldsymbol{\mu}_{\mathbf{a}}(\mathbf{x}), \sigma_{\mathrm{I,c},\mathbf{a}}^2 I_{3\times3}\right) \tag{6.3}$$

$$p(\mathbf{m}_{\mathrm{I,c}} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{m}_{\mathrm{I,c}}; \boldsymbol{\mu}_{\mathbf{m}}(\mathbf{x}), \sigma_{\mathrm{I,c},\mathbf{m}}^2 I_{3\times3}\right) \tag{6.4}$$

where $I_{3\times3}$ is the three-dimensional identity matrix and $\sigma_{\mathrm{I,c},\boldsymbol{\omega}}, \sigma_{\mathrm{I,c},\mathbf{a}}, \sigma_{\mathrm{I,c},\mathbf{m}}$ are the standard deviations of the measurement noise of the individual sensors, which can be determined in various ways [1].

Here, all measurements are with respect to the IMU frame of reference $\mathcal{F}_{\mathrm{I}}$. The actual angular velocity and acceleration contained in the state vector $\mathbf{x}$ are expressed with respect to the body-fixed frame of reference $\mathcal{F}_{\mathrm{b}}$ (see Section 2.4). Consequently, the mounting position and orientation $(\mathbf{p}_{\mathrm{I}}, \mathbf{q}_{\mathrm{I}})$ of the IMU with respect to $\mathcal{F}_{\mathrm{b}}$ has to be taken into account. Furthermore, the actual magnetic field vector $\mathbf{m}$ and the gravity g, which are assumed to be constant with respect to the global frame of reference $\mathcal{F}_{\mathrm{g}}$ throughout the operational area, have to be transformed to $\mathcal{F}_{\mathrm{I}}$. When calculating the expected accelerations, we take into account the fictitious forces in the moving IMU frame of reference

$\mathcal{F}_\mathrm{I}$. This results in

$$\begin{bmatrix} 0 \\ \boldsymbol{\mu}_{\boldsymbol{\omega}}(\mathbf{x}) \end{bmatrix} = \mathbf{q}_\mathrm{I} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \odot \mathbf{q}_\mathrm{I}^{-1} \tag{6.5}$$

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{\mu}_{\mathbf{a}}(\mathbf{x}) \end{bmatrix} = \mathbf{q}_\mathrm{I} \odot \begin{bmatrix} 0 \\ \mathbf{a} + \boldsymbol{\omega} \times \mathbf{v} + \boldsymbol{\alpha} \times \mathbf{p}_\mathrm{I} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{p}_\mathrm{I} \end{bmatrix} \odot \mathbf{q}_\mathrm{I}^{-1} \\ + (\mathbf{q} \odot \mathbf{q}_\mathrm{I}) \odot [0, 0, 0, g]^T \odot (\mathbf{q} \odot \mathbf{q}_\mathrm{I})^{-1} \end{aligned} \tag{6.6}$$

$$\begin{bmatrix} 0 \\ \boldsymbol{\mu}_{\mathbf{m}}(\mathbf{x}) \end{bmatrix} = (\mathbf{q} \odot \mathbf{q}_\mathrm{I}) \odot \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} \odot (\mathbf{q} \odot \mathbf{q}_\mathrm{I})^{-1} \tag{6.7}$$

where $\odot$ is the quaternion product and $\mathbf{q}^{-1}$ is the inverse of $\mathbf{q}$ [37]. As we assume that the measurements of the individual sensors are statistically independent given the state $\mathbf{x}$, the overall measurement likelihood is the product

$$p(\mathbf{z}_{\mathrm{I,c}} \mid \mathbf{x}) = p(\boldsymbol{\omega}_{\mathrm{I,c}} \mid \mathbf{x})\, p(\mathbf{a}_{\mathrm{I,c}} \mid \mathbf{x})\, p(\mathbf{m}_{\mathrm{I,c}} \mid \mathbf{x}) \tag{6.8}$$

of the individual likelihoods.

6.2 Sensor Model for Filtered Orientations

In this section, we propose a probabilistic sensor model for the orientation estimates of the on-board filter of an IMU. We assume that the IMU provides an accurate orientation estimate in a quaternion $\mathbf{z}_{\mathrm{I,f}}$. We model the measurement likelihood of the orientation estimates by assuming Gaussian noise in all three rotation axes, i.e.

$$\mathbf{z}_{\mathrm{I,f}} = \mathbf{q} \odot \mathbf{q}_\mathrm{I} \odot \tilde{\mathbf{q}}([\gamma_1, \gamma_2, \gamma_3]) \quad \text{with} \quad \gamma_i \sim \mathcal{N}\left(\gamma_i; 0, \sigma_{\mathrm{I,f}}^2\right)\ ,\ (i = 1, 2, 3) \tag{6.9}$$

where $\mathbf{q}$ is the orientation contained in the state vector $\mathbf{x}$, $\mathbf{q}_\mathrm{I}$ is the mounting orientation of the IMU with respect to $\mathcal{F}_\mathrm{b}$, and $\odot$ is the quaternion product. The function $\tilde{\mathbf{q}}(\boldsymbol{\varphi})$ represents the quaternion from the incremental rotation $\boldsymbol{\varphi} = [\varphi_x, \varphi_y, \varphi_z]^T$ around all three axes [37]. Further, $\sigma_{\mathrm{I,f}}$ is the standard deviation of the errors $\gamma_1, \gamma_2, \gamma_3$ of the orientation estimates of the IMU. Equation (6.9) can be formulated as

$$(\mathbf{q} \odot \mathbf{q}_\mathrm{I})^{-1} \odot \mathbf{z}_{\mathrm{I,f}} = \tilde{\mathbf{q}}(\gamma_1, \gamma_2, \gamma_3) \tag{6.10}$$

where $\mathbf{q}^{-1}$ is the inverse of $\mathbf{q}$ [37]. In a first-order Taylor approximation of $\tilde{\mathbf{q}}$, which assumes constant angular velocity during the incremental rotation (see Section 6.7 of [37] for details), this results in

$$(\mathbf{q} \odot \mathbf{q}_\mathrm{I})^{-1} \odot \mathbf{z}_{\mathrm{I,f}} \approx \left[1, \tfrac{1}{2}\gamma_1, \tfrac{1}{2}\gamma_2, \tfrac{1}{2}\gamma_3\right]^T \tag{6.11}$$

and finally specifies the probabilistic sensor model

$$p(\mathbf{z}_{\mathrm{I,f}} \mid \mathbf{x}) = \mathcal{N}\left([\mathbf{0}, I_{3\times3}]\left((\mathbf{q} \odot \mathbf{q}_\mathrm{I})^{-1} \odot \mathbf{z}_{\mathrm{I,f}}\right); \mathbf{0}, \tfrac{\sigma_{\mathrm{I,f}}^2}{4}\, I_{3\times3}\right)\ . \tag{6.12}$$

Here, $I_{3\times3}$ is the three-dimensional identity matrix.

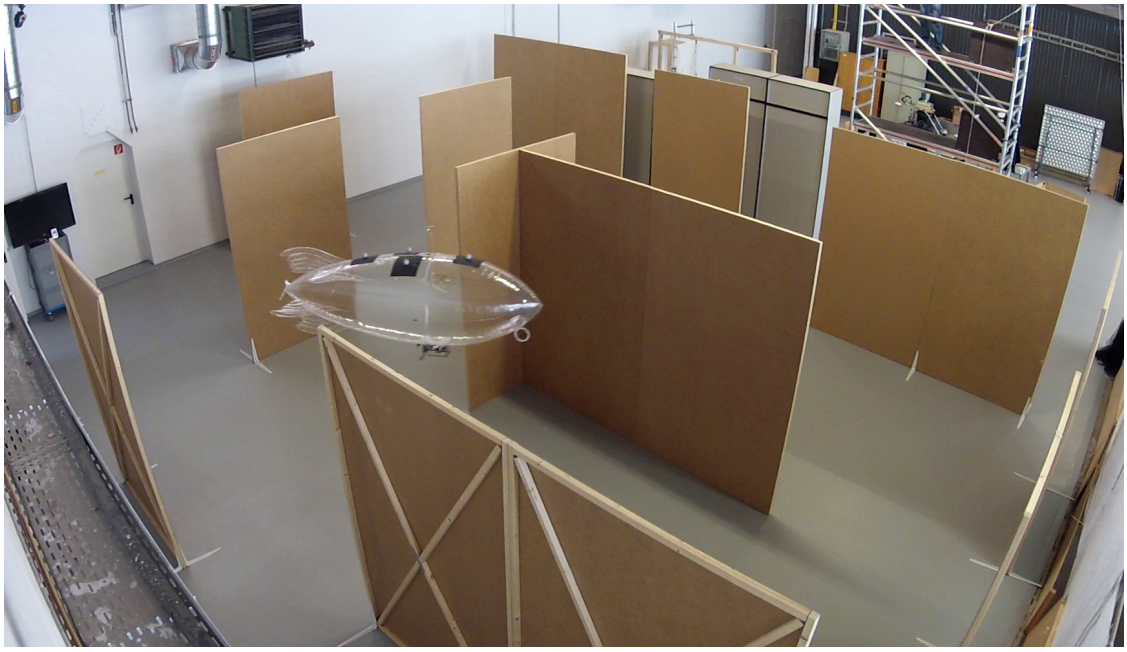


Figure 6.1: The third prototype of our miniature blimp operating in the maze-like indoor environment.

6.3 Experimental Evaluation

We evaluated and compared both sensor models in the context of indoor airship localization with our third blimp prototype (see Section 2.1.3). The blimp was equipped with five sonar sensors, three air flow sensors, and an IMU for localization. The IMU [156] weighs $9\,\mathrm{g}$ and is equipped with gyroscopes, accelerometers, and a three-axis magnetometer. The IMU sensor data is fused using an extended Kalman filter that runs on the processor integrated into the IMU and provides accurate attitude and heading estimates [118]. In addition to the orientation estimates, the IMU provides the measurement data of the individual sensor chips, which is compensated for (temperature-dependent) biases and magnetic cross-axis effects. For our experiments, we created a complex maze-like environment in a large hall. The environment had a size of approximately $10\,\mathrm{m} \times 10\,\mathrm{m}$ and is shown in Figure 6.1. It was mapped with the help of a laser range finder and represented as an OctoMap with a resolution of $10\,\mathrm{cm}$ as shown in Figure 6.2. During all experiments, we obtained accurate ground truth states of the blimp from the optical motion capture system as a reference.

To evaluate and compare the performance of both IMU sensor models, we applied them together with the sonar sensor model and the air flow sensor model in the particle filter localization (see Section 3.3). In our implementations of the particle filter, we determined the set of objects relevant for the sonar measurements by ray-casting using a fixed angular resolution of 3° and stored a precalculation of these objects for efficient

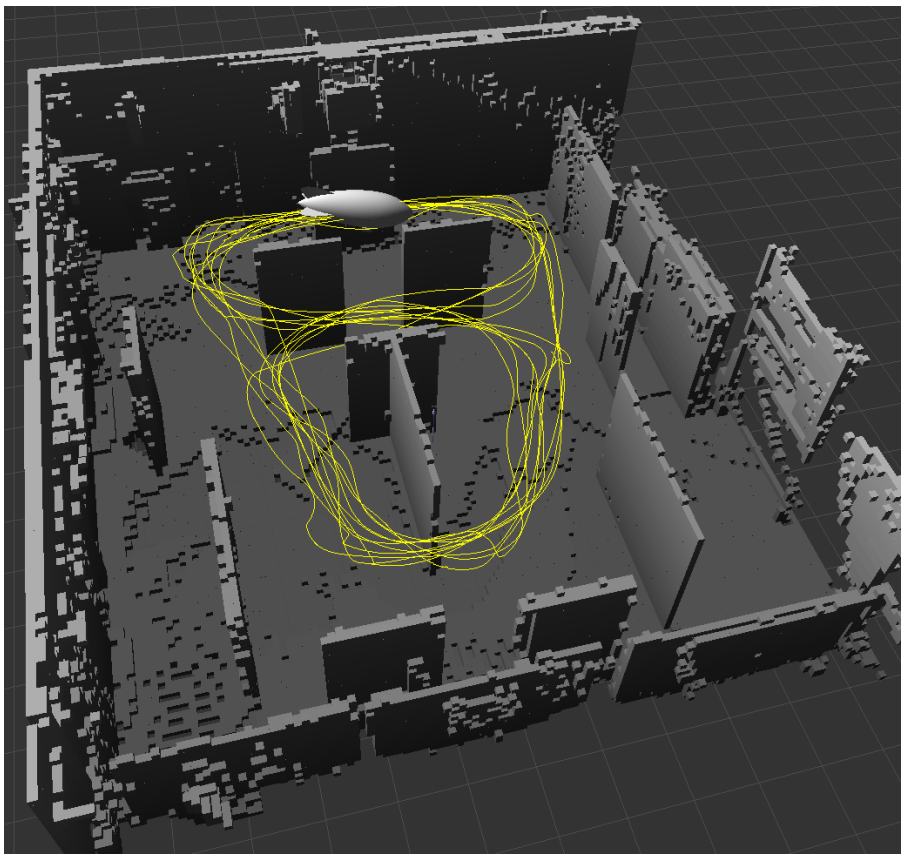


Figure 6.2: The OctoMap representation of the indoor environment in which the blimp was manually operated during the experiments. The trajectory of the blimp has a length of 276.6 m and is depicted in yellow. The blimp itself is shown at its final position on the trajectory.

lookups of the measurement likelihood. In the air flow sensor model, we chose the bandwidth $l = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$ and precalculated the approximation with a grid resolution of $0.02\,\frac{\mathrm{m}}{\mathrm{s}}$. We determined the standard deviations of the IMU measurement noise in a statistical identification by comparing the measurements and filter estimates to the ground truth data. In order to ensure the conditional independence of the measurements, we integrated a sonar measurement only every $0.2\,\mathrm{s}$ and an air flow measurement every $1\,\mathrm{s}$ into the belief of the filter. Additionally, we applied a convolution with a Gaussian kernel with a standard deviation of $0.1\,\mathrm{m}$ to the measurement likelihood function of the sonar sensors. This facilitated a reduction of the minimum-required density of the particles and therefore reduced the number of particles needed for successful localization.

In the preparation of the experiment, we learned the parameters of all models from the motion capture reference trajectory together with the real sensor and control data recorded during manually controlled flight experiments. Neither during the model training nor during the localization experiments, there were any dynamic, unmapped obstacles disturbing the measurements.

In our experiment, we manually controlled the blimp for $10\,\mathrm{min}$ through the maze-like environment. The collision-free trajectory had a length of $276.6\,\mathrm{m}$ and is depicted in Figure 6.2. We evaluated and compared the localization methods on the sensor and control data recorded during operation. As a measure of the localization error, we used the Euclidean distance between the ground truth position and the position estimate of the particle filter, which is calculated as the weighted average of all particles. We considered a localization run as successful, if the position estimate of the filter never deviated by more than $2\,\mathrm{m}$ from the ground truth position during the whole run (see Section 8.4 for details on this threshold). For each successful localization run, we measured the root mean square (RMS) of the three-dimensional position error over the whole trajectory.

We evaluated the localization results of two different particle filter implementations:

- *Compensated*: The particle filter implementation with the IMU sensor model on compensated measurement data.
- *Filtered*: The particle filter implementation with the IMU sensor model on filtered orientations.

Both implementations additionally applied the sonar sensor model described in Chapter 4 and the air flow sensor model introduced in Chapter 5.

Figure 6.3 and 6.4 show a statistical evaluation of the localization experiments. Since the *Compensated* model also provides information about the translational acceleration, it can achieve a higher localization accuracy than the *Filtered* model for a sufficiently high number of particles. However, the *Compensated* implementation estimates the translational acceleration and the rotational velocity of the airship in the particle filter and therefore demands a high number of particles to cover the high-dimensional state space.

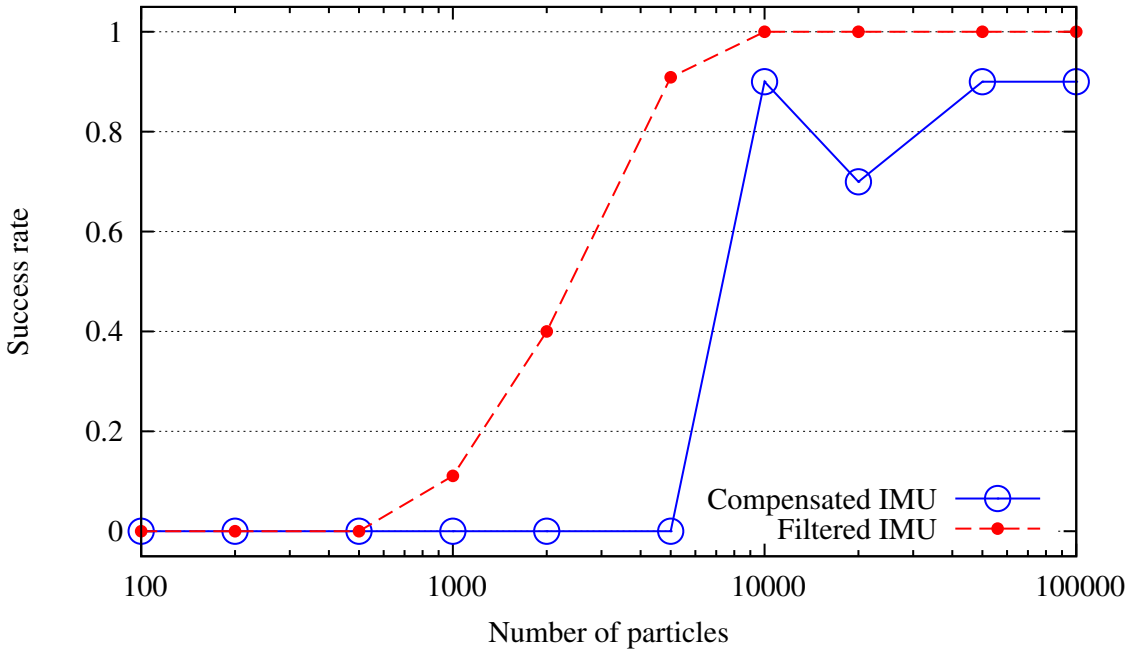


Figure 6.3: The success rate of the individual localization approaches.

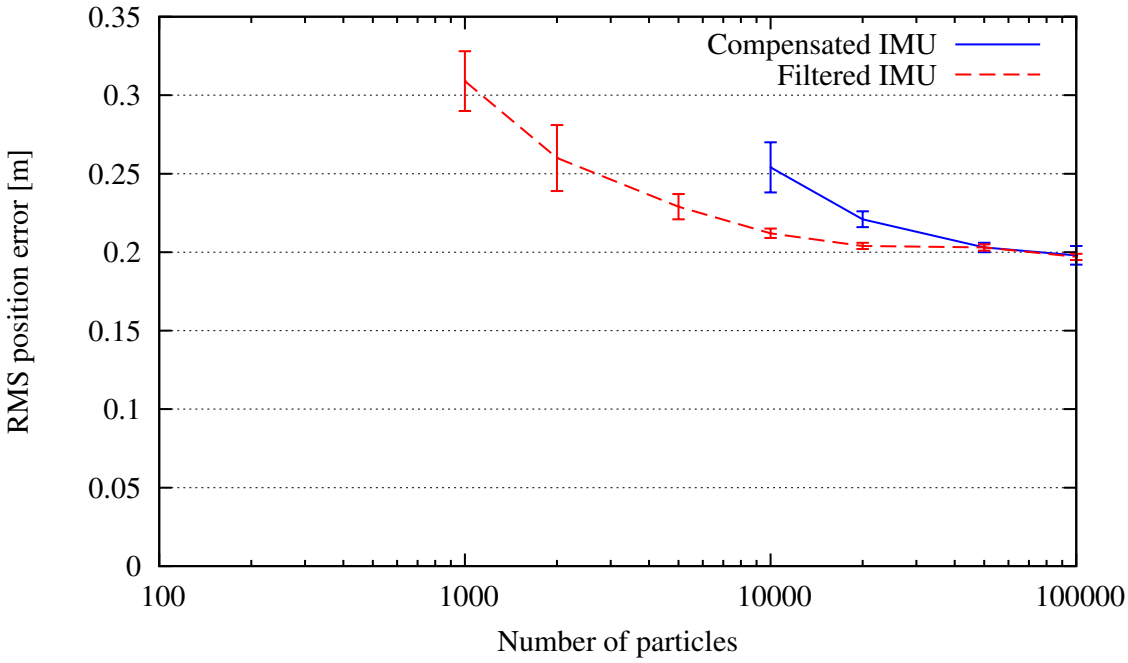


Figure 6.4: The average RMS localization error of the individual localization approaches. The error bars indicate the 95 % confidence intervals over ten successful runs.

In contrast, the *Filtered* model relies on the preprocessed orientation estimates of the on-board EKF of the IMU, which partially shifts uncertainty from the particle filter to the more efficient EKF in the two-stage filter design. On the one hand, this results in a better performance of the *Filtered* model when only a small number of particles can be used. On the other hand, the *Filtered* model is an approximation, as the on-board EKF only relies on IMU measurement data. This is a slight drawback compared to the *Compensated* implementation in which the IMU measurements are fused with all available information – including the orientation, velocity, and acceleration estimates of the motion model. However, the *Filtered* model outperformed the *Compensated* model for reasonable numbers of particles in the context of blimp localization.

6.4 Conclusions

In this chapter, we introduced probabilistic sensor models for two variants of IMU measurements. Both models take into account the uncertainties of the input data and provide a Gaussian sensor model, which enables an efficient integration of sensor data into Bayes filter implementations. We compared both models in an experiment with a real blimp in the context of localization in an indoor environment. Although the filtered model is an approximation, it significantly outperforms a standard model for compensated IMU measurements in case of a small number of particles. Throughout this work, we aim at developing an efficient approach to online blimp localization for autonomous navigation. This requires a localization approach that can deal with the Monte Carlo approximation with a limited number of particles in the particle filter. Hence, we prefer to use the sensor model for filtered orientations over the sensor model for compensated IMU measurement data in the remaining parts of this work.

Chapter 7

Simultaneous Localization and Estimation of Motion Model Parameters

With regard to self-localization, flying robots have several limitations compared to ground vehicles. Due to their limited payload, flying vehicles are restricted to a few lightweight sensors. Additionally, the kinematics of flying robots requires sophisticated motion models, which are typically hard to calibrate. However, since the sensors provide only a limited amount of information, the motion models need to be highly accurate to reduce the potential increase of uncertainty caused by the movements of the vehicle. In this chapter, we present a novel approach to simultaneous localization and estimation of motion model parameters and their adaption in the context of a particle filter. To deal with sudden changes of parameters, our approach utilizes random sampling augmented by additional damping to avoid oscillations caused by the delayed detection of the changes. As we demonstrate in experiments with a real blimp, our method can deal with very sparse and imprecise sensor information and outperforms a standard Monte Carlo localization approach.

• • • • •

Recently, UAVs got smaller and are therefore applicable even in autonomous navigation tasks in narrow indoor environments. In the previous chapter, we showed how one can accurately localize a miniature indoor blimp that is equipped with sonar and air flow sensors as well as with an IMU. The smaller flying robots get, however, the less sensor information is available due to their strictly limited number of lightweight and imprecise

sensors. This increases the importance of the prediction model of the Bayes filter localization, which in our case is called the motion model of the robot. Most ground vehicles are equipped with wheel encoders and can sense their motion relative to the ground in a fairly accurate way. Motion models of aerial robots, however, in general cannot rely on direct measurements of the velocity and need to estimate accelerations due to thrust and air drag. Consequently, the motion models are based on the complex kinematics of the vehicle, which can be modeled by physical approximations and depends on numerous parameters. In practice, these parameters are usually hand-tuned by a human or derived from calibration experiments using expert knowledge or ground truth pose estimates.

In this chapter, we consider the problem of localizing a small-sized blimp in indoor environments using Monte Carlo localization with very sparse sensor information. Our first blimp prototype has an effective payload of $100\,\mathrm{g}$ and is equipped with four miniature sonar sensors. Due to their huge opening angle, the sonar sensors provide only little information about the orientation of the blimp. To cope with this lack of sensor information, we improve the proposal distribution by simultaneously estimating the uncertain parameters of the motion model. Our approach enables an online localization and has a couple of beneficial properties. It does not rely on a previous calibration of the motion model parameters and can adapt to changing parameters during operation. Our method explicitly includes a damping to prevent an overshooting of the parameters for situations in which parameter changes can only be detected with certain delays. We envision a wide range of applications of our approach including a non-constant payload of the blimp, deformations of the blimp due to collisions, or even the application on ground robots, to deal with, for example, ground-dependent wheel slippage. Furthermore, the estimated parameters, in combination with the underlying physical motion model, can be used for an online adaption of motion controllers [142, 183] in autonomous navigation scenarios.

In the following, we introduce our approach to simultaneous Monte Carlo localization and parameter estimation and propose an approach to deal with changed parameters even if they are detected with a certain delay. In experiments in simulation and with a real robotic blimp, we demonstrate the performance of our approach compared to standard Monte Carlo localization with previously calibrated parameters.

7.1 Simultaneous Monte Carlo Localization and Parameter Estimation

In mobile robot localization, parametric models are often applied to predict the motion of the robot [55, 88, 183]. Let $\boldsymbol{\Theta}$ be the parameter vector containing unknown parameters of the motion model. In the following, we derive how unknown (and probably time-varying) parameters of the motion of the robot can be taken into account in the recursive

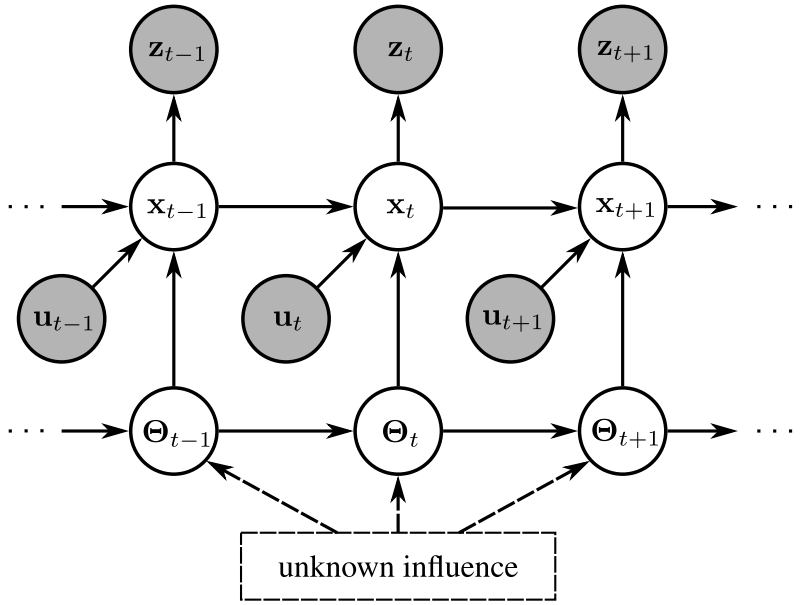


Figure 7.1: The extended dynamic Bayes network for localization of a mobile robot. It characterizes the evolution of the states $\mathbf{x}$, the controls $\mathbf{u}$, the measurements $\mathbf{z}$, and the parameters of the motion model $\boldsymbol{\Theta}$.

probabilistic state estimation in the particle filter, which was introduced in Section 3.2. In the presence of such unknown parameters, the underlying Bayes network is extended as depicted in Figure 7.1. Here, the hidden variables are the states and the additional, non-observable parameter nodes. Consequently, the full localization posterior given in Equation (3.5) is extended to $p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$, which can be factorized to

$$\begin{aligned} p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \, & p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \boldsymbol{\Theta}_{1:t}, \mathbf{u}_{1:t}) \\ & \cdot p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \end{aligned} \tag{7.1}$$

using Bayes rule where η is a normalizer. We factorize the second conditional probability twice and obtain

$$\begin{aligned} p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) &= p(\mathbf{x}_t, \boldsymbol{\Theta}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \qquad (7.2) \\ &= p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \\ &\quad \cdot p(\boldsymbol{\Theta}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \, . \qquad (7.3) \end{aligned}$$

Under the Markov assumption, Equation (7.1) together with Equation (7.3) can be simplified to

$$\begin{aligned} p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \, & p(\mathbf{z}_t \mid \mathbf{x}_t) \\ & \cdot p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\Theta}_t, \mathbf{u}_t) \\ & \cdot p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t-1}, \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \\ & \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; . \end{aligned} \tag{7.4}$$

Here, we assume that future controls give no information about the current state of the robot, which is a standard assumption that is elaborately justified by Thrun et al. [169].

To implement this recursive filtering scheme, we use a particle filter in which a set $\mathcal{M}_t$ of weighted particles represents the belief at time t. Each particle represents a hypothesis of a robot state and parameter vector. As proposal distribution, we use the motion model combined with the model of the parameter behavior

$$\begin{aligned} \pi(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \; & p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\Theta}_t, \mathbf{u}_t) \\ & \cdot p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t-1}, \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \\ & \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\Theta}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; , \end{aligned} \tag{7.5}$$

which results in the importance weight

$$\begin{aligned} w_t^{[i]} &= \frac{p(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{\pi(\mathbf{x}_{1:t}, \boldsymbol{\Theta}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})} \\ &\propto p(\mathbf{z}_t \mid \mathbf{x}_t) \end{aligned} \tag{7.6}$$

of the i-th particle at time t.

Assuming the parameters to be constant over time, the belief update (7.4) can be performed according to the following three alternating steps:

1. In the *prediction step*, for each particle, we draw a new particle according to the parameterized motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\Theta}_t, \mathbf{u}_t)$ given the action $\mathbf{u}_t$.

2. In the *correction step*, we integrate a new observation $\mathbf{z}_t$ by assigning a new weight $w_t^{[i]}$ to each particle according to the sensor model $p(\mathbf{z}_t \mid \mathbf{x}_t)$.

3. In the *resampling step*, we draw a new generation of particles from $\mathcal{M}_t$ (with replacement) so that each sample in $\mathcal{M}_t$ is selected with a probability that is proportional to its weight.

Following Liu and West [105], we reduce the sample attrition by adding a small noise to the parameter vector of each sample during resampling. To prevent a loss of information in the parameter vector sampling, we apply kernel smoothing

$$\boldsymbol{\Theta}_t^{[i]} \sim \mathcal{N}(\boldsymbol{\Theta}_t^{[i]}; a\boldsymbol{\Theta}_{t-1}^{[i]} + (1-a)\boldsymbol{\mu}_{\boldsymbol{\Theta}_{t-1}}, h^2 \Sigma_{\boldsymbol{\Theta}_{t-1}}) \tag{7.7}$$

where $\boldsymbol{\mu}_{\boldsymbol{\Theta}_{t-1}}$ and $\Sigma_{\boldsymbol{\Theta}_{t-1}}$ are the mean and the covariance of the parameter vector over the particle set at time $t-1$. The constant factors $a = \frac{3\gamma-1}{2\gamma}$ and $h^2 = 1-a^2$ only depend on a discount factor γ, which we set to 0.95.

7.2 Adaption to Changed Parameters

In certain cases, we cannot assume the physical properties of the motion of the robot to be constant for the complete period of operation. For example, wear and tear, changed payload, collisions, low batteries, or even manual mounting of banner ads can change the behavior of the robot. Once the parameter vector has converged within the particle filter, an adaption to one or more changed parameters would take a large number of sampling steps or could simply be impossible.

Fortunately, this problem can be solved in a similar way as the well-known "kidnapped robot" problem by sampling an appropriate number of particles at random states [169]. We analogously cope with parameter changes by drawing the parameter vector $\boldsymbol{\Theta}$ uniformly from the parameter space for those random samples. The proportion of uniform samples is determined by monitoring the likelihood of the sensor measurements

$$\overline{p}_t = \frac{1}{N} \sum_{i=1}^{N} p(\mathbf{z}_t \mid \mathbf{x}_t^{[i]}) \tag{7.8}$$

averaged over all particles. We adopt the technique of Gutmann and Fox [63] and extend the resampling by setting the parameter vector of each particle to a uniform sample with probability

$$\max\left(0, 1 - \frac{p_{\text{short}}}{\nu\, p_{\text{long}}}\right) . \tag{7.9}$$

Here, p_{short} and p_{long} are short-term and long-term averages of the sensor likelihood updated by

$$p_{\text{short},t} = (1 - \alpha_{\text{short}})\, p_{\text{short},t-1} + \alpha_{\text{short}}\, \overline{p}_t \tag{7.10}$$

$$p_{\text{long},t} = (1 - \alpha_{\text{long}})\, p_{\text{long},t-1} + \alpha_{\text{long}}\, \overline{p}_t \tag{7.11}$$

in each correction step with the exponential decay factors $0 < \alpha_{\text{long}} \ll \alpha_{\text{short}} \leq 1$. The parameter ν allows for adjusting the level at which random samples are added. Thus, this approach only adds random samples if the short-term average of the sensor likelihood is less than ν times the long-term average.

However, the sole addition of random parameter samples does not yield superior results. This is due to the fact that the adaption to changed parameters does not start until the pose estimate is slightly displaced from the ground truth caused by wrong parameters

and the average observation likelihood is dropped. When random parameter samples are added, those particles that quickly correct the displacement will get a higher observation likelihood. This leads to an overshooting of the estimated parameter values and causes the parameter values to oscillate for several time steps (see Figure 7.4). We address this oscillation problem by defining a lower bound of the covariance $h^2\Sigma_{\boldsymbol{\Theta}}$ so that the parameter vectors are sampled according to

$$\boldsymbol{\Theta}_t^{[i]} \sim \mathcal{N}(\boldsymbol{\Theta}_t^{[i]}; a\boldsymbol{\Theta}_{t-1}^{[i]} + (1-a)\boldsymbol{\mu}_{\boldsymbol{\Theta}_{t-1}}, h^2\, \widetilde{\max}(\Sigma_{\boldsymbol{\Theta}_{t-1}}, \rho\, \boldsymbol{\mu}_{\boldsymbol{\Theta}_{t-1}}\, \boldsymbol{\mu}_{\boldsymbol{\Theta}_{t-1}}^T)) \tag{7.12}$$

where the $\widetilde{\max}$-operator builds a pointwise maximum over the diagonal elements of the matrices and takes all other elements from the first argument. Here, ρ is the relative covariance bound. This damps the oscillation and enables a lower localization error after the change of parameters. Furthermore, this approach better handles very slow changes of parameters, which are typically not detected through a sudden drop in the observation likelihood.

Combining these techniques enables an autonomous robot to simultaneously localize and estimate previously unknown parameters of the motion model in an online fashion and to adapt to changed parameters during operation, which is especially important in case of sparse or imprecise sensor information.

7.3 Experimental Evaluation

We implemented and evaluated our novel approach to simultaneous Monte Carlo localization and parameter estimation using simulation experiments and data acquired with a real indoor blimp. In particular, we extended the particle filter described in Section 3.3 by the techniques described above. To measure the localization error, we used the Euclidean distance between the weighted average of all particles and the ground truth pose.

Since miniature airships are typically not equipped with sensors such as wheel encoders for directly measuring their motion, it has to be estimated based on forces and torques acting on them. Except for the air drag forces, all constituent parts can be determined in a straightforward way [183]. We approximate the drag force and torque of the hull in an uncoupled way in our motion model introduced in Section 2.6. The air drag model is parameterized on the translational drag coefficients D_1, D_2 along the x and y/z axis, respectively, as well as on the rotational drag coefficient D' along the yaw axis. In addition, the drag force of the fins depends on the fin drag coefficient D_f. Consequently, we aim to estimate the parameter vector $\boldsymbol{\Theta} = [D_1, D_2, D', D_\mathrm{f}]^T$ for our blimp during localization.

7.3.1 Experiments with a real blimp

In our experiments with our first blimp prototype (see Section 2.1.1), we used the same experimental setup as the one described in Section 4.3. The blimp was equipped only with four small, lightweight wide-angle sonar sensors, which provide distance measurements to the environment. The environment had a size of $14\,\text{m} \times 7\,\text{m}$ and we obtained ground truth positions from the camera integrated in the gondola and optical markers placed on the floor.

In an experiment of about $9\,\text{min}$ of manually operated flight, the blimp covered a distance of about $180\,\text{m}$. Since we did not use an IMU and the wide-angle sonar sensors provide hardly any information about the orientation of the blimp, the localization relied on an accurate prediction of the motion model. As can be seen in Figure 7.2, the simultaneous localization and estimation of initially unknown parameters substantially benefits from the improved proposal distribution. It resulted in a significantly lower rotational localization error compared to the results obtained using the implementation with previously learned (see Section 2.6) and fixed parameters of the motion model. Note that the previously estimated parameters are not used for our proposed simultaneous localization and parameter estimation. During localization, the estimate of the parameter vector typically converged within the first minute of the experiment. Although the dimensionality of the state estimation problem was increased by the four parameters, the localization success rate revealed that the number of particles required to reliably localize the blimp is substantially smaller when using the simultaneous parameter estimation.

Furthermore, we tested our localization system with simultaneous parameter estimation in a global localization scenario with initially unknown pose and motion model parameters. We carried out 10 runs with $5{,}000$ particles each. While the pose estimate typically converged within the first $10\,\text{s}$, it still took $1\,\text{min}$ to determine the parameters.

7.3.2 Simulation of Changing Parameters

To evaluate the capability of our localization system to adapt to changing parameters of the motion model, we performed a series of experiments on the simulated blimp described in Section 2.6.4. The true poses were passed as observations to the localization system and their likelihoods were modeled as a Gaussian distribution with high translational and low rotational precision ($\sigma_{\text{trans}} = 0.2\,\text{m}$, $\sigma_{\text{rot}} = 15°$).

In an experiment of about $10\,\text{min}$ of manually operated flight, we evaluated different relative covariance bounds ρ in our localization system using $5{,}000$ particles. Note that we experimentally tuned the parameters $\alpha_{\text{short}} = 0.2$, $\alpha_{\text{long}} = 0.01$, and $\nu = 0.8$ to obtain best results. Although we introduced only little noise to the velocity of the blimp during simulation, the parameter estimation during localization was a challenging task due to the correlation of the different parameters (see Figure 7.4). Furthermore, estimating all

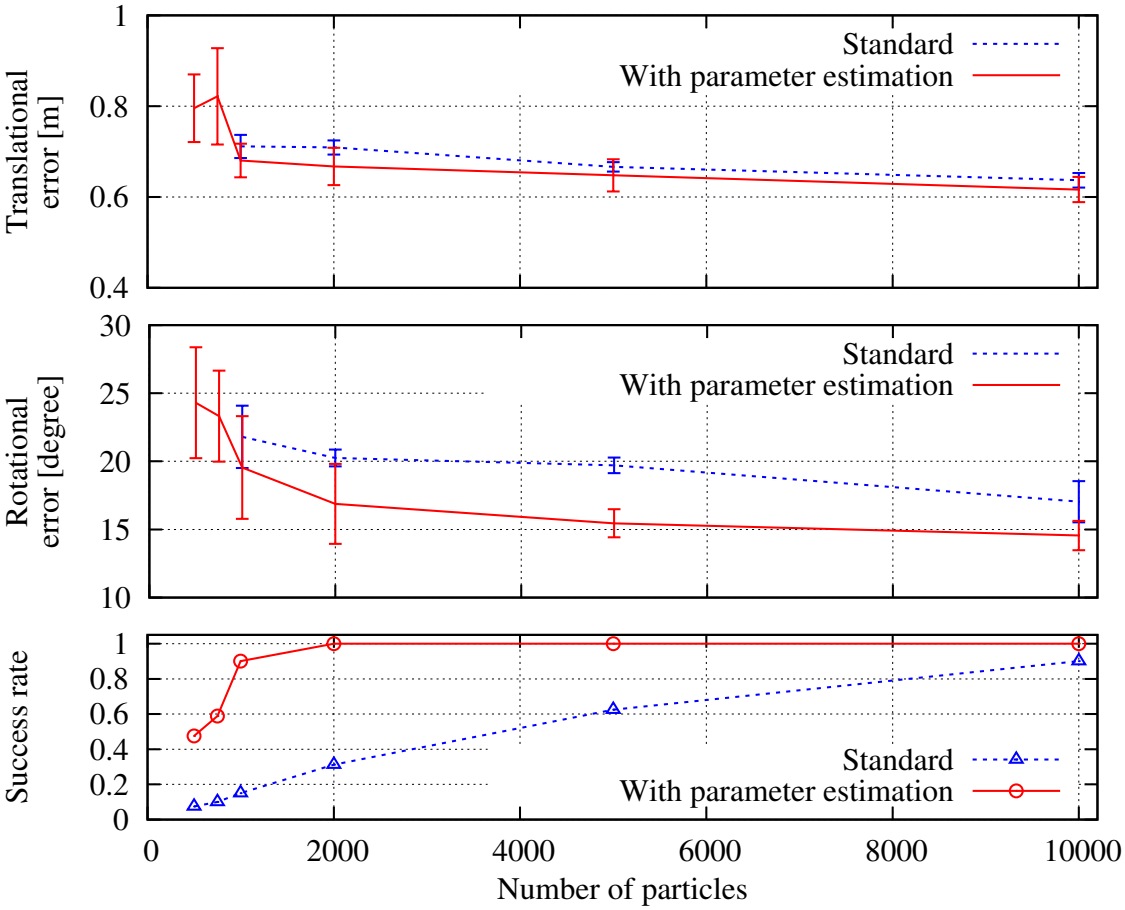


Figure 7.2: The average RMS localization error and the success rate of the simultaneous Monte Carlo localization and parameter estimation compared to the standard localization with previously learned parameters of the motion model. The error bars indicate the 95 % confidence intervals over ten runs.

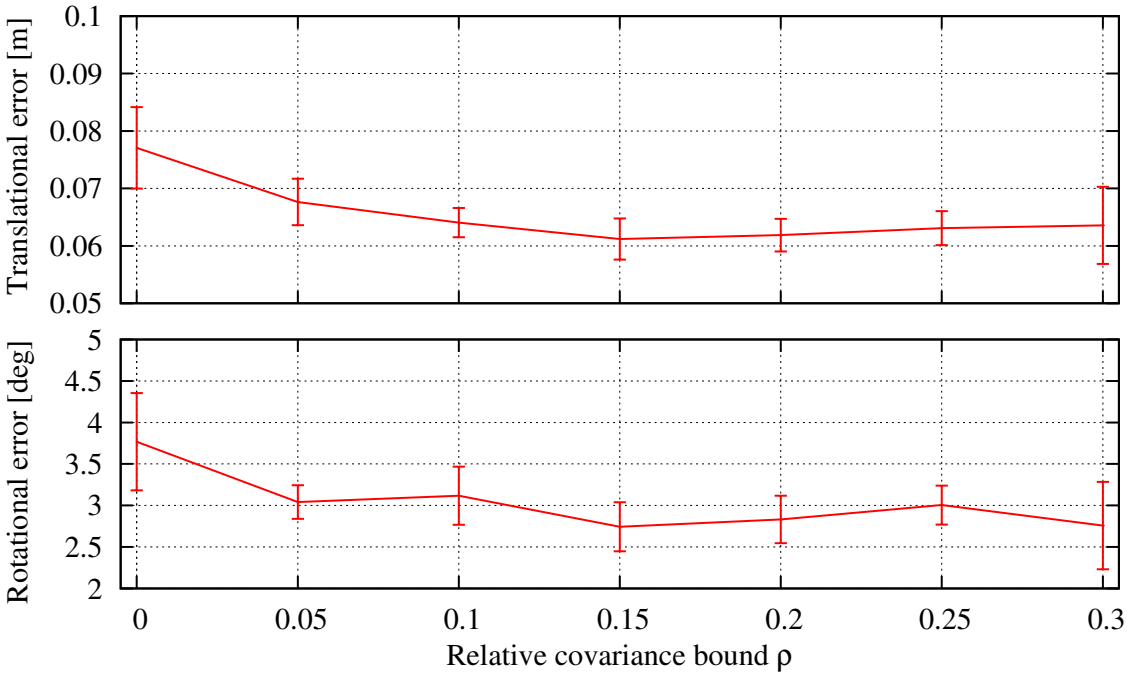


Figure 7.3: The average translational and rotational RMS localization error of a 10 min localization experiment on simulated data with changing parameters of the motion model. The error bars indicate the 95 % confidence intervals over ten runs.

parameters requires a certain spectrum of movements. For example, during the first seconds of our experiment, the blimp was controlled to move only forward, which resulted in a quick convergence of solely the drag coefficient D_x.

Figure 7.3 presents the localization accuracy on the simulated data during which the parameters of the motion model were changed as depicted in Figure 7.4. Both the translational and the rotational localization error are significantly lower for a relative covariance bound of $\rho > 0.15$ than without a lower bound ($\rho = 0$) of the covariance of the parameters. Figures 7.4 and 7.5 show the comparison of the simultaneous localization and parameter estimation with and without a lower bound on the covariance. Although most changes in the parameters are detected with a delay in both variants, the covariance bounding effectively inhibits the oscillation of the parameter estimates and enables the adaption even to slight changes in the parameters.

7.4 Related Work

In the past, several authors considered the problem of localizing small flying vehicles. The majority of approaches, however, employed previously learned motion models. For example, Ko et al. [88] used the ground truth estimates of a motion capture system for

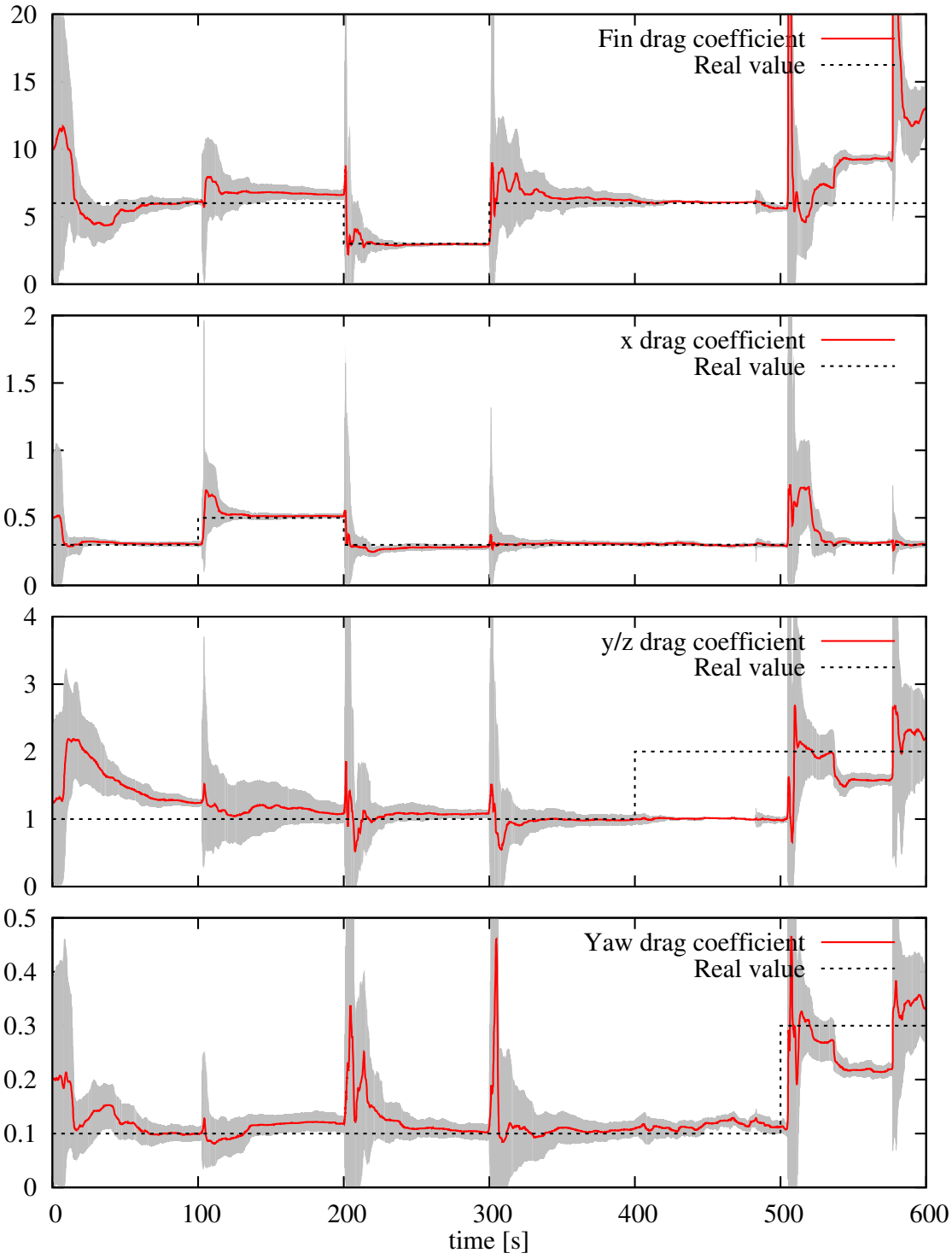


Figure 7.4: The parameter estimation results in an exemplary localization run of the simulated blimp without covariance bounding. The area between the 2.5 % and 97.5 % quantiles of the estimated posterior is marked gray.

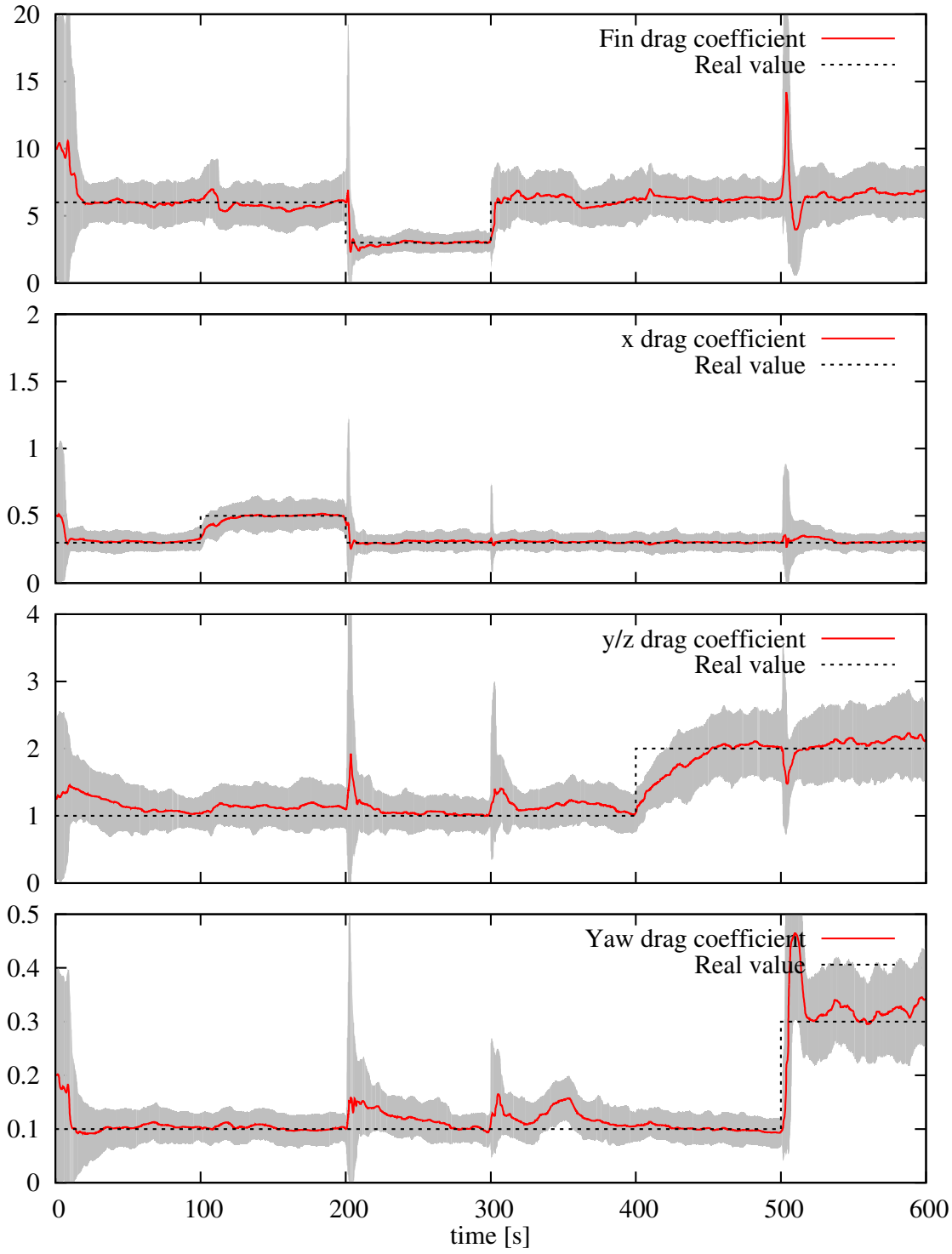


Figure 7.5: The parameter estimation results in an exemplary localization run of the simulated blimp with a covariance bounding of $\rho = 0.2$. The area between the 2.5 % and 97.5 % quantiles of the estimated posterior is marked gray.

tuning the motion model. Our work presented in the previous chapters relies on ground truth poses and uses additional IMU sensor information for localization. Acquiring the ground truth data involves either expensive and bulky systems or lacks sufficient precision to infer highly accurate motion models.

Several approaches have been proposed to improve the proposal distribution in Monte Carlo localization using information from sensors other than wheel encoders or control commands. For example, Thrun et al. [167] sampled additional "dual" particles from the observation likelihood to improve the robustness of the system and to better recover from localization failures. Although this approach is very effective, sampling from the observation likelihood is computationally demanding for range sensors such as sonar or laser range finders. Consequently, sampling from the observation model is mainly used for vision-based localization. Grisetti et al. [58] matched laser range scans to improve proposals for wheeled robots operating on planar ground. Later, a similar approach was applied together with IMU data to localize miniature quadrotors operating in six degrees of freedom [60, 67]. However, it is unclear whether small and lightweight sensors, such as three miniature wide-angle sonar sensors, allow for such a scan matching approach.

One of the first systems for Monte Carlo localization with online calibration of the motion model was developed by Roy and Thrun [144]. They incrementally updated the calibration parameters for differential drive robots based on a maximum likelihood position estimate obtained by scan matching. However, their approach relies on the direct calculation of parameters out of two consecutive pose estimates, which is not possible for more sophisticated motion models such as those for blimps. The case of sudden changes of the models or their parameters due to failures or collisions of wheeled robots was addressed by Plagemann and Burgard [129, 130]. They extended the particle filter localization by motion models of different complexity and used a parameter sampling similar to our approach. However, they assumed that the model and its parameters are initially known and switched online between a finite number of models.

Beside Monte Carlo localization, Kalman filters are a popular technique for mobile robot localization. For example, Larsen et al. [99] and Martinelli et al. [108] extended the state vector by additional parameters for odometry errors. However, motion models of flying vehicles are typically nonlinear and our wide-angle sonar sensors are not suitable to be used within a Kalman filter. In the context of localization of UAVs, Bryson and Sukkarieh [19] estimated the difference between IMU and motion model prediction in an additional extended Kalman filter to update the model parameters and the IMU bias. In contrast to our approach, this requires to update the model parameters directly based on the prediction error similar to Roy and Thrun [144].

Some approaches utilize the expectation maximization algorithm to simultaneously localize a robot and adapt its motion model. For example, Eliazar and Parr [43] and Yap and Shelton [182] iterated between estimating the path of the robot and optimizing the

parameters of the motion model. However, these approaches are not intended for online applications and require to quickly optimize the parameters of the motion model given a trajectory of the robot, which is not the case, e.g. for more complex motion models of flying robots such as blimps. Kaboli et al. [76] used the Markov Chain Monte Carlo technique to learn sensor and motion model parameters from raw sensor and control data by sampling trajectories and parameters, which typically requires substantial computational resources.

In contrast to these previous approaches, our approach provides an online estimation and adaption of previously unknown parameters of a complex and potentially nonlinear motion model in the context of localization with particle filters. Furthermore, compared to multiple model tracking systems, our approach can better deal with very slow changes in the continuous parameter space.

7.5 Conclusions

In this chapter, we presented a novel approach to Monte Carlo localization of autonomous robots with simultaneous estimation of the parameters of the motion model. In contrast to other approaches, our technique enables an online localization without prior knowledge of all motion model parameters and can adapt to changed parameters during operation. To avoid oscillations after parameter changes on systems for which these changes can only be detected with a certain delay, our approach includes an appropriate damping mechanism. The techniques presented in this chapter can be applied to state estimation for general systems with unknown and possibly changing parameters of the system dynamics. In experiments carried out with a real blimp and in simulation, we demonstrated that our system significantly outperforms standard Monte Carlo localization with previously learned parameters in terms of the localization accuracy and the number of particles needed.

Chapter 8

Odometry Motion Model

One advantage of miniature airships is their ability to move safely and to hover for extended periods of time. At the same time, they are challenging, as their complex second-order kinematics makes the prediction of their pose and velocity through physical simulation difficult and imprecise. In this chapter, we consider the problem of particle filter-based online localization for a miniature blimp with lightweight ultrasound and air flow sensors as well as an IMU. We introduce an efficient odometry motion model, which is based on the measurements of air flow sensors and an IMU and which is less computationally demanding compared to the standard physical simulation-based control motion model. In experiments with a real blimp, our approach has proven to allow accurate and reliable online localization of a miniature blimp and requires an order of magnitude fewer particles compared to the localization based on the standard control motion model. Furthermore, we demonstrate the substantial improvements in terms of localization accuracy when taking into account the temporal correlations of the air flow measurements in our novel odometry motion model.

• • • • •

When used as autonomous mobile systems for indoor navigation, miniature airships require the ability to accurately estimate their position. Since miniature airships can usually carry only a limited number of lightweight sensors, their localization algorithms rely on an accurate prediction of the motion of the robot. However, motion prediction of airships is typically based on a physical simulation of the forces and torques acting on the vehicle, e.g. thrust and drag. This prediction is computationally expensive and induces a high amount of uncertainty.

In this chapter, we consider the problem of online localization for a miniature blimp in indoor environments through probabilistic state estimation with a particle filter. The

blimp is equipped with five tiny sonar sensors, three air flow sensors, and an IMU. We therefore introduce an efficient odometry motion model, which combines the measurements of multiple air flow sensors and an IMU for accurate probabilistic motion prediction in the particle filter. In contrast to the standard simulation-based control motion model, our odometry motion model requires an order of magnitude fewer particles for localization and is less computationally demanding because it does not rely on complex physical simulations. Therefore, it enables an accurate online localization. We present the results obtained in experiments carried out with a real robotic blimp in a complex indoor environment. The results demonstrate that our approach to online localization outperforms the standard particle filter localization based on the control motion model applied in many state-of-the-art autonomous blimp navigation systems [55, 88, 183].

8.1 Odometry for Miniature Airships

For ground robots, odometry motion models are often applied for localization. These models integrate the measurements of wheel rotations, which is typically considerably more accurate than predicting and integrating the motor accelerations based on the control commands [35, 103, 114]. A similar technique has been successfully applied with optical flow on various kinds of robots [29, 38, 85, 179]. Unfortunately, most robotic airships are not equipped with appropriate motion sensors. They therefore have to rely on a physical motion simulation based on controls, which suffers from large errors [88], even when adaptive motion models are used. In this section, we transfer the idea of using odometry measurements together with probabilistic motion models to flying vehicles equipped with an IMU and three or more air flow sensors. Thereby, we explicitly consider the measurement uncertainty of the odometry sensors when drawing particles from the odometry motion model used as the proposal distribution in the particle filter.

Figure 8.1 shows the particle filter localization process with our IMU and air flow odometry motion model. When using the odometry motion model, which is based on a first-order differential equation, the state vector of the particle filter can be reduced to the pose $\mathbf{x} = \left[\mathbf{p}^T, \mathbf{q}^T\right]^T$, which drastically decreases the dimensionality of the filtered state from twelve to six. In the remainder of this section, we assume that $\tilde{\mathbf{u}}$ represents the odometry measurements, which is a common practice in the context of odometry motion models for wheeled robots [169]. In our case, the odometry measurements

$$\tilde{\mathbf{u}}_t = z_{\mathrm{F},1}, \ldots, z_{\mathrm{F},L}, \mathbf{z}_{\mathrm{I}} \tag{8.1}$$

are those from all air flow sensors and the IMU. In our efficient implementation of the particle filter with the odometry motion model, the proposal distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \tilde{\mathbf{u}}_t)$ is conditioned on this odometry measurement vector. Consequently, the measurement

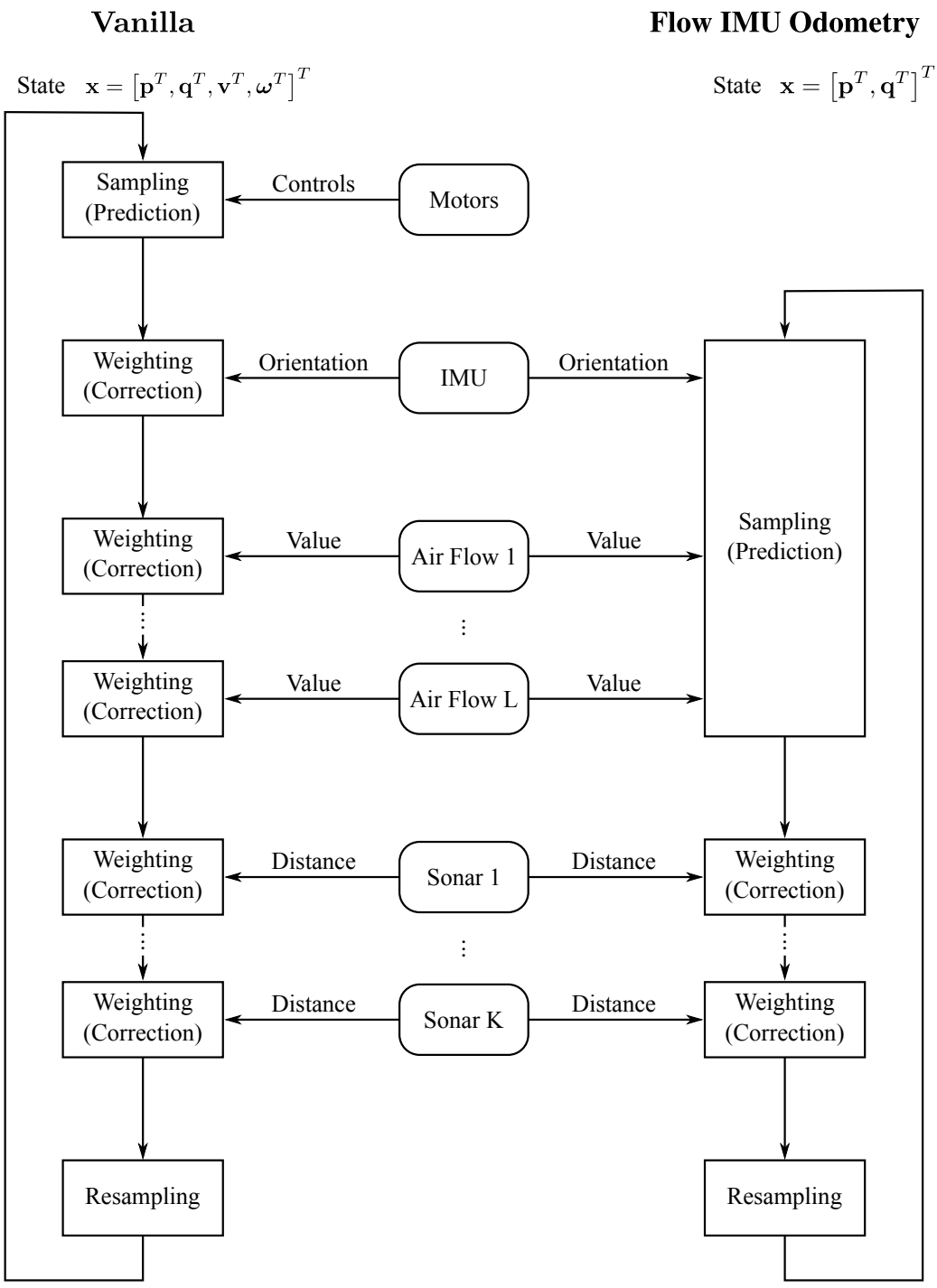


Figure 8.1: The process of the particle filter localization for a mobile robot equipped with an IMU, L air flow sensors, and K sonar sensors. The left side shows the individual processing steps of the set of particles in the vanilla implementation introduced in Section 3.3. The actuator and sensor devices providing control and measurement data are shown in the middle. The right side shows the individual processing steps of the set of particles with our novel IMU and air flow odometry motion model.

likelihood

$$p(z_{\mathrm{S},1}, \ldots, z_{\mathrm{S},\ell} \mid \mathbf{x}) = \prod_{i=1}^{\ell} p(z_{\mathrm{S},i} \mid \mathbf{x}) \tag{8.2}$$

used for weighting the particles in the correction step of the particle filter takes into account the remaining sensor data, which here are the sonar measurements. Hence, throughout this chapter, we assume that $\mathbf{z}$ represents the measurements of all sonar sensors.

8.2 IMU and Air Flow Sensor Odometry Motion Model

For dead-reckoning odometry of flying vehicles, the full six-dimensional velocity consisting of $\mathbf{v}$ and $\boldsymbol{\omega}$ is required. While the rotational part $\boldsymbol{\omega}$ is directly measured by the IMU, the translational part $\mathbf{v}$ cannot be obtained in a straightforward way for flying vehicles. However, in indoor scenarios, where the air is assumed to be static, each air flow sensor can measure its translational velocity along its measurement axis. In the following, we assume the vehicle to be equipped with an IMU and $L \geq 3$ air flow sensors whose measurement axes are linearly independent. This is the minimum requirement for estimating the three-dimensional translational velocity $\mathbf{v}$. Furthermore, the number of air flow sensors can be increased for additional redundancy.

Let the robot be equipped with L air flow sensors, which are mounted at the positions $\mathbf{r}_1, \ldots, \mathbf{r}_L$ and whose measurement axes are $\mathbf{n}_1, \ldots, \mathbf{n}_L$. For a compact representation, we combine the measurements $z_1, \ldots, z_L$ of all air flow sensors and formulate the joint measurement function

$$\begin{bmatrix} z_1 \\ \vdots \\ z_L \end{bmatrix} = \begin{bmatrix} h((\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_1) \cdot \mathbf{n}_1) + \varepsilon_1 \\ \vdots \\ h((\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_L) \cdot \mathbf{n}_L) + \varepsilon_L \end{bmatrix} \tag{8.3}$$

according to Equation (5.1) with $\varepsilon_i \sim \mathcal{N}\left(\varepsilon_i; 0, \sigma((\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_i) \cdot \mathbf{n}_i)^2\right)$. We solve Equation (8.3) for $\mathbf{v}$ by applying the inverse measurement function h^{-1} and obtain

$$\begin{bmatrix} h^{-1}(z_1 - \varepsilon_1) \\ \vdots \\ h^{-1}(z_L - \varepsilon_L) \end{bmatrix} = \begin{bmatrix} (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_1) \cdot \mathbf{n}_1 \\ \vdots \\ (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_L) \cdot \mathbf{n}_L \end{bmatrix} \tag{8.4}$$

$$= \begin{bmatrix} \mathbf{v} \cdot \mathbf{n}_1 \\ \vdots \\ \mathbf{v} \cdot \mathbf{n}_L \end{bmatrix} + \begin{bmatrix} (\boldsymbol{\omega} \times \mathbf{r}_1) \cdot \mathbf{n}_1 \\ \vdots \\ (\boldsymbol{\omega} \times \mathbf{r}_L) \cdot \mathbf{n}_L \end{bmatrix} \tag{8.5}$$

$$= \begin{bmatrix} \mathbf{n}_1, \ldots, \mathbf{n}_L \end{bmatrix}^T \mathbf{v} + \begin{bmatrix} \mathbf{r}_1 \times \mathbf{n}_1, \ldots, \mathbf{r}_L \times \mathbf{n}_L \end{bmatrix}^T \boldsymbol{\omega} \tag{8.6}$$

$$= A_{\mathrm{f}} \mathbf{v} + B_{\mathrm{f}} \boldsymbol{\omega} \tag{8.7}$$

with the constant matrices $A_\mathrm{f} = [\mathbf{n}_1, \ldots, \mathbf{n}_L]^T$ and $B_\mathrm{f} = [\mathbf{r}_1 \times \mathbf{n}_1, \ldots, \mathbf{r}_L \times \mathbf{n}_L]^T$ depending on the arrangement of the air flow sensors on the vehicle. We invert A_f using the left pseudo-inverse $A_{\mathrm{f,left}}^{-1} = (A_\mathrm{f}^T A_\mathrm{f})^{-1} A_\mathrm{f}^T$, which requires that $\mathrm{rank}(A_\mathrm{f}) = 3$. This can be guaranteed by at least three air flow sensors having linearly independent measurement axes. We obtain

$$\mathbf{v} = A_{\mathrm{f,left}}^{-1} \left(\begin{bmatrix} h^{-1}(z_1 - \varepsilon_1) \\ \vdots \\ h^{-1}(z_L - \varepsilon_L) \end{bmatrix} - B_\mathrm{f} \boldsymbol{\omega} \right) , \tag{8.8}$$

which is the least-squares solution to $\mathbf{v}$ of the over-constrained system in Equation (8.7) in the case of $L > 3$ [176]. In the case of $L = 3$, it is the exact solution and the left pseudo-inverse is equal to the inverse A_f^{-1}.

In the odometry motion model, we obtain the rotational velocity $\boldsymbol{\omega}_\mathrm{I} = \boldsymbol{\omega} + \boldsymbol{\delta}$ from the IMU according to Equation (6.2). Here, we assume that the IMU is mounted aligned to the body-fixed frame of reference. Otherwise, the corresponding rotation has to be taken into account as described in Section 6.1. The measurement error $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{\delta}; \mathbf{0}, \Sigma_{\boldsymbol{\omega},\mathrm{I}})$ is modeled as zero mean Gaussian noise so that the resulting velocity estimates are

$$\boldsymbol{\omega} = \boldsymbol{\omega}_\mathrm{I} - \boldsymbol{\delta} \tag{8.9}$$

and

$$\mathbf{v} = A_{\mathrm{f,left}}^{-1} \left(\begin{bmatrix} h^{-1}(z_1 - \varepsilon_1) \\ \vdots \\ h^{-1}(z_L - \varepsilon_L) \end{bmatrix} - B_\mathrm{f} (\boldsymbol{\omega}_\mathrm{I} - \boldsymbol{\delta}) \right) . \tag{8.10}$$

In the particle filter, we utilize the dead-reckoning odometry motion model as the proposal distribution in the prediction step. Thereby, we sample the measurement errors $\varepsilon_1, \ldots, \varepsilon_L$ and $\boldsymbol{\delta}$ from zero mean Gaussian distributions where the covariances can be identified from recorded measurement data with ground truth information. Subsequently, we propagate the errors through Equation (8.9) and Equation (8.10) and compute the position hypothesis $\mathbf{p}^{[i]}$ of each particle from the resulting velocities by numerical integration. Additionally, we exploit the orientation estimated by the on-board sensor data fusion of the IMU by sampling the orientation hypothesis $\mathbf{q}^{[i]}$ of each particle directly from the IMU sensor model described in Section 6.2.

8.3 Odometry Data with Temporally Correlated Measurement Errors

One fundamental assumption of the Bayes filter is the Markov assumption, which states that the random variables of the measurements and the motion of the robot are conditionally independent given the state of the system [169]. However, as can be seen in

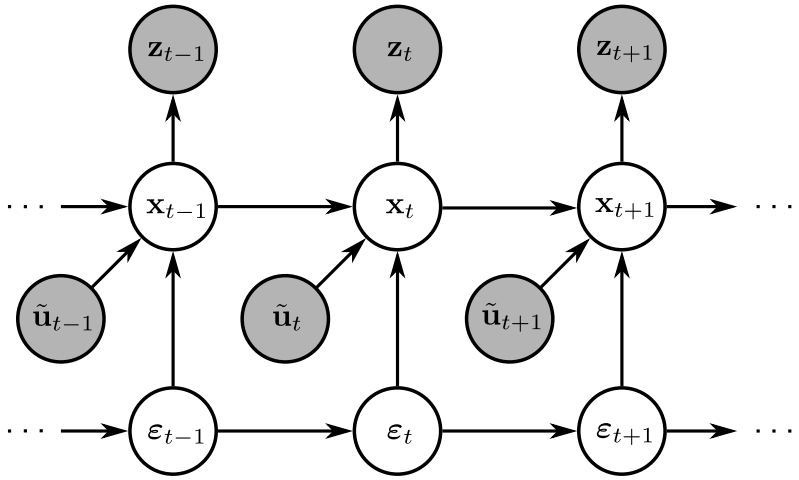


Figure 8.2: The extended dynamic Bayes network for localization of a mobile robot taking into account the temporal correlations in the odometry measurement errors. It characterizes the evolution of the states $\mathbf{x}$, the odometry $\tilde{\mathbf{u}}$, the measurements $\mathbf{z}$ and the measurement errors of the odometry data $\boldsymbol{\varepsilon}$.

Figure 5.5, the measurement errors of an air flow sensor, and therefore the corresponding random variables, are temporally correlated. This effect is caused by turbulences and the motion of the surrounding air, which is displaced by the vehicle and partially accelerated with it. Although Bayes filters have been found to be surprisingly robust to violations of the Markov assumption [169], we explicitly take the correlation into account in the filtering process.

To achieve this, we combine the measurement errors of all air flow sensors together in the vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_L]^T$ and explicitly consider their history as illustrated in Figure 8.2. In particular, we estimate the history of errors together with the state of the robot and extend the posterior of the particle filter to

$$p(\mathbf{x}_{1:t}, \boldsymbol{\varepsilon}_{1:t} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{u}}_{1:t}) = \eta\, p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t}, \tilde{\mathbf{u}}_{1:t})\, p(\mathbf{x}_{1:t}, \boldsymbol{\varepsilon}_{1:t} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \quad (8.11)$$

where η is a normalizer. We factorize the second conditional probability twice and obtain

$$\begin{aligned} p(\mathbf{x}_{1:t}, \boldsymbol{\varepsilon}_{1:t} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) &= p(\mathbf{x}_t, \boldsymbol{\varepsilon}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1}, \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \qquad (8.12) \\ &= p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t}, \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \\ &\quad \cdot p(\boldsymbol{\varepsilon}_t \mid \mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1}, \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t})\ . \qquad (8.13) \end{aligned}$$

Under the Markov assumption in the extended network shown in Figure 8.2, Equa-

tion (8.11) together with Equation (8.13) can be simplified to

$$\begin{aligned} p(\mathbf{x}_{1:t}, \boldsymbol{\varepsilon}_{1:t} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{u}}_{1:t}) &= \eta\, p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t, \tilde{\mathbf{u}}_t) \\ &\quad \cdot p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t-1}) \qquad (8.14) \\ &= \eta\, p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t, \tilde{\mathbf{u}}_t) \\ &\quad \cdot p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1}) \\ &\quad \cdot p(\mathbf{x}_{1:t-1}, \boldsymbol{\varepsilon}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \tilde{\mathbf{u}}_{1:t-1})\ . \qquad (8.15) \end{aligned}$$

Here, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t, \tilde{\mathbf{u}}_t)$ is the odometry motion model as described in Section 8.2, conditioned on the air flow measurement error. Note that in this context, $\tilde{\mathbf{u}}$ represents the air flow and IMU odometry measurement data. The term $p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1})$ is the air flow measurement error transition model and Equation (8.15) follows from d-separation [16] on the dynamic Bayes network (Figure 8.2). Learning this full high-dimensional probability density function would require a large amount of data and is prone to overfitting. Therefore, we apply the joint Gaussian approximation

$$\mathbf{e}_{1:t} \sim \mathcal{N}\left(\mathbf{e}_{1:t}; \mathbf{0}, \Sigma_{\boldsymbol{\varepsilon}}\right) \quad \text{with} \quad \mathbf{e}_{1:t} = \left[\boldsymbol{\varepsilon}_1^T, \ldots, \boldsymbol{\varepsilon}_t^T\right]^T \quad \text{and} \quad \Sigma_{\boldsymbol{\varepsilon}} = \mathrm{Cov}(\mathbf{e}_{1:t})\ . \quad (8.16)$$

In the prediction step of the particle filter, we sample from the odometry motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t, \tilde{\mathbf{u}}_t)\, p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1})$ as the proposal distribution by first sampling the temporally correlated measurement error $\boldsymbol{\varepsilon}_t$ from the conditional distribution $p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1})$. This conditional distribution can be derived from the joint Gaussian (8.16) with the partitioned covariance

$$\Sigma_{\boldsymbol{\varepsilon}} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \qquad (8.17)$$

so that the blocks are $\Sigma_{11} = \mathrm{Cov}(\mathbf{e}_{1:t-1})$ and $\Sigma_{22} = \mathrm{Cov}(\boldsymbol{\varepsilon}_t)$. This leads to the conditional Gaussian distribution [41]

$$p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1}) = \mathcal{N}\left(\boldsymbol{\varepsilon}_t; \Sigma_{21}\Sigma_{11}^{-1}\mathbf{e}_{1:t-1}, \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)\ , \qquad (8.18)$$

which is suitable for an efficient sampling. In the second step, we sample the motion of the vehicle from $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t, \tilde{\mathbf{u}}_t)$ conditioned on the (temporally correlated) measurement error $\boldsymbol{\varepsilon}_t$ of the air flow sensors.

The method described above accounts for the temporal correlations in the air flow measurement errors in the particle filter. However, in practice, amending each particle with the full history of the measurement errors results in the computational complexity $O(TN)$ of each resampling step where T is the length of the trajectory and N is the number of particles. As can be seen in Figure 5.5, in practice, the correlations of the measurement errors are limited to a few seconds so that one can safely limit the size of

the considered history to a constant value h depending on the temporal correlations and the measurement frequency. This results in the approximation

$$p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{1:t-1}) \approx p(\boldsymbol{\varepsilon}_t \mid \boldsymbol{\varepsilon}_{t-h:t-1}) \tag{8.19}$$

so that the resampling step of the particle filter is in $O(N)$ as in its standard implementation.

8.4 Experimental Evaluation

We evaluated our efficient localization approach with the odometry motion model using our third blimp prototype (see Section 2.1.3). The blimp was equipped with five tiny sonar sensors, three air flow sensors mounted on poles, and an IMU. In our experiments, we used the same experimental setup and recorded measurement and control data as described in Section 6.3. The trajectory traveled during 10 min of manually operated flight in a maze-like indoor environment had a length of 276.6 m (see Figure 6.2). During the experiment, we obtained accurate ground truth states from an optical motion capture system. Despite the fact that the IMU relies on magnetic measurements, which can be subject to serious disturbances, the RMS error of the orientation estimates was only $3.22°$ during the experiment in the indoor environment.

We implemented two variants of our novel approach as well as two state-of-the-art approaches to airship localization:

1. *Vanilla*: The vanilla implementation of the particle filter with the standard physical simulation-based control motion model as described in Section 3.3.

2. *Vanilla+MP*: The vanilla implementation extended by the simultaneous estimation of the air drag parameters of the control motion model. This approach is described in Chapter 7.

3. *Flow-IMU-Odometry*: Our novel localization approach with the odometry motion model using air flow and IMU measurements *without* taking into account the temporal correlations of the air flow measurements.

4. *Flow-IMU-Odometry+TC*: Our novel localization approach with the odometry motion model using air flow and IMU measurements taking into account the temporal correlations of the air flow measurements as described in Section 8.3.

In our implementations of the particle filter we applied the same parameters and precalculations of the sensor and motion models as described in Section 6.3. Furthermore, we trained all models and evaluated the temporal correlations of air flow measurement

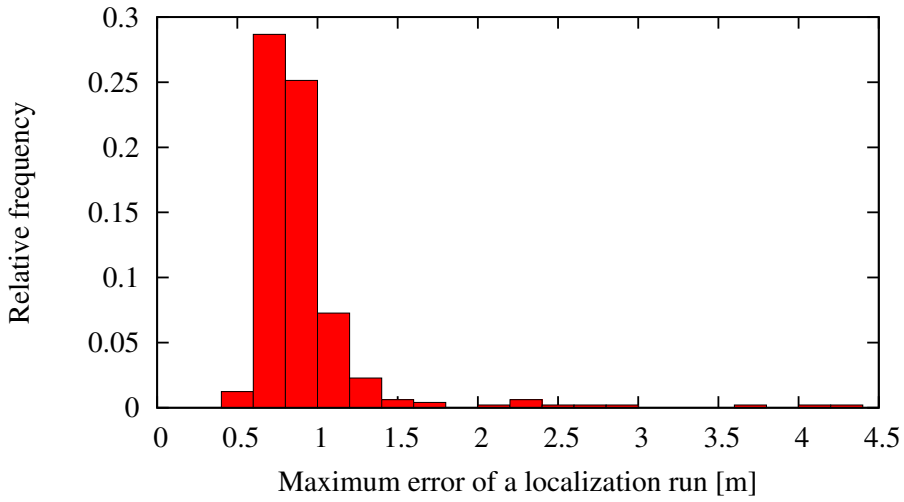


Figure 8.3: The relative frequency of the maximum localization error of all localization runs of all experiments. Additionally, in 32 % of the runs, the particle filter lost track of the blimp and the maximum error exceeded 4.5 m, which is not shown in the plot.

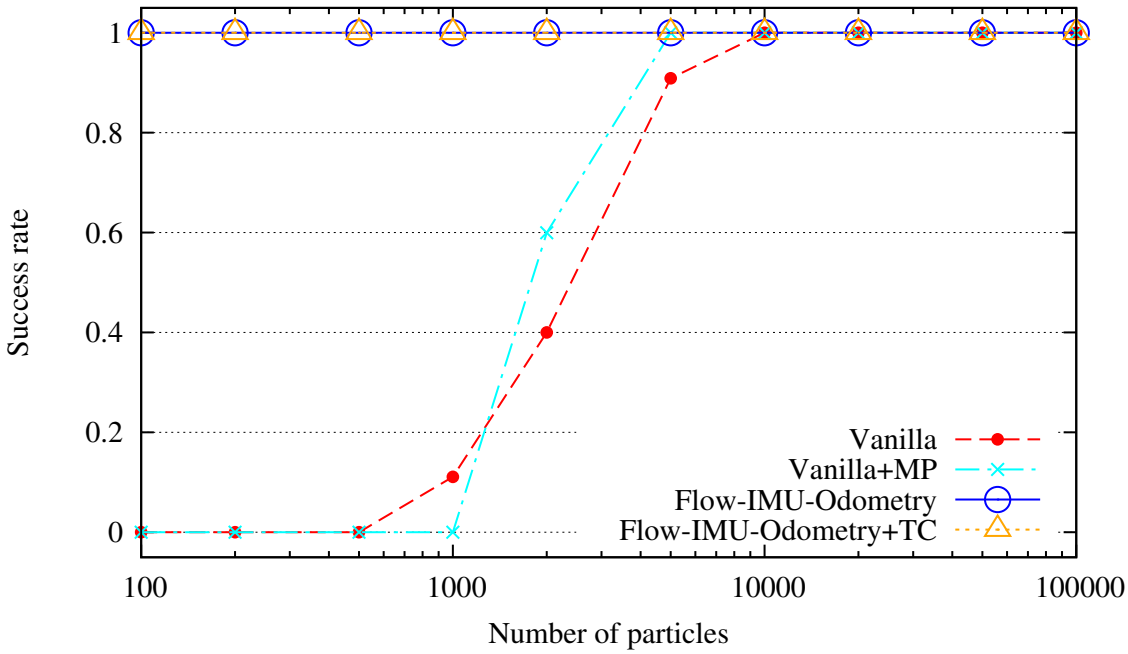


Figure 8.4: The success rate of the individual localization approaches.

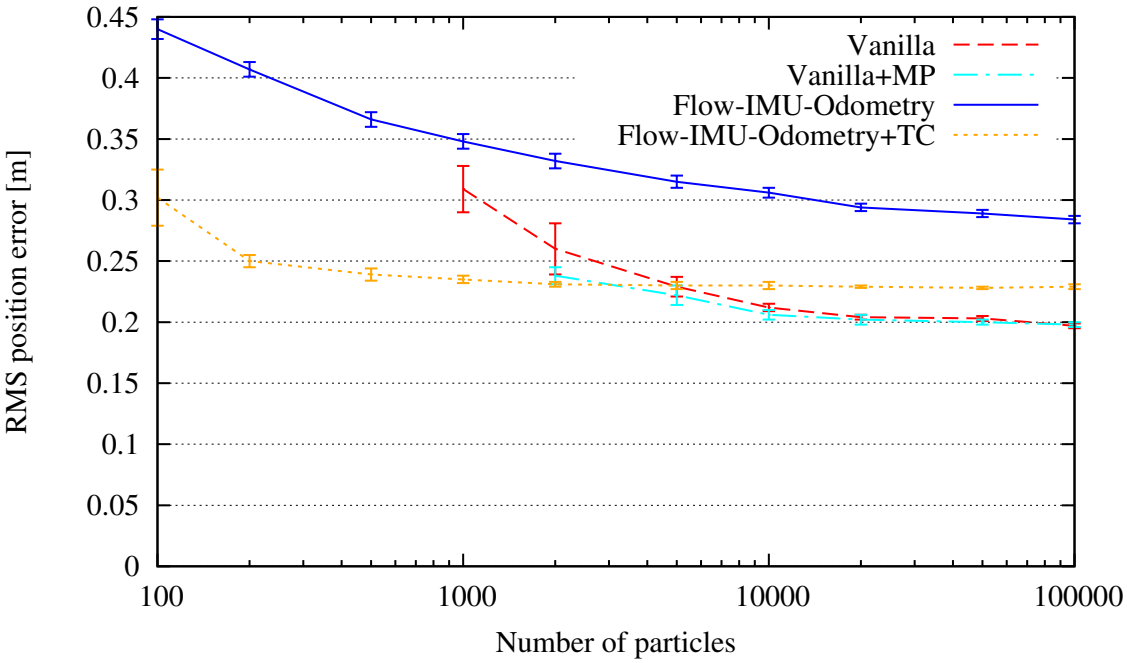


Figure 8.5: The average RMS localization error of the individual localization approaches. The error bars indicate the 95 % confidence intervals over ten successful runs.

errors from recorded data together with accurate ground truth states in the preparation of the experiment.

We evaluated and compared the localization methods on the sensor and control data recorded during operation. As a measure for the localization error, we used the Euclidean distance between the ground truth position and the position estimate of the particle filter, which is the weighted average of all particles. We evaluated the maximum error of each individual run of our experiments and found two groups of results as shown in Figure 8.3: good runs with a maximum error lower than 2 m and outliers with high errors (higher than 2 m). Therefore, we considered a localization run as successful if the position estimate of the filter never deviated more than 2 m from the ground truth position during the whole run. For each successful localization run, we evaluated the root mean square (RMS) of the three-dimensional position error over the whole trajectory.

Offline Comparison

In an offline experiment, we evaluated all localization methods on the recorded data with a varying number of particles. The success rate of the individual methods is depicted in Figure 8.4. As can be seen, the flow odometry approach facilitates a reliable localization even if a low number of particles is used. This is due to the fact that in this method, the velocities are directly measured by the odometry sensors, namely the IMU and the air flow sensors such that they do not need to be estimated in the particle filter. This decreases the dimensionality of the state estimation problem and therefore fundamentally

reduces the number of particles needed to densely represent the area of high likelihood in the posterior about the pose of the vehicle.

The average RMS position errors are depicted in Figure 8.5. The vanilla implementations generated significantly lower position errors when using a huge number of particles. This is caused by the fact that they integrate all sensor information and, in contrast to our novel approaches, additionally take into account the information about the control commands sent to the rotors. The simultaneous estimation of the motion model parameters (*Vanilla+MP*) has proven to outperform the vanilla implementation when only very sparse sensor information is available (see Chapter 7). Here, much more sensor data was available in addition, namely IMU and air flow measurements, so that there is no significant difference in the RMS position error for $100{,}000$ particles. A further result of this experiment is the significantly lower localization error with the odometry motion model when taking into account the temporal correlations of the air flow measurements. This is mainly due to the slow turbulences of the air accompanying the blimp causing slowly varying systematic measurement errors.

Online Comparison

For autonomous operation, the localization algorithm should be able to provide accurate state estimates during operation for motion planning and closed-loop control. Therefore, we compared our implementation of all localization methods listed above with respect to online operation. All localization methods were executed on an Intel® Atom™ N270 $1.6\,\text{GHz}$ with $1\,\text{GB}$ RAM in a single thread. This processor could potentially be carried by the blimp and is considerably faster than the Gumstix computer, especially in floating point calculations. In this setting, we determined the maximum number of particles that just enables the particle filter running online, i.e. processing the data as fast as it was generated by sensors and actuators.

Table 8.1 shows the maximum number of particles for online localization. The physical simulations of the control motion model of the vanilla implementations are computationally demanding so that their maximum number of particles is substantially lower and far from enabling a reliable online localization, as can be seen in Figure 8.4. Since the odometry motion model is computationally modest compared to the control motion model and the dimensionality of the state space is reduced, our novel implementations clearly outperform the vanilla implementations in terms of efficiency and allow a reliable online localization on a low-power embedded computer.

8.5 Related Work

Recently, autonomous navigation for unmanned aerial vehicles (UAVs) has become a growing research field. Especially in the context of quadrotor helicopters, several researchers considered online localization or SLAM with on-board sensors in GPS-denied

Method	# Particles
Vanilla	107
Vanilla+MP	105
Flow-IMU-Odometry	564
Flow-IMU-Odometry+TC	455

Table 8.1: The maximum number of particles that enables online localization of the blimp.

environments [5, 61, 67, 154, 177]. Furthermore, the problem of localizing fixed-wing vehicles with on-board sensors has been successfully addressed by Bry et al. [18]. The relatively high payload of these systems, however, allows them to carry more powerful sensors, e.g. laser range finders, and also facilitates more extensive on-board computations compared to miniature blimps.

Corresponding to the odometry sensors most wheeled robot platforms are equipped with, one can use airspeed sensors on flying vehicles for state estimation or control. For example Euston et al. [44] fused IMU and airspeed measurements of a UAV for attitude estimation. Furthermore, many researchers used optical flow on camera images for dead-reckoning odometry [29, 38, 85] or target detection and obstacle avoidance [179]. However, these approaches apply maximum likelihood state estimation or control. In contrast, we explicitly represent the measurement uncertainty of the sensors and define a probabilistic motion model for robust state estimation in recursive Bayes filters.

Besides optical flow and airspeed measurements, there are other techniques for the prediction of incremental movements of UAVs. Most localization and control approaches for miniature blimps rely on physical simulation-based control motion models, which are computationally demanding and require the tedious calibration of several parameters [55, 73, 84, 88, 183]. Furthermore, the motion of UAVs can be predicted based on the acceleration and angular rate measurements of an IMU [18]. However, due to the sedate navigation of blimps, their acceleration is low compared to gravity, which results in a poor signal to noise ratio of acceleration measurements.

In contrast to these approaches, our method closely follows the odometry motion model applied on most ground robots [35, 169]. We transfer the principle of dead-reckoning odometry to our blimp and combine the translational velocity information of air flow sensors with the rotational velocity estimated by an IMU. As opposed to the common localization approaches of wheeled robots, where the uncertainty of the precise wheel odometry sensors is approximated by considering Gaussian noise on the integrated 2D movement [169], we model the measurement uncertainty of the flow sensors and the IMU individually and propagate the uncertainty through the measurement equations.

Taking into account the correlations between measurement errors, which violate the Markov assumption in the recursive Bayes filter, has been addressed by only a few re-

searchers. For example Plagemann et al. [131] and Pfaff et al. [128] model the spatial correlations of the noise of adjacent laser beams in the sensor model of laser range finders. Furthermore, most localization approaches cope with the temporal correlations of the noise of non-odometry measurements in a simplistic way. Instead of modeling the correlations explicitly, the localization modules of the popular robot navigation frameworks CARMEN [113] and ROS [54], for instance, integrate a laser range measurement only whenever the robot executes a motion that exceeds a certain distance or rotation threshold. However, to our knowledge, there is no work on modeling the temporal correlations of odometry errors in the prediction step of the Bayes filter.

8.6 Conclusions

In this chapter, we presented a novel approach to probabilistic online localization for a miniature blimp equipped with lightweight ultrasound and air flow sensors as well as an IMU. For robust state estimation in a particle filter, we introduced a probabilistic odometry motion model that takes into account the measurement uncertainties of the individual sensors and decreases the dimensionality of the state space in the particle filter. Furthermore, our novel motion model can compute the odometry from the air flow sensors and the IMU in a linear way and is therefore computationally much more efficient than the standard physical simulation-based control motion model. Additionally, we introduced a general approach to take into account the temporal correlations of odometry errors in the prediction step of the particle filter. Our method applies a joint Gaussian approximation and therefore enables an efficient sampling of temporally correlated odometry noise in the particle filter.

We implemented our approach and carried out extensive experiments with a real blimp in a complex indoor environment. In all experiments, our approach has been proven to enable accurate and reliable online localization of a miniature blimp and to outperform the particle filter localization based on the standard control motion model. Since the odometry motion model provides accurate estimates of the velocity of the blimp, the dimensionality of the state space in the filter is decreased. Therefore, the number of particles required for a reliable localization is reduced by one order of magnitude. Additionally, we demonstrate significant improvements in terms of localization accuracy by taking into account the temporal correlations of the air flow measurements in our novel odometry motion model.

Part II

Motion Planning and Control

Chapter 9

Basic Planning and Control Techniques

Planning and control for autonomous robots is a well-studied topic and the diversity of robotic systems has spawned a variety of planning and control algorithms. In this chapter, we give a formal problem definition for motion planning and discuss several planning techniques with regard to motion planning for autonomous airships. We introduce the A* graph search algorithm and the sampling-based tree planning approach, which both are the basis of our online planning system introduced in the following chapter. Furthermore, we introduce the linear quadratic regulator (LQR) for optimal control of linear Gaussian systems.

• • • • •

There are two fundamental tasks in autonomous navigation for intelligently behaving mobile robots. First, the robot needs to perceive its environment in order to estimate the state of the world, which is its own state and the state of the environment. Second, given the estimated state, the robot decides about a single action or a sequence of actions that are suitable to reach a predefined goal or to accomplish its job. This chapter gives an overview of techniques to determine suitable actions for a mobile robot. In mobile robotics, this task is often divided into two separate parts. The motion planning task focuses on the generation of a collision-free path from a start state to a goal state. The motion control task copes with the execution of the path and corrects small deviations from the desired path by adjusting the control commands during execution. In this chapter, we will provide a formal problem definition and introduce basic planning and control techniques that are suitable for autonomous mobile robot navigation.

9.1 Motion Planning

Throughout this thesis, we consider the autonomous navigation of an indoor airship in a static environment that is known in advance. Accordingly, we assume that a complete description of the robotic system and the environment is provided for motion planning.

We define the general motion planning problem by the following components:

- The **state space** $\mathcal{X}$ is the manifold the robot is operating in. The free state space $\mathcal{X}_{\text{free}} \subseteq \mathcal{X}$ represents the environment in a way so that the free state space only contains states that are valid. Often, a state is considered as valid if the robot is not in collision with the environment in that state.
- The **action space** $\mathcal{U}$, also called control space, defines the possible control commands the robot can execute.
- The **start state** $\mathbf{s} \in \mathcal{X}$.
- The **goal state** $\mathbf{g} \in \mathcal{X}$ or the goal region $\mathcal{G} \subset \mathcal{X}$.
- The **motion function** describing the motion of the robot given its current state and the applied control command.

The task of the planning algorithm is to compute a path that leads the robot from the start state to a goal state. In particular, the solution path is a sequence of states together with the controls moving the robot from the corresponding state to the subsequent one. In case of planning under differential constraints, each control is supplemented by the duration the control has to be applied to reach the successor state.

Depending on the robotic system, both the state space and the action space can be either discrete or continuous. For planning problems with a discrete state and action space, the problem can be formulated as a graph search where the set of states represents the nodes connected by actions and the cost of each edge is determined by the cost of the corresponding action. In such static graphs, A* [66] is the most popular graph search technique. In continuous state and control spaces, sampling-based planning techniques have proven to be a suitable means to solve even high-dimensional planning problems of real robotic systems. The most popular techniques are probabilistic road maps (PRMs) [82] and sampling-based tree planners [32] such as rapidly-exploring random trees (RRTs) [102]. However, there are certain limitations when planning for a miniature airship that requires kinodynamic motion planning [39]. Kinodynamic planning is a case of planning under differential constraints where second-order constraints and velocity and acceleration bounds must be satisfied [101]. As the controls needed for connecting two given states in PRMs and bidirectional RRTs cannot be calculated analytically in nonlinear kinodynamic systems [173], these techniques are not suitable

for miniature airships. Thus, unidirectional sampling-based tree planners have been the most popular means to cope with such systems. In the following, we introduce the basic planning techniques on which our approaches to autonomous navigation of indoor airships are based.

9.1.1 A* Graph Search

The A* algorithm [66] is an informed search method for finding the cheapest path from a start node to a goal node in a graph. In a nutshell, A* is a kind of best-first search. In every iteration it expands the node with the lowest value of the evaluation function f, taking into account a heuristic estimating the cost to the goal. In this way, A* minimizes the number of expanded nodes such that it is optimally efficient [147]. In the following, we give a more formal description of the A* algorithm according to Choset et al. [28] and Russel and Norvig [147].

Let $G = (V, E)$ be a finite graph with a set of nodes V (also called vertices) and a set of edges $E \subseteq V \times V$ connecting the nodes. Each edge $e_{i,j} = (v_i, v_j) \in E$ is a tuple of two nodes v_i and v_j representing a directed connection from $v_i \in V$ to $v_j \in V$. In motion planning graphs, each node represents a possible state $\mathbf{x} \in \mathcal{X}$ and each edge (v_i, v_j) means that the robot can move from v_i to v_j. A solution path in G from a start node v_s to a goal node v_g is a sequence of nodes

$$(v_{i_1}, \ldots, v_{i_n}) \tag{9.1}$$

so that $v_{i_1} = v_s$ and $v_{i_n} = v_g$ and there is a connecting edge between each pair of subsequent nodes v_{i_j} and $v_{i_{j+1}}$, i.e.

$$\{(v_{i_j}, v_{i_{j+1}}) \mid j \in [1, n-1]\} \subseteq E \; . \tag{9.2}$$

For optimal path planning, we define a cost function $c : E \to \mathbb{R}$ that assigns a cost to each edge of the graph. Consequently, the cost of a solution path is the sum of the cost of all edges along the path

$$\tilde{c}\left((v_{i_1}, \ldots, v_{i_n})\right) = \sum_{j=1}^{n-1} c\left((v_{i_j}, v_{i_{j+1}})\right) \; , \tag{9.3}$$

and an optimal path is a solution path with minimal cost with respect to all solution paths.

The A* graph search is guided by a heuristic function $h : V \to \mathbb{R}$, which estimates the cost of the cheapest path from the given node to the goal. Algorithm 5 shows the A* graph search. During search, the A* algorithm maintains the cheapest path from the start node to each node expanded so far. The predecessor of each node on this path is stored

Algorithm 5 A*GRAPHSEARCH

Input: Graph $G = (V, E)$, cost function $c : E \to \mathbb{R}$, heuristic function $h : V \to \mathbb{R}$, start node $v_s \in V$, goal node $v_g \in V$

Output: The cheapest path from v_s to v_g.

$p[v_s] \leftarrow$ NULL *// predecessor map*
$g[v_s] \leftarrow 0$ *// cost map*
$C \leftarrow \emptyset$ *// closed set*
Initialize $Q \leftarrow \{(v_s, h(v_s, v_g))\}$ as a priority queue that is sorted by the second element of the tuple.
while $Q \neq \emptyset$ **do**
  $v \leftarrow \text{pop}(Q)$
  $C \leftarrow C \cup \{v\}$
  if $v = v_g$ **then** *// Return the cheapest path given by the predecessor map.*
    $P = [v]$
    while $p[v] \neq$ NULL **do**
      $v \leftarrow p[v]$
      $\text{prepend}(P, v)$
    end while
    return P
  end if
  $S = \{v' \in V \mid \exists (v, v') \in E\}$ *// the successors of v*
  for all $v' \in S \setminus C$ **do**
    if $v' \notin Q \ \vee \ g[v] + c((v, v')) < g[v']$ **then**
      $p[v'] \leftarrow v$
      $g[v'] = g[v] + c((v, v'))$
      if $v' \notin Q$ **then**
        $\text{push}(Q, (v', g[v'] + h(v')))$
      else
        $\text{decreasekey}(Q, (v', g[v'] + h(v')))$
      end if
    end if
  end for
end while
return [ ] *// No solution*

in the predecessor map $p : V \to V \cup \{\text{NULL}\}$. The cost of this path of each node is stored in the cost map $g : V \to \mathbb{R}$. As a kind of best-first search, A* maintains a set of nodes open for expansion. In every iteration, it expands the node with the best value of the evaluation function f. For an efficient implementation, the open set is represented by a priority queue sorted by the evaluation function. In particular, the evaluation function of a node v is

$$f(v) = g(v) + h(v) \ , \tag{9.4}$$

which is the estimated cost of the cheapest solution path through v given by the cost $g(v)$ of the cheapest path to v and the estimated cost $h(v)$ from v to the goal. To avoid expanding a node twice in a cyclic graph, the closed set C stores all nodes expanded so far. The expansion of a node is described in line 16 to 27 of Algorithm 5. It considers all successors of the expanded node and adds them to the priority queue together with the corresponding value of the evaluation function (line 22). In case a successor is already contained in the queue and a shorter path is found, the corresponding element in the queue is updated (line 24). As soon as the goal is reached, the cheapest path is reconstructed by a backwards stepping through the predecessor map (line 8 to 15).

Since A* explores the graph incrementally through the expansion of nodes, the graph has not to be explicitly specified and stored completely in advance, which usually is a great advantage in motion planning tasks with huge state spaces. Furthermore, A* returns the optimal (i.e. the cheapest) path in a finite graph if the following three preconditions hold [147]:

- The cost function c is non-negative.
- The heuristic function is admissible, i.e.

$$h(v) \leq h^\star(v) \quad \forall v \in V \ , \tag{9.5}$$

  where $h^\star(v)$ is the real cost of the cheapest path from v to the goal.

- The heuristic function is consistent (also called monotonic), i.e.

$$h(v) \leq c(v, v') + h(v') \quad \forall v, v' \in V \ . \tag{9.6}$$

In addition, A* is complete, i.e. it will find a solution if there is one, and it is optimally efficient, which means that given any heuristic function there is no other algorithm that computes the optimal path and in doing so expands fewer nodes than A* [147].

Consequently, A* is a popular approach in optimal motion planning on low-dimensional discrete or discretized state and control spaces.

9.1.2 Sampling-based Tree Planning

Sampling-based tree planners have proven to be a suitable means in many robotic motion planning problems because they naturally can deal with continuous state and control spaces. They have become a popular approach in challenging motion planning problems, especially in high-dimensional state spaces where a discretization with fixed resolution would lead to a huge number of nodes.

Sampling-based tree planners iteratively grow a tree of motions in the state space using random sampling. Since they basically require only the forward kinematics of the system when growing the tree, they can solve both, geometric planning problems as well as planning problems under differential constraints. In general, tree planners initialize their tree with the start state as a single (root) node. When growing the tree of motions and corresponding states, each iteration is performed according to the following four steps:

1. **Node selection**. Select a node of the tree from which the tree should be expanded.

2. **Motion selection**. Choose a control and a duration of the motion expanding the tree from the selected node. When planning in continuous state and action spaces, the state propagation function $f : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^+ \to \mathcal{X}$ maps a control applied for a certain time span at a given state to the subsequent state.

3. **Validity check**. Check whether the path and the successor state resulting from the chosen motion are valid and if so expand the tree. This step requires a function $v : \mathcal{X} \to \{\bot, \top\}$, which evaluates the validity of a given state – usually by collision checking given the map of the environment.

4. **Goal check**. Check whether the goal is reached and the planning process can be completed. The function $g : \mathcal{X} \to \{\bot, \top\}$ indicates whether the specified goal or goal region is reached through the given state. If so, the resulting path is obtained by backwards stepping through the tree.

The first and the second step of each iteration are usually decided through random sampling from heuristics. The heuristics substantially influence the effectiveness of the planning algorithm and are designed domain or even problem dependent in most approaches [32]. There is a wide literature of efficient approaches to geometric motion planning. However, for planning under differential constraints, which is required for indoor airships, the literature is comparably sparse, as this problem is usually more complex to solve.

In principle, there are two distinct objectives during tree planning. On the one hand, the tree should grow in a way so that the entire state space will eventually be explored

by the tree. In that context, LaValle and Kuffner [102] introduced the *probabilistic completeness* of sampling-based planning algorithms; that means that if a solution exists it will eventually be found [28, 101]. On the other hand, an important objective is the efficiency of a planner. In order to achieve a high efficiency, the exploitation of knowledge about the current tree as well as the planning task can be helpful. With that knowledge, the growth of the tree can be focused towards the goal so that the number of required expansions of the tree is minimized. These two objectives, namely *exploration* and *exploitation*, are contradictory to some extent and therefore demand carefully chosen sampling heuristics. In the following, we give an overview of sampling heuristics suitable for planning under differential constraints according to Şucan and Kavraki [32].

There are several heuristics for node selection. A common technique is to introduce a Voronoi bias, i.e. to guide the tree towards large Voronoi regions to force a quick exploration of the state space. This method draws a uniform sample from the state space and chooses the node that is nearest to the sampled one according to a distance metric $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ on the state space [100]. This is one of the most successful heuristics but depends on a good distance metric [25]. Another approach is to consider the out-degree of the nodes in the tree in order to quickly reach unexplored space [72]. Furthermore, the node selection through the decomposition of the state space – often in a hierarchical way or in combination with a lower dimensional projection of the state space – has shown to contribute to the exploration of the state space [31, 97]. While the heuristics mentioned above are focusing on the exploration of the state space, the exploitation of knowledge in the motion selection to guide the tree towards the goal has been addressed by several methods. A directed expansion or keeping track of previously used directions can focus the growth of the tree towards promising regions of the state space [22, 25]. Such regions can be determined by computing paths – often in a discretized, lower-dimensional projection of the state space – that lead to the goal [132]. Furthermore, biasing the expansion of the tree towards the goal by sampling from the goal region periodically (also called goal-biasing) can improve the efficiency of the planning algorithm [33, 102] but can also degrade the performance in case of strong local minima [31].

Many of these techniques have been combined in several ways to successfully cope with various motion planning problems in simulation and for real robots. However, in the literature, there are hardly any approaches that can generically deal with motion planning problems under differential constraints and none of them efficiently provides online motion planning in high-dimensional state spaces.

9.2 The Linear Quadratic Regulator (LQR)

For the model predictive optimal control of linear systems, the linear quadratic regulator (LQR) is a popular means in the field of robotics and engineering. The linear feedback control law for keeping a system close to a given trajectory can be calculated efficiently in closed form such that it is suitable for real-time control even of high-dimensional systems. In the following, we summarize the finite-horizon discrete-time LQR controller according to Bertsekas [14].

The given trajectory, which is usually computed by the planning algorithm, specifies the desired full state and control information $(\mathbf{x}_t^{\blacktriangledown}, \mathbf{u}_t^{\blacktriangledown})$ at discrete time steps $t \in [1, T]$. The model of the linear Gaussian system can in general be specified by the system dynamics equation

$$\mathbf{x}_{t+1} = A_t \mathbf{x}_t + B_t \mathbf{u}_t + \mathbf{w}_t \tag{9.7}$$

where $\mathbf{w}_t$ is the white Gaussian noise and A_t and B_t are the system-dependent process matrices. Taking into account the desired trajectory, which complies with the system dynamics through $\mathbf{x}_{t+1}^{\blacktriangledown} = A_t \mathbf{x}_t^{\blacktriangledown} + B_t \mathbf{u}_t^{\blacktriangledown}$, Equation (9.7) can be written as

$$(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^{\blacktriangledown}) = A_t(\mathbf{x}_t - \mathbf{x}_t^{\blacktriangledown}) + B_t(\mathbf{u}_t - \mathbf{u}_t^{\blacktriangledown}) + \mathbf{w}_t \tag{9.8}$$

in terms of the deviation from the desired trajectory.

At time t, the LQR controller aims to minimize the cost function

$$\mathrm{E}\left[\sum_{\ell=t}^{T} \left((\mathbf{x}_\ell - \mathbf{x}_\ell^{\blacktriangledown})^T P (\mathbf{x}_\ell - \mathbf{x}_\ell^{\blacktriangledown}) + (\mathbf{u}_\ell - \mathbf{u}_\ell^{\blacktriangledown})^T Q (\mathbf{u}_\ell - \mathbf{u}_\ell^{\blacktriangledown})\right)\right], \tag{9.9}$$

which quadratically penalizes the expected deviation of the actual future states and controls $(\mathbf{x}_t, \mathbf{u}_t), \ldots, (\mathbf{x}_T, \mathbf{u}_T)$ from those defined by the trajectory. Here, E denotes the expectation of the linear Gaussian system and the weight matrices P and Q are symmetric and positive definite.

The optimal control $\mathbf{u}_t$ at time t minimizes the quadratic cost function (9.9). It can be derived as the linear function

$$(\mathbf{u}_t - \mathbf{u}_t^{\blacktriangledown}) = G_t(\mathbf{x}_t - \mathbf{x}_t^{\blacktriangledown}) \tag{9.10}$$

of the actual state and the trajectory where G_t is the linear feedback gain matrix. The gain matrix G_t is computed recursively by the discrete-time Riccati equation

$$M_T = P \tag{9.11}$$

$$\forall \ell \in [t, T-1]: \quad G_\ell = -(B_\ell^T M_{\ell+1} B_\ell + Q)^{-1} B_\ell^T M_{\ell+1} A_\ell \tag{9.12}$$

$$M_\ell = P + A_\ell^T M_{\ell+1} A_\ell + A_\ell^T M_{\ell+1} B_\ell G_\ell \,. \tag{9.13}$$

This equation can be computed in linear time with respect to the length of the trajectory. In practice, the horizon of the model prediction is often limited to a constant size h_{LQR} so that the gain matrix of the LQR controller can be calculated by considering the time steps $\ell \in [t, t + h_{\text{LQR}}]$ instead of the whole remaining trajectory.

Chapter 10

Online Motion Planning and Control

Miniature airships are especially challenging with respect to autonomous navigation since they behave nonlinearly, are typically under-actuated, and are also subject to drift. These aspects, paired with their high-dimensional state space, demand efficient planning and control techniques. In this chapter, we present a highly effective approach to autonomous navigation of miniature blimps in mapped environments, which applies a multi-stage algorithm to accomplish strongly goal-directed tree-based kinodynamic motion planning. It performs path-guided sampling and selects optimal actions leading the robot towards sampled subgoals. Thus, our approach can quickly provide a partial trajectory, which is extended and refined in the consecutive planning steps during operation. The navigation system has been implemented and is able to reliably operate a robotic blimp in a real-world setting. Further experiments demonstrate that our approach outperforms a standard tree planner.

• • • • •

Miniature airships naturally float in the air and therefore can fulfill long-term operation tasks and navigate safely. However, these favorable properties come at the cost of some challenges imposed on autonomous navigation for airships. The very limited acceleration capabilities together with the serious under-actuation make it practically infeasible to neglect second-order dynamics. Due to the nonlinear, non-holonomic, and drift-prone second-order dynamical system, kinodynamic motion planning has to be performed in the 12-dimensional state space consisting of the pose and velocity of the robot. Furthermore, the cost of the shortest path does in general not follow any metrics [97] and the commonly applied decoupling of the planning of the trajectory shape from that of the velocities is not applicable.

In this chapter, we consider the task of indoor navigation on a round trip in a mapped environment, e.g. in a continuous surveillance task. We present an approach to online autonomous navigation including the state estimation, a multi-stage planner, a mission control module, and a controller. We approximate the kinodynamic motion planning, which in general is PSPACE-hard [137], by first applying A* search to generate a collision-free path on a discretized low-dimensional subspace of the state space. In the second stage, a tree planner applies path-guided sampling to quickly generate a trajectory, which is extended and refined in consecutive planning cycles by re-using a pruned version of the tree. We show the reliability and performance of our navigation system in extensive experiments in simulation and with a real robotic blimp. Furthermore, we show that our planning approach outperforms a standard goal-biased RRT planner [84].

10.1 Related Work

In the past, several authors considered the problem of autonomous navigation of blimps. The majority of approaches, however, focused on the control of robotic blimps in the absence of obstacles.

Some authors successfully applied model-free learning to control a single selected degree of freedom of a real indoor blimp [87, 142]. In contrast to that, Liu et al. [107] and Zufferey et al. [183] learned controllers for the full state space of the blimp. However, both papers report a large number of iterations when learning a controller that is specific for a single trajectory or goal configuration. Several model predictive approaches have been proposed, namely decoupling of components [73], extending the classic LQR [52], or nonlinear control [117]. As opposed to the LQR controller we designed for our navigation system, the tuning of the control design parameters of those controllers usually is time-consuming.

During the last decades, motion planning for mobile robots has been an area of active research [28, 101] and has spawned a wide literature of efficient approaches to geometric motion planning [81, 82, 93, 150]. However, these approaches rely on the assumption that the differential constraints, which are inherent in most physical robotic systems, are ignored in the planning task and the corresponding complexity is moved to the controller. This has been a common method for systems that are not required to operate close to their maximum velocity and acceleration, for example, slowly moving robotic arms.

In contrast, the literature of planning under differential constraints, which is required for indoor airships, is comparably sparse as this problem is usually more complex to solve. Many high-dimensional kinodynamic motion planning problems have been successfully addressed using sampling-based tree planning techniques [32]. For example, Kim and Ostrowski [84] proposed an RRT with goal biasing for blimp motion planning. However, their planner was designed for an outdoor blimp in an obstacle-free environ-

ment. Ladd and Kavraki [96] account for under-actuation and drift of robots most planners are suffering from. Unlike our approach, they aim to explore the full state space, which is time-consuming even in scenarios that are not very complex.

The concept of multi-stage planning is popular for discrete goal-directed online motion planning in real robot applications [26, 53, 160]. Maček et al. [109] use the nodes of a path on a route graph as subgoals in an RRT together with randomly sampled subgoals and actions for online motion planning. However, they do not re-use valid parts of the tree of the previous planning cycle. Rickert et al. [138] and Plaku et al. [132, 133] presented tree planners that quickly explore the lower-dimensional workspace of the robot through multiple paths to the goal that are not necessarily collision-free. In contrast to this, our planner is guided by one collision-free low-dimensional path, which is additionally augmented by velocity information. Therefore, our algorithm results in a more focused exploration of the state space and enforces that even partial trajectories do not head for a dead end.

In our approach, we apply a novel combination of many existing techniques in order to build an efficient navigation system, which has several desirable properties. It takes into account obstacles and is strongly goal-directed and efficient. Therefore, it is suited for online navigation of a real indoor blimp.

10.2 Efficient Motion Planning for Airships

As described in Section 2.4, we model our blimp as a floating rigid body in a three-dimensional environment. Consequently, its state for kinodynamic motion planning is described by its pose and velocity in the 12-dimensional state space $\mathcal{X} \subseteq \mathrm{SE}(3) \times \mathbb{R}^6$. The blimp can be controlled by a three-dimensional vector $\mathbf{u} \in \mathcal{U} = [-1, 1]^3$ defining the relative forward, upward, and rotational thrust about the vertical axis. In a nutshell, the motion model can be described as a function

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \tag{10.1}$$

of the state $\mathbf{x}_t$ and the control $\mathbf{u}_t$ at time t. Here, we assume that the control command is applied for the certain duration Δt of one time step, which is fixed during planning. The successor state is computed through numerical integration as described in Section 2.6.

The environment of the blimp was mapped beforehand and is represented using the OctoMap framework (see Section 2.3). For collision checking, we compute a distance map and conservatively approximate the blimp by a set of spheres that are arranged along its longitudinal axis. Since collision checking in this way is just a lookup to the distance map at the centers of the spheres, it can be done efficiently. We specify a certain state $\mathbf{g} \in \mathcal{X}$ as goal, which should be reached within a predefined radius.

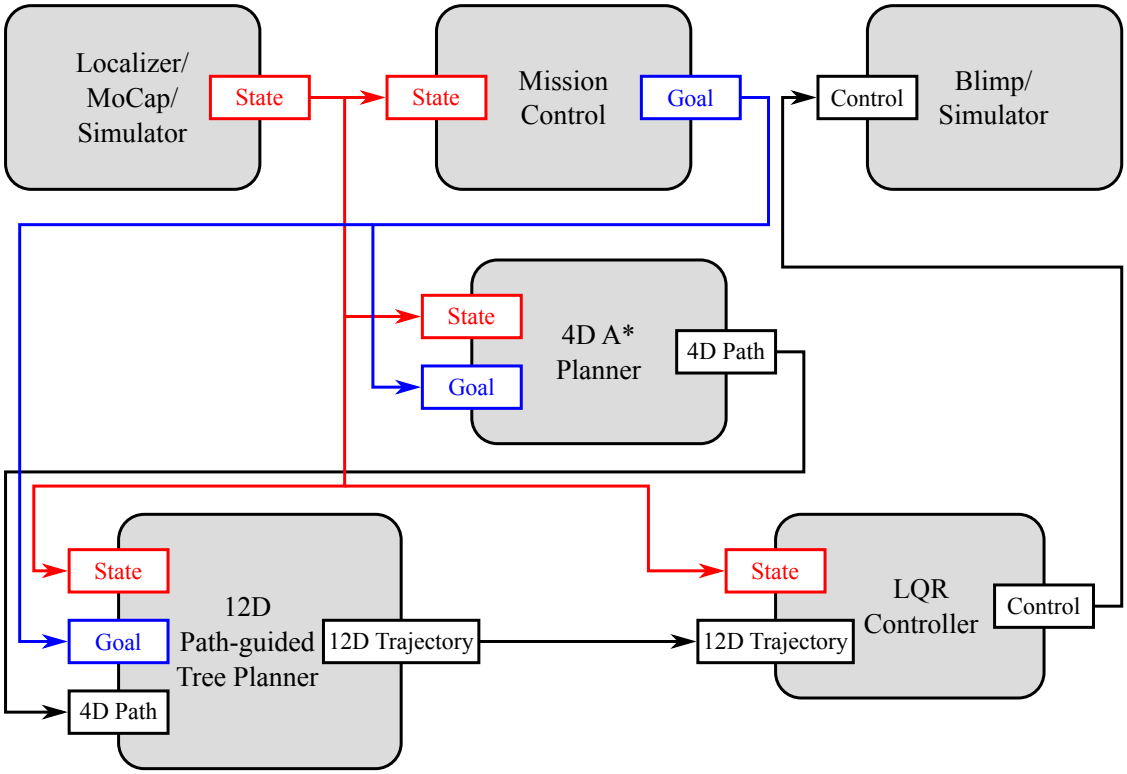


Figure 10.1: The interaction of the modules of our approach to autonomous blimp navigation. In the simulation experiments, we replaced the state estimation of the motion capture (MoCap) system and the blimp executing controls by a simulator module.

The interaction of the planning modules of our approach to autonomous navigation is shown in Figure 10.1. Our planning algorithm works in two stages. First, we apply A* search (see Section 9.1.1) to compute an optimal path assuming a simplified motion model on a discretized 4-dimensional subspace of the state space. In the second stage, a sampling-based tree planner efficiently searches for a trajectory in the full 12-dimensional state space by utilizing the A* path in order to draw samples in a goal-directed way. This prevents the tree planner from getting trapped, e.g. in a maze.

10.2.1 Low-dimensional Optimal Path Generation

The low-dimensional path generation provides a collision-free path. In this planning step, we consider a 4-dimensional subspace $\mathcal{X}' \subseteq \mathbb{R}^3 \times \mathrm{SO}(2)$ of the state space defined by the translation and the yaw-orientation of the blimp. This reduces the full state space by the velocity and the roll and pitch angles, which are not directly controllable. To allow for path generation by A* search, we discretize this subspace into a grid and guide the search by Euclidean distance heuristics. To additionally improve the efficiency of the A* search, we ignore the yaw-orientation at positions where the circumcircle of the blimp is collision-free. As the blimp can turn approximately on the spot when moving very slowly, we define the set of allowed actions as moving one grid cell forward, backward, upward, downward and rotating to the left and to the right. We define the resulting path computed on the 4D grid as a sequence of 4D states $\mathcal{P}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_N)$ with $\mathbf{x}'_i \in \mathcal{X}'$ for all $i \in [1, N]$.

10.2.2 Path-guided Sampling-based Tree Planning

In a preprocessing step, our tree planner augments each new 4D path $\mathcal{P}'$ resulting in the augmented (12D) path $\mathcal{P} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. Thereby, the roll and pitch angles are zero, since airships are usually stable in these dimensions. The forward, upward, and yawing velocity are determined based on the clearance to obstacles as well as the curvature of the 4D path and a maximum centripetal acceleration. Here, both, a low clearance or a high curvature, lead to a reduced velocity. Note that the augmented path $\mathcal{P}$ is not necessarily dynamically feasible but aims to focus the sampling of the tree planner to reasonable areas of the huge state space.

Algorithm 6 shows the pseudocode of our RRT-based tree planning approach with path-guided sampling and re-planning. In the first planning cycle, we initialize the tree with the current state propagated to the time t_{max} at which the planning cycle will be finished. In all subsequent planning cycles, we prepare the tree generated in the previous planning cycle for re-use by searching for the node that will be reached at t_{max} and pruning (line 1 to 3) similar to Bekris and Kavraki [11].

Algorithm 6 PATH-GUIDEDRE-PLANNING

Input: Previous tree $\mathcal{T}$, planning timeout t_{max}, augmented path $\mathcal{P}$, current state $\mathbf{x}$, goal $\mathbf{g}$
Output: A (partial) solution trajectory

1: Determine node $\mathbf{x}_{\text{start}}$ of $\mathcal{T}$ at time t_{max}
2: Prune everything below $\mathbf{x}_{\text{start}}$ from $\mathcal{T}$
3: $\mathcal{T}$.root $= \langle \mathbf{x}_{\text{start}}, \mathbf{0} \rangle$
4: $\mathbf{x}_{\text{closest}} = \mathbf{x}_{\text{start}}$
5: **while** CURRENTTIME() $< t_{\text{max}}$ **do**
6: $\quad \mathbf{x}_{\text{rnd}} \leftarrow$ GAUSSIANSAMPLEFROMPATH($\mathcal{P}$)
7: $\quad \mathbf{x}_{\text{near}} \leftarrow$ NEARESTNEIGHBOR($\mathcal{T}, \mathbf{x}_{\text{rnd}}$)
8: $\quad \mathbf{u}' \leftarrow$ OPTIMALACTION($\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rnd}}$)
9: $\quad \mathbf{x}_{\text{new}} \leftarrow \mathbf{f}(\mathbf{x}_{\text{near}}, \mathbf{u}')$ *// propagate the state*
10: $\quad$ **if** $v(\mathbf{x}_{\text{new}})$ **then** *// state validity check*
11: $\quad\quad \mathcal{T}$.insert($\langle \mathbf{x}_{\text{new}}, \mathbf{u}' \rangle$)
12: $\quad\quad$ EXTENDSAMPLINGINTERVAL($\mathcal{P}, \mathbf{x}_{\text{new}}$)
13: $\quad\quad \mathbf{x}_{\text{closest}} \leftarrow$ UPDATECLOSEST($\mathbf{x}_{\text{closest}}, \mathbf{x}_{\text{new}}$)
14: $\quad$ **end if**
15: **end while**
16: **return** SOLUTIONTRAJECTORY($\mathcal{T}, \mathbf{x}_{\text{closest}}$)

Figure 10.2 shows an example of an extension step of the tree. GAUSSIANSAMPLE-FROMPATH (line 6) utilizes the augmented path $\mathcal{P}$ to draw goal-directed samples from the state space. First, a position on the path is sampled from the current sampling interval. Then, the sample $\mathbf{x}_{\text{rnd}}$ is drawn from a Gaussian with the sampled augmented path element as mean. This implicitly induces goal-biasing and ensures that a valid partial trajectory is returned if the time available for planning runs out before the tree reaches the goal.

Our tree planner selects the NEARESTNEIGHBOR, which will be extended towards the sampled state $\mathbf{x}_{\text{rnd}}$ based on the weighted Euclidean metrics [84]

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T D (\mathbf{x}_1 - \mathbf{x}_2) \tag{10.2}$$

with a diagonal distance matrix D. Since the distance matrix is diagonal, we can find the nearest state efficiently by utilizing a kd-tree [51] containing all tree nodes scaled by the square root of D.

We select the optimal action leading from the nearest neighbor $\mathbf{x}_{\text{near}}$ towards the sampled state $\mathbf{x}_{\text{rnd}}$ as we describe in Section 10.2.3.

Finally, the sampling interval on $\mathcal{P}$ (see Figure 10.2) is extended according to the growth of the tree (line 12) and the node $\mathbf{x}_{\text{closest}}$ that is nearest to the goal is determined by UPDATECLOSEST (line 13). This can be done by selecting the node that is closest to

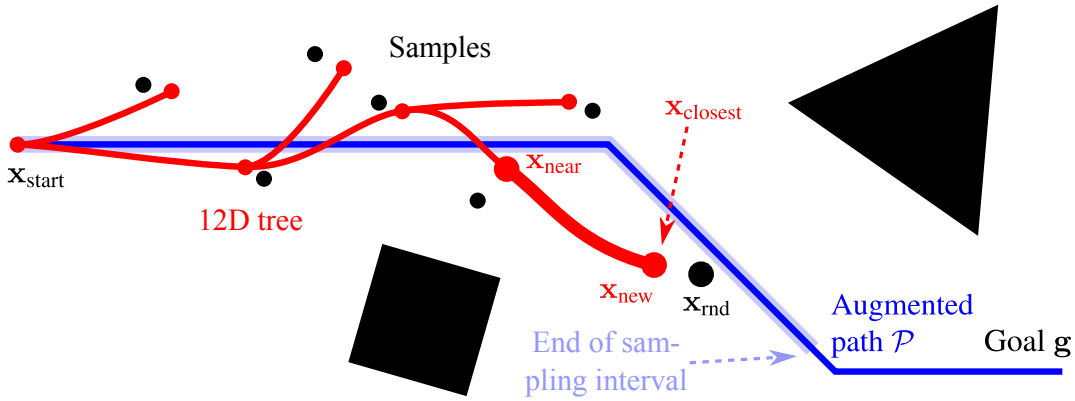


Figure 10.2: An example of an extension step of the tree with path-guided sampling. The augmented A* path $\mathcal{P}$ is shown in blue, the current sampling interval is highlighted in light blue. The node $\mathbf{x}_{\text{rnd}}$ is sampled around $\mathcal{P}$. $\mathbf{x}_{\text{near}}$, which is the nearest node to $\mathbf{x}_{\text{rnd}}$, is extended towards $\mathbf{x}_{\text{rnd}}$ resulting in the new node $\mathbf{x}_{\text{new}}$. The node $\mathbf{x}_{\text{closest}}$ is chosen as the node that is closest to the end of the current sampling interval. Obstacles are shown in black.

the end of the sampling interval on the guiding path. It ensures a good choice even when only a partial trajectory has been computed and the robot has to veer away from the goal when navigating through a maze.

10.2.3 Optimal Action Selection

We select the optimal action $\mathbf{u}'$ leading from a state $\mathbf{x}_t$ towards a target state $\mathbf{x}'$ with respect to the metric ρ. This means that we want to select the action

$$\mathbf{u}' = \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmin}}\, \rho(\mathbf{f}(\mathbf{x}_t, \mathbf{u}), \mathbf{x}') \tag{10.3}$$

that minimizes the metric distance to the target state $\mathbf{x}'$ after executing one motion step starting from $\mathbf{x}_t$.

By linearizing the motion model with respect to the control around the state $\mathbf{x}_t$ and the neutral control $\mathbf{0}$, we obtain

$$\mathbf{f}(\mathbf{x}_t, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}_t, \mathbf{0}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_t, \mathbf{0})\, \mathbf{u}\ . \tag{10.4}$$

Plugging Equation (10.4) and Equation (10.2) into Equation (10.3) results in

$$\begin{aligned}
\mathbf{u}' &= \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmin}} (C_t\mathbf{u} + \mathbf{y}_t)^T D (C_t\mathbf{u} + \mathbf{y}_t) && (10.5)\\
&= \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmin}}\, \mathbf{u}^T C_t^T D\, C_t\mathbf{u} + \mathbf{u}^T C_t^T D\, \mathbf{y}_t + \mathbf{y}_t^T D\, C_t\mathbf{u} + \mathbf{y}_t^T D\, \mathbf{y}_t && (10.6)\\
&= \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmin}}\, \frac{1}{2}\mathbf{u}^T C_t^T D\, C_t\mathbf{u} + \mathbf{u}^T C_t^T D\, \mathbf{y}_t && (10.7)
\end{aligned}$$

with $C_t := \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_t, \mathbf{0})$ and $\mathbf{y}_t := \mathbf{f}(\mathbf{x}_t, \mathbf{0}) - \mathbf{x}'$ by omitting the constant term and exploiting the symmetry of D.

In an unbounded control space $\mathcal{U}$ this can be solved in closed form. For robots with bounded controls $\mathbf{u}_{\text{low}} \leq \mathbf{u} \leq \mathbf{u}_{\text{high}}$ such as our blimp, this problem can be solved efficiently, e.g. using a quadratic programming-based solver [46].

10.3 LQR Control

The trajectory computed by the planning algorithm consists of the full state and control information $(\mathbf{x}_t^{\blacktriangledown}, \mathbf{u}_t^{\blacktriangledown})$ at discrete time steps $t \in [1, T]$. In order to keep the robot on this trajectory, we apply finite-horizon discrete-time linear-quadratic regulation (LQR) control as described in Section 9.2.

The LQR controller efficiently computes control commands for linear systems. To apply this control method to miniature airship systems, we use the common method of linearizing the motion model along the desired trajectory. We compute the Jacobians

$$A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_t^{\blacktriangledown}, \mathbf{u}_t^{\blacktriangledown}) \qquad \text{and} \qquad B_t = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_t^{\blacktriangledown}, \mathbf{u}_t^{\blacktriangledown}) \tag{10.8}$$

with respect to the state and the control command, respectively, in a first-order Taylor approximation at the desired state and control given by the trajectory. Even for nonlinear systems such as an airship, this approximation provides good control results as long as the vehicle stays close to the desired trajectory. The Jacobians specify the linearized system

$$(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^{\blacktriangledown}) \approx A_t(\mathbf{x}_t - \mathbf{x}_t^{\blacktriangledown}) + B_t(\mathbf{u}_t - \mathbf{u}_t^{\blacktriangledown}) \tag{10.9}$$

for which the feedback gain matrix G_t and finally the optimal control command

$$\mathbf{u}_t = \mathbf{u}_t^{\blacktriangledown} + G_t(\mathbf{x}_t - \mathbf{x}_t^{\blacktriangledown}) \tag{10.10}$$

can be computed as introduced in Section 9.2.

10.4 Experimental Evaluation

We implemented and evaluated the approach described above in simulation and with the second prototype of our real robotic blimp (see Section 2.1.2) operating in a large indoor environment with two rooms. In our experiments, we consider the task of continuously navigating on a round trip specified in advance by an ordered set of goals, which are shown in Figure 10.3.

In the preparation of the experiment, we learned the parameters of the motion model described in Section 2.6 from about 10 min of manually operated flight observed by a

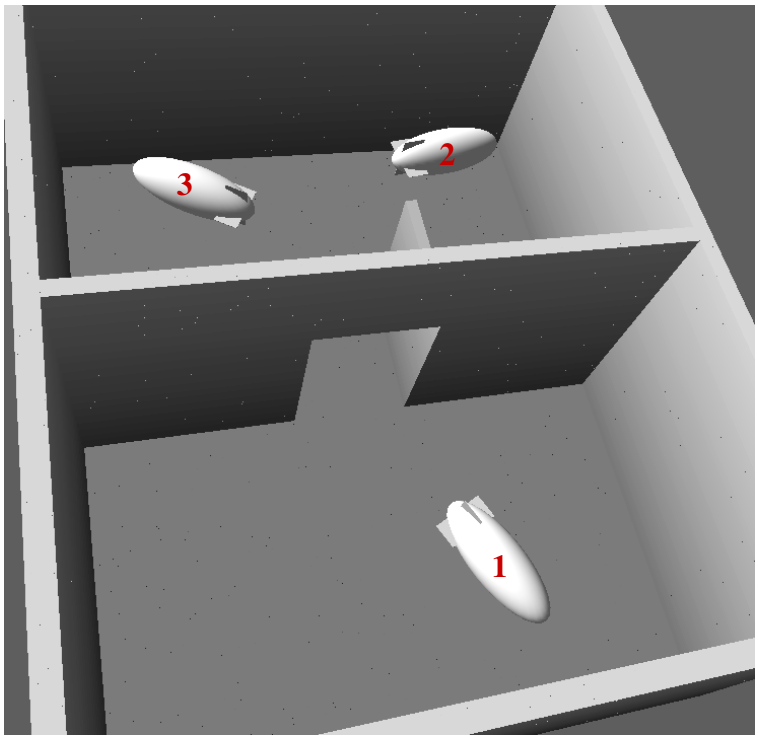


Figure 10.3: The experimental environment consists of two rooms connected by an open door where each room has a size of $8\,\mathrm{m} \times 6\,\mathrm{m}$. The round trip navigation task is defined by three goals, which are set sequentially by the mission control module.

Vicon motion capture system. The parameters P and Q of the LQR controller were chosen according to Bryson's rule. We chose a $0.25\,\mathrm{m}$ and 45° resolution for the low-dimensional A* path generation. To achieve fast online computations, we precalculated the distance map of the environment and the numerically derived Jacobians of the motion model for the typical range of velocities. In all experiments, we set the planning timeout $t_{\max}$ to $1\,\mathrm{s}$. With this setting, the tree-based planner extended the tree by 150 to 500 nodes in each period depending on the initial tree size and the ratio of successful attempts to extend the tree.

The mission control module (see Figure 10.1) continuously checks whether the blimp is approaching the current goal. If the distance to this goal drops below a threshold, it switches to the next goal and provides it to the planner modules. All experiments were run on an Intel® Core™ 2 Duo processor running at $2.53\,\mathrm{GHz}$.

10.4.1 Simulation

Our simulation module described in Section 2.6.4 handles control commands and simulates the motion of the blimp according to the parametric motion model learned from real recorded data of the blimp. Additionally, it provides the simulated position and velocity as state information. Due to the difference in time discretization used in the individual modules, the simulation deviates slightly from the prediction of the planner.

In an extensive experiment, the online simulated blimp traveled for $110\,\mathrm{min}$ on a round trip and reached each of the 3 goals 70 times. For that, it calculated $6{,}628$ trajectories including all re-planning steps. The controller executed all trajectories without any collision. The calculation of the control feedback matrices for a new trajectory took $1.1\,\mathrm{ms}$ on average with a maximum value of $6.8\,\mathrm{ms}$. The 4D A* planner took $14.4\,\mathrm{ms}$ on average with a maximum value of $81.7\,\mathrm{ms}$ for calculating a full path.

We compared our planning algorithm to an RRT planner with goal-biased sampling as proposed by Kim and Ostrowski [84] for airship navigation. We experimentally found that drawing $10\,\%$ of the samples from a Gaussian around the goal was a good trade-off between exploration and exploitation. We also ran the goal-biased planner for $110\,\mathrm{min}$ of online operation on the same round trip during which it failed in 14 of 191 attempts to plan a trajectory to a goal. In case of failure, the goal-biased planner created only a partial trajectory that ran into a dead end and resulted in a collision. The planning times until the trajectory provided by the planner reached the goal are compared in Figure 10.4. While our planning algorithm never needed more than $6\,\mathrm{s}$, the goal-biased sampling resulted in a wide-spread distribution, which caused the planner to fail in 14 attempts.

As shown in Table 10.1, the average travel time for the trajectories planned by our algorithm had a considerably lower standard deviation, since the path-guided sampling turned out to be more goal-directed. In fact, the travel times resulting from our planning algorithm proved to be significantly shorter in a paired t-test with a p-value of $0.6\,\%$.

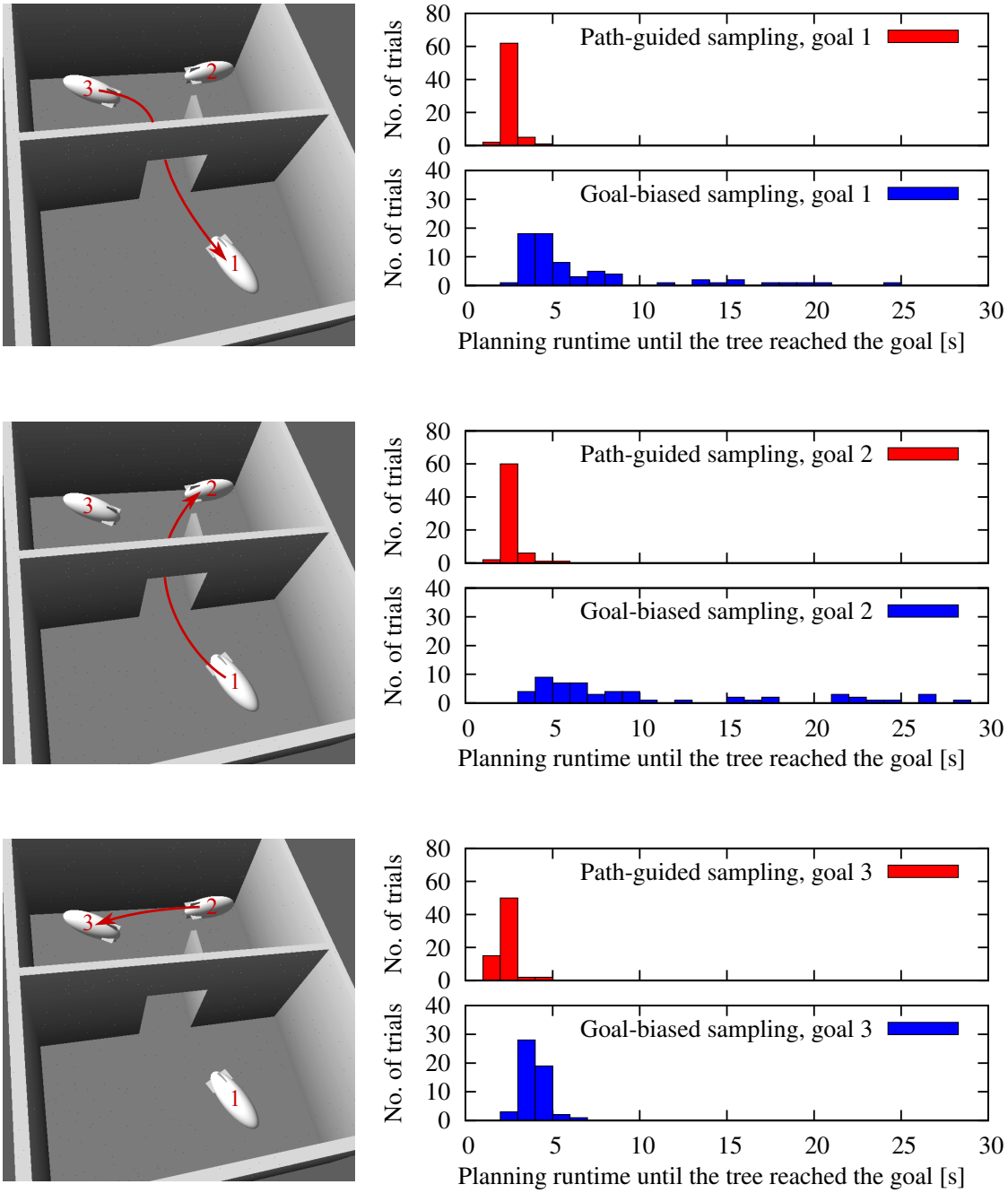


Figure 10.4: Comparison of the planning times for three different goals of our path-guided and the goal-biased planner.

Goal	Path-guided		Goal-biased	
	Mean	StdDev	Mean	StdDev
1	35.2	13.6	35.1	19.1
2	51.4	10.0	62.7	18.2
3	18.4	6.9	22.4	11.7

Table 10.1: Comparison of the travel times of the planned trajectories using path-guided and goal-biased sampling. All units are seconds.

10.4.2 Real Blimp

We performed an extensive experiment with the second prototype of our robotic blimp (see Section 2.1.2) showing that the models used for planning and control are realistic and that our approach is able to deal with real noise and moderate modeling approximations.

For our experiments, we did not use the on-board sensors to localize the blimp. Instead, we localized the blimp using a Motion Analysis motion capture system with eight digital Raptor-E cameras tracking four retroreflective markers mounted around the gondola of the blimp (see Section 2.5). Due to practical reasons, we only built up the door frame and the contour of the door, as building up the all obstacles and walls would prevent tracking the blimp with a reasonable number of motion capture cameras. Since the pose estimate provided by the motion capture system is very accurate (in our setting the error is typically below 3 mm), we additionally applied online collision checking based on motion capture pose estimates and the map.

In our experiment, the blimp autonomously traveled on the round trip for about 20 min and passed 28 goals without any collision. In this setup, the problem of Inevitable Collision States (ICS) [11] did not arise due to the quick planning and the comparably low velocity of the goal states. Figure 10.5 shows the blimp passing the narrow passage of the planning task. Two exemplary trajectories generated by our planning approach during operation are shown in Figure 10.6. In our experiment, the root mean square (RMS) translational deviation from the trajectory was 0.24 m and the RMS deviation in the yaw-orientation was $8.6°$. This is due to air motion caused by the air conditioning system of the adjacent clean rooms and the moderate approximations of the motion model. The RMS deviation from zero roll was $0.35°$. Since the velocity profile on the 4D path guides the sampler to move at low velocities in the vicinity of obstacles, the blimp passed the door slowly in the majority of those safety-critical situations. Thus, the controller operated the blimp with a low deviation from the desired trajectory when passing the door.

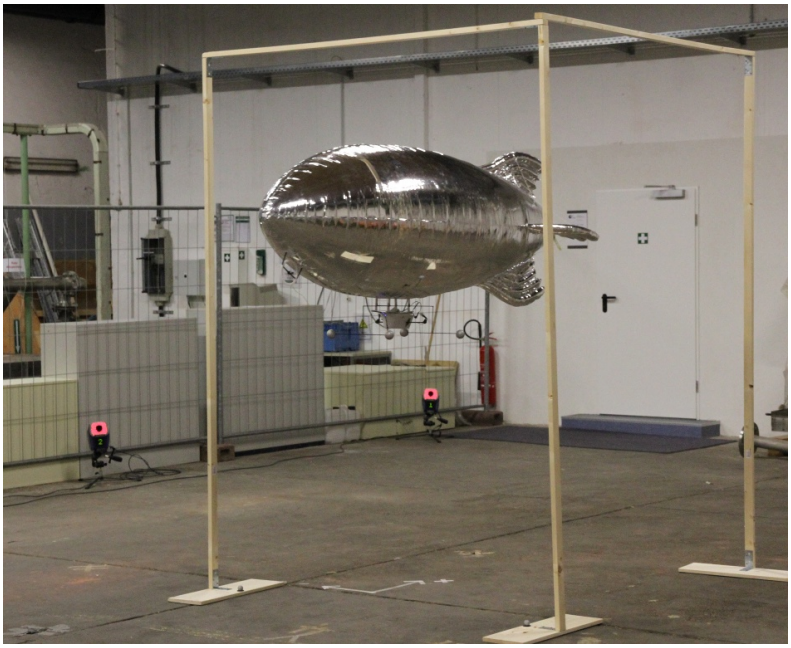


Figure 10.5: The robotic indoor blimp operating in the experimental setting observed by cameras of the motion capture system. The blimp is passing the safety-critical narrow passage with a low velocity so that the controller can accurately keep the blimp on the trajectory in this situation.

10.5 Conclusions

In this chapter, we presented an approach to autonomous navigation of a blimp in a known indoor environment including motion capture state estimation. To efficiently approximate the high-dimensional nonlinear kinodynamic motion planning, we apply a multi-stage planning technique. In the first stage, a collision-free path is generated through A* search on a low-dimensional subspace of the state space. We utilize this path to efficiently generate kinematically feasible trajectories by goal-directed, path-guided tree planning in the full 12-dimensional state space. In contrast to other approaches, our algorithm selects optimal actions towards sampled subgoals and is able to quickly provide a possibly partial trajectory, which is extended in the consecutive planning steps. Including a motion capture state estimate, a mission control module, and an LQR controller, our approach successfully controlled a robotic blimp. We performed extensive experiments in simulation and with a real robotic blimp. In all experiments, our navigation system efficiently and reliably operated the blimp and outperformed a standard goal-biased RRT planner.

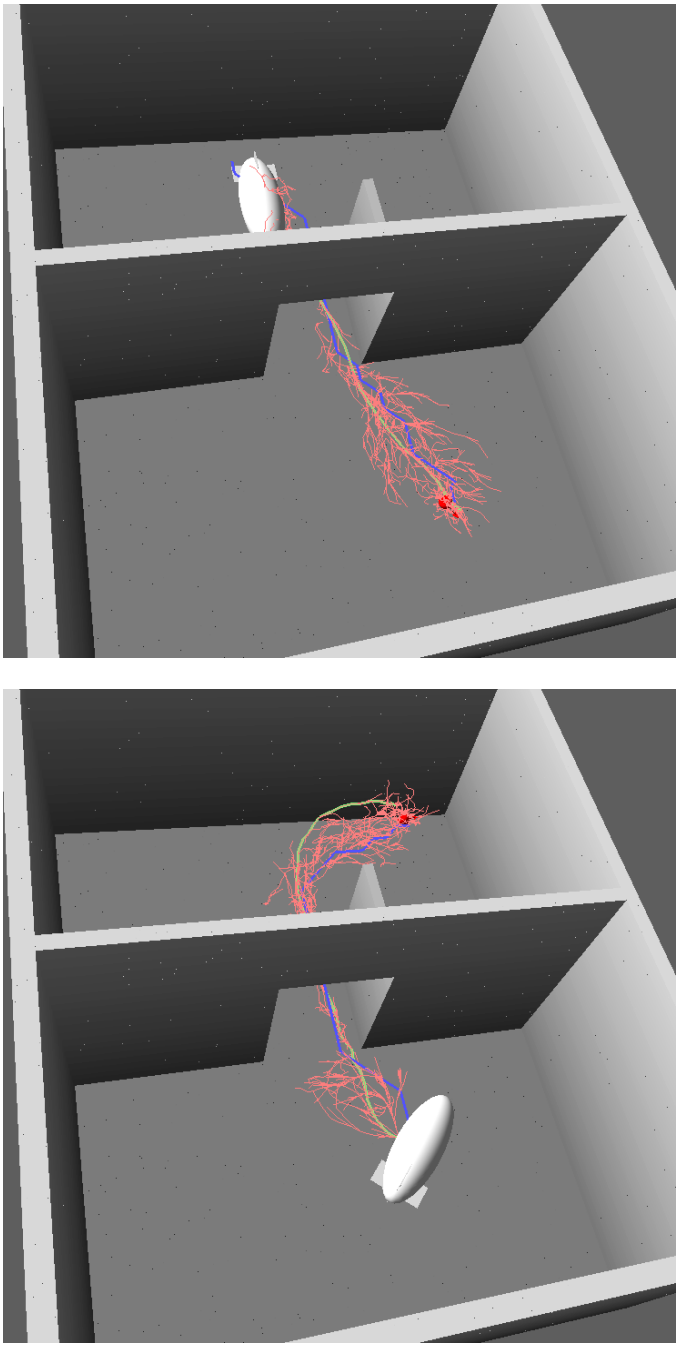


Figure 10.6: Two exemplary trajectories generated by our path-guided planner during the experiment with the real robotic blimp. The goals are shown as a big red ball with an arrow indicating the desired orientation of the blimp. The four-dimensional A* path is shown as a thick blue line, the tree built by the tree planner is shown in red, and the chosen branch is marked by a thick yellow line.

Discussion

Chapter 11

Conclusions and Future Work

In this thesis, we proposed several novel approaches to address the challenges of miniaturized, low-cost, and resource-constrained embedded systems in mobile devices. Using the example of autonomous navigation for miniature airships, we presented techniques that can deal with the common challenges of such systems, namely limited computational power, imperfect and weak actuators, and imprecise sensors.

We introduced methods for robust, efficient, and accurate self-localization, which is a fundamental problem in the domain of autonomous navigation for mobile robots. In particular, we investigated probabilistic sensor data fusion in recursive state estimators and presented an implementation of the particle filter that is suitable for localizing an airship with nonlinear system dynamics and ambiguous sensor measurements in its high-dimensional state space. Our localization method includes carefully designed probabilistic sensor models for small sensors that are applicable on miniature indoor airships.

We discussed a novel sonar sensor model that explicitly considers the characteristics of tiny sonar sensors with large opening angles. In contrast to other models, our approach is rigorously based on the physics of the measurement process and takes into account the ambiguity of measurements, which is induced by the signal reflection by objects with different sizes and distances. We developed a novel probabilistic model for air flow sensors, which is suitable for dead-reckoning odometry as well as for probabilistic state estimation. In contrast to other approaches, we explicitly consider the measurement uncertainty and the heteroscedastic characteristics of the thermal air flow sensors and address them using appropriate regression techniques. Additionally, we presented an effective probabilistic model for fusing the orientation estimates of an IMU into the state estimate of our localization approach. Our model has proven to enable a significantly more accurate localization than a standard model.

Furthermore, we showed that the localization accuracy can be significantly improved through the simultaneous estimation of the parameters of the system dynamics, especially if only sparse and imprecise sensor information is available. We proposed a general approach that can estimate initially unknown and possibly changing parameters of the motion model during localization and avoids oscillations of the parameter estimates.

We developed an efficient probabilistic odometry motion model for flying vehicles. Our novel motion model can compute the odometry from air flow sensors and the IMU in a linear way and therefore is computationally much more efficient than the standard control motion model, which is based on physical simulations. Additionally, we introduced a general approach to account for the temporal correlations of odometry errors in the prediction step of the particle filter. Our odometry motion model decreases the dimensionality of the state space in the recursive filter and therefore enables an accurate online localization for miniature airships.

In addition to our solutions to online localization, we addressed the task of planning and closed-loop control for autonomous navigation of mobile robots. In the context of miniature airships, this is especially challenging, since their complex and nonlinear system dynamics paired with their weak actuators demand to solve the kinodynamic motion planning problem in the high-dimensional state space. We presented a solution to accurate and efficient planning and control by applying a multi-stage planning technique and an LQR controller. Our planning algorithm efficiently generates kinematically feasible trajectories through path-guided sampling based on a low-dimensional A* path. In contrast to other approaches, our algorithm selects optimal controls towards sampled subgoals and is able to quickly provide a possibly partial trajectory, which is extended in subsequent planning steps.

We implemented, thoroughly tested, and evaluated our approaches presented in this thesis. In extensive experiments, we demonstrated that our techniques enable a miniature indoor blimp to accurately localize itself in a known, complex indoor environment in an online fashion. Furthermore, our planning and control system has proven to enable a reliable operation of the blimp in a real-world setting. Additionally, we showed that our solutions significantly outperform comparable state-of-the-art techniques for autonomous navigation.

In summary, this thesis answers the following questions:

- How can a self-localizing mobile robot maximize the amount of information extracted from imprecise and ambiguous measurements obtained by imperfect sensors?

- How to deal with unknown or changing parameters of the system dynamics during state estimation?

- How to efficiently approximate the high-dimensional state estimation while providing a robust and accurate solution to online self-localization?

- How can a mobile robot equipped with weak actuators effectively plan suitable actions in an online fashion to autonomously reach its desired goal?

We demonstrated that the methods presented in this thesis enable robots to effectively and reliably operate in their environment, and we believe that the proposed solutions are relevant for future low-cost, small, and resource-constrained embedded systems that are useful in industrial settings and everyday life.

Although we presented a modular robotic blimp together with a flexible and powerful autonomous navigation system and encouraging experimental results, there are several possibilities for extensions that remain for future investigation.

For example, we see additional potential in considering the influence of multiple reflections of the ultrasound signal in the environment. Our current approach considers the direct reflection of the signal from individual objects in different distances. Although our sensor model has proven to be robust against multiple reflections, a modeling of the corresponding effects could be beneficial to accurate and robust localization. However, this would require a more detailed map of the environment, which also provides normals of the surfaces of objects and introduces additional complexity in the computations.

Another extension of our work would be to expand our probabilistic model for air flow sensors that can measure the flow in two or three dimensions. For example, the two-dimensional flow sensors of Cubukcu et al. [34] could replace the two one-dimensional flow sensors mounted on top of the blimp providing comparable measurements at a lower weight. This could make it possible to attach more of these sensors to the blimp in order to exploit the capabilities of our probabilistic air flow sensor model to deal with $L \geq 3$ sensors for an increased robustness to measurement noise. In addition, the combination with highly integrated optical flow sensors, such as the sensors applied in optical mice, could be beneficial to an accurate and robust odometry for flying vehicles.

Furthermore, an interesting extension of our approach to simultaneous parameter estimation would be the estimation of wind in the environment. We envision an effective approach that detects strong differences between the motion predicted by the physical simulation-based control motion model and the motion predicted by our air flow odometry motion model. This could enable a flying robot to estimate the wind field in its surrounding and to take into account the expected disturbances during planning and control.

Another promising direction is the consideration of the temporal correlation of measurement noise. This temporal correlation is often caused by non-modeled systematic errors, which especially occur in the field of miniature and low-cost devices. In Section 8.3, we introduced a corresponding extension for the particle filter. Since various implementations of the recursive Bayes filter are popular for embedded systems, a similar extension to other implementations such as the Kalman filters would cover a wide range of applications.

Moreover, one could combine the work of Part I and II by considering the autonomous navigation of blimps with self-localization using its on-board sensors. Especially

in the context of localization with sparse and imprecise sensor information, this usually requires to take into account the uncertainty of the state estimates during planning and control. The general formulation of this problem is known as the partially observable Markov decision process (POMDP) [77], which requires efficient approximations to be computationally tractable [134, 145, 146]. Future research could deal with developing efficient POMDP approximations that exploit the characteristics of indoor airship navigation.

Another interesting direction for future research is the consideration of autonomous navigation algorithms in the presence of transient faults. Such discrete errors on bit-level of digital circuits are not only an issue in planetary exploration or disaster scenarios under high radiation but also increasingly occur in the shrinking nanometer technology of modern microprocessors with aggressive supply voltage down-scaling to save energy. In the context of autonomous navigation, we envision the effective combination of partially fault-tolerant algorithms with techniques for error detection and recovery in software or hardware. This can extend our recent work on fault-injection and fault-tolerant algorithms [139, 140, 151, 158] and combine it with the work presented in this thesis.

Of course, our future research is not limited to robotics. Today, miniature low-cost embedded devices are omnipresent in industry and everyday life. In this context, we believe that we can exploit the techniques and the insights of our work in designing accurate and efficient algorithms for state estimation and planning in a broad area of applications.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] P. Abbeel, A. Coates, M. Montemerlo, A.Y. Ng, and S. Thrun. Discriminative training of Kalman filters. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[2] R. Al-Jarrah and H. Roth. Developed blimp robot based on ultrasonic sensors using possibilities distribution and fuzzy logic. *Journal of Automation and Control Engineering*, 1(2), 2013.

[3] D. Alspach and H.W. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

[4] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. 2D simultaneous localization and mapping for micro aerial vehicles. In *Proc. of the European Micro Aerial Vehicle Conf. (EMAV)*, 2006.

[5] A. Bachrach, S. Prentice, R. He, and N. Roy. RANGE - Robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.

[6] Y. Bar-Shalom, T. Kirubarajan, and X. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2002.

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

[8] 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.

[9] M. Beinhofer, J. Müller, and W. Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics & Autonomous Systems*, 2013. doi:10.1016/j.robot.2012.08.009.

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

[11] K.E. Bekris and L.E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[12] *BitBake*. BerliOS, 2013. URL http://developer.berlios.de/projects/bitbake/.

[13] S. Bermúdez i Badia, P. Pyk, and P.F.M.J. Verschure. A biologically based flight control system for a blimp-based UAV. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[14] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2005.

[15] M. Billinghurst and H. Kato. Collaborative augmented reality. *Communications of the ACM*, 45(7):64–70, 2002.

[16] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.

[17] M.K. Brown. Feature extraction techniques for recognizing solid objects with an ultrasonic range sensor. *IEEE Jornal of Robotics and Automation*, 1(4):191–205, 1985.

[18] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[19] M. Bryson and S. Sukkarieh. Vehicle model aided inertial navigation for a UAV using low-cost sensors. In *Proc. of the Australasian Conf. on Robotics & Automation (ACRA)*, 2004.

[20] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 1996.

[21] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic Markov localization approach. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[22] B. Burns and O. Brock. Single-query entropy-guided path planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[23] A. Cao and J. Borenstein. Experimental characterization of Polaroid ultrasonic sensors in single and phased array configuration. In *Proc. of the UGV Technology Conf. at the SPIE AeroSense Symposium*, 2002.

[24] H. Chao and Y.Q. Chen. Surface wind profile measurement using multiple small unmanned aerial vehicles. In *Proc. of the American Control Conf.*, 2010.

[25] P. Cheng and S.M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[26] J. Chestnutt and J. Kuffner. A tiered planning strategy for biped navigation. In *Proc. of the IEEE - RAS / RSJ Conf. on Humanoid Robots*, 2004.

[27] A. Cho, J. Kim, S. Lee, and C. Kee. Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube. *IEEE Transactions on Aerospace and Electronic Systems*, 47(1):109–117, 2011.

[28] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavarki, and S. Thrun. *Principles of Robot Motion Planning*. MIT-Press, 2005.

[29] J. Conroy, G. Gremillion, B. Ranganathan, and J.S. Humpert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Journal of Autonomous Robots*, 27(3):189–198, 2009.

[30] J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1989.

[31] I.A. Şucan and L.E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Proc. of the Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2008.

[32] I.A. Şucan and L.E. Kavraki. On the implementation of single-query sampling-based motion planners. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[33] I.A. Şucan, J.F. Kruse, M. Yim, and L.E. Kavraki. Kinodynamic motion planning with hardware demonstrations. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[34] A.S. Cubukcu, E. Zernickel, U. Buerklin, and G.A. Urban. A 2D thermal flow sensor with sub-mW power consumption. *Sensors and Actuators A: Physical*, 163:449–456, 2010.

[35] 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)*, 1999.

[36] *SRF10 Ultrasonic range finder*. Devantech Ltd, Attleborough, Norfolk, UK, 2008. URL http://www.robot-electronics.co.uk/htm/srf10tech.htm.

[37] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, 2006.

[38] M. Dille, B. Grocholsky, and S. Singh. Outdoor downward-facing optical flow odometry with commodity sensors. In *Proc. of the Int. Conf. on Field and Service Robots (FSR)*, 2009.

[39] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, 1993.

[40] A. Doucet, N. de Freitas, and N. Gordan. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[41] M.L. Eaton. *Multivariate Statistics: A Vector Space Approach*. Wiley, 1983.

[42] A. Elfes, S.S. Bueno, M. Bergman, and J.J.G. Ramos. A semi-autonomous robotic airship for environmental monitoring missions. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[43] A.I. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. In *Proc. of the Int. Conf. on Machine Learning*, 2004.

[44] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing UAV. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[45] H. Fei, R. Zhu, Z. Zhou, and J. Wang. Aircraft flight parameter detection based on a neural network using multiple hot-film flow speed sensors. *Smart Materials and Structures*, 16(4):1239–1245, 2007.

[46] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *Int. Journal of Robust and Nonlinear Control*, 18 (8):816–830, 2008.

[47] P. Fitzpatrick, G. Metta, and L. Natale. Towards long-lived robot genes. *Robotics & Autonomous Systems*, 56(1):29–45, 2008.

[48] J. Folkesson. Robustness of the quadratic antiparticle filter for robot localization. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2011.

[49] J. Folkesson. The antiparticle filter - an adaptive nonlinear estimator. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2011.

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

[51] J.H. Freidman, J.L. Bentley, and R.A Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[52] H. Fukushima, S. Ryosuke, M. Fumitoshi, Y. Hada, K. Kawabata, and H. Asama. Model predictive control of an autonomous blimp with input and output constraints. In *Proc. of the Int. Conf. on Control Applications*, 2006.

[53] J. Garimort, A. Hornung, and M. Bennewitz. Humanoid navigation with dynamic footstep plans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[54] B.P. Gerkey. *The amcl localization package in ROS*. Willow Garage, 2013. URL http://www.ros.org/wiki/amcl.

[55] S.B.V. Gomes and J.G. Jr. Ramos. Airship dynamic modeling for autonomous operation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[56] C. Gonsior. *Verbesserte Bewegungsmodelle für die Lokalisierung von Miniaturluftschiffen*. BSc thesis, University of Freiburg, Department of Computer Science, 2009. In German.

[57] W.E. Green, K.W. Sevcik, and P.Y. Oh. A competition to identify key challenges for unmanned aerial robots in near-earth environments. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, 2005.

[58] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1): 34–46, 2007.

[59] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[60] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[61] S. Grzonka, G. Grisetti, and W. Burgard. A fully autonomous indoor quadrotor. *IEEE Transactions on Robotics*, 8(1):90–100, 2012.

[62] *Verdex pro XL6P COM*. Gumstix Inc., Portola Valley, CA, USA, 2013. URL http://www.gumstix.com.

[63] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[64] 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.

[65] Q. Hada, K. Kawabata, H. Kaetsu, and H. Asama. Autonomous blimp system for aerial infrastructure. In *Proc. of the Int. Conf. on Ubiquitous Robots and Ambient Intelligence*, 2005.

[66] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*, 1968.

[67] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[68] F. Höflinger, J. Müller, M. Törk, L.M. Reindl, and W. Burgard. A wireless micro inertial measurement unit (IMU). In *Proc. of the IEEE Int. Instrumentation and Measurement Technology Conf. (I2MTC)*, 2012.

[69] F. Höflinger, J. Müller, R. Zhang, W. Burgard, and L.M. Reindl. A wireless micro inertial measurement unit (IMU). *IEEE Transactions on Instrumentation & Measurement*, 2013. Accepted for publication.

[70] A. Hornung, K.M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[71] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[72] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1997.

[73] E. Hygounenc, I.-K. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project at LAAS/CNRS: Achievements in flight control and terrain mapping. *Int. Journal of Robotics Research*, 23(4):473–511, 2004.

[74] H. Ishida. Blimp robot for three-dimensional gas distribution mapping in indoor environment. In *AIP Conf. Proceedings*, volume 1137, 2009.

[75] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conf. (ACC)*, 1995.

[76] A. Kaboli, M. Bowling, and P. Musilek. Bayesian calibration for Monte Carlo localization. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2006.

[77] L.P. Kaelbling, M.L. Littmann, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[78] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[79] R.E. Kalman. A new approach to linear filtering and prediction problems. *ASME-Journal of Basic Engineering*, 82(1):35–45, 1960.

[80] G. Kantor, D. Wettergreen, J.P. Ostrowski, and S. Singh. Collection of environmental data from an airship platform. In *Proc. of the SPIE Conf. on Sensor Fusion and Decentralized Control in Robotic Systems*, 2001.

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

[82] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[83] J. Keshavan and J.S. Humpert. MAV stability augmentation using weighted outputs from distributed hair sensor arrays. In *Proc. of the American Control Conference*, 2010.

[84] J. Kim and J.P. Ostrowski. Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[85] S. Kim and S. Lee. Robust velocity estimation of an omnidirectional mobile robot using a polygonal array of optical mice. *Int. Journal of Control, Automation and Systems*, 6(5):713–721, 2008.

[86] N. Kirchner and T. Furukawa. Infrared localization for indoor UAVs. In *Proc. of the 1st Int. Conf. on Sensing Technology*, 2005.

[87] J. Ko, D.J. Klein, D. Fox, and D. Hähnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[88] J. Ko, D.J. Klein, D. Fox, and D. Hähnel. GP-UKF: Unscented Kalman filters with Gaussian process prediction and observation models. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[89] N. Kohler. Bewegungsplanung für autonome Luftschiffe. Master's thesis, University of Freiburg, Department of Computer Science, 2010. In German.

[90] K. Konolige and K. Chou. Markov localization using correlation. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1999.

[91] J.H. Kotecha and P.M. Djuric. Gaussian particle filtering. *IEEE Transactions on Signal Processing*, 51(10):2592–2601, 2003.

[92] M. Kruusmaa, G. Toming, J. Salumäe, J. Ježov, and A. Ernits. Swimming speed control and on-board flow sensing of an artificial trout. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

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

[94] 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.

[95] C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[96] A.M. Ladd and L.E. Kavraki. Fast tree-based expoloration of state space for robots with dynamics. In *Algorithmic Foundations of Robotics VI*. Springer, STAR 17, 2005.

[97] A.M. Ladd and L.E. Kavraki. Motion planning in the presence of drift, underactuation and discrete systems changes. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[98] H. Lamb. The inertia coefficients of an ellipsoid moving in fluid. Technical Report Reports and Memoranda, No. 623, British Aeronautical Research Committee, 1918.

[99] T.D. Larsen, M. Bak, N. Andersen, and O. Ravn. Location estimation for an autonomously guided vehicle using an augmented Kalman filter to autocalibrate the odometry. In *Proc. of the SPIE Conf. on Mobile Robots*, 1998.

[100] S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, 1998.

[101] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[102] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, 2001.

[103] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2000.

[104] J.J. Leonard and H.F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, 1992.

[105] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 10. Springer Verlag, 2001.

[106] J.S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, 1996.

[107] Y. Liu, Z. Pan, D. Stirling, and F. Naghdy. Q-learning for navigation control of an autonomous blimp. In *Proc. of the Australasian Conf. on Robotics & Automation (ACRA)*, 2009.

[108] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart. Simultaneous localization and odometry calibration for mobile robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

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

[110] F. Mayer, A. Häberli, G. Ofner, H. Jacobs, O. Paul, and H. Baltes. Single-chip CMOS anemometer. In *Technical Digest of the IEEE Int. Electron Devices Meeting (IEDM)*, 1997.

[111] D. Meschede. *Gerthsen Physik*. Springer Verlag, 2004. In German.

[112] J. Meyer, M. Kuderer, J. Müller, and W. Burgard. Online marker labeling for automatic skeleton tracking in optical motion capture. In *Proc. of the ICRA Workshop on Computational Techniques in Natural Motion Analysis and Reconstruction*, 2013.

[113] M. Montemerlo, N. Roy, S. Thrun, D. Hähnel, C. Stachniss, and J. Glover. *CARMEN – The Carnegie Mellon Robot Navigation Toolkit*, 2002. URL http://carmen.sourceforge.net.

[114] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2002.

[115] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9 (2):61–74, 1988.

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

[117] A.B. Moutinho. *Modeling and Nonlinear Control for Airship Autonomous Flight*. PhD thesis, University of Lisbon, 2007.

[118] J. Müller. Techniken für die Navigation autonomer Luftschiffe. Master's thesis, University of Freiburg, Department of Computer Science, 2008. In German.

[119] J. Müller and W. Burgard. Efficient probabilistic localization for autonomous indoor airships using sonar, air flow, and IMU sensors. *Advanced Robotics*, 27 (9):711–724, 2013. doi:10.1080/01691864.2013.779005.

[120] J. Müller, C. Stachniss, K.O. Arras, and W. Burgard. Socially inspired motion planning for mobile robots in populated environments. In *Proc. of the Int. Conf. on Cognitive Systems (CogSys)*, 2008.

[121] J. Müller, A. Rottmann, L.M. Reindl, and W. Burgard. A probabilistic sonar sensor model for robust localization of a small-size blimp in indoor environments using a particle filter. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[122] J. Müller, C. Gonsior, and W. Burgard. Improved Monte Carlo localization of autonomous robots through simultaneous estimation of motion model parameters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[123] J. Müller, N. Kohler, and W. Burgard. Autonomous miniature blimp navigation with online motion planning and re-planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[124] J. Müller, O. Paul, and W. Burgard. Probabilistic velocity estimation for autonomous miniature airships using thermal air flow sensors. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[125] P.N. Newman. MOOS – mission orientated operating suite. Technical Report 08, Massachusetts Institute of Technology, 2008.

[126] Bureau of Enquiry and Analysis for Civil Aviation Safety (BEA). Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro – Paris, 2012.

[127] *The build framework for embedded Linux*. OpenEmbedded, 2013. URL http://www.openembedded.org.

[128] P. Pfaff, C. Plagemann, and W. Burgard. Improved likelihood models for probabilistic localization based on range scans. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[129] C. Plagemann and W. Burgard. Sequential parameter estimation for fault diagnosis in mobile robots using particle filters. In *Autonome Mobile Systeme 2005 (AMS)*. Springer Verlag, 2005.

[130] C. Plagemann, C. Stachniss, and W. Burgard. Efficient failure detection for mobile robots using mixed-abstraction particle filters. In H.I. Christensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*. Springer Verlag, 2006.

[131] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A nonparametric Bayesian measurement model for range finders. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[132] E. Plaku, L.E. Kavraki, and M.Y. Vardi. Discrete search leading continuous exploration for kinodynamic motion planning. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[133] E. Plaku, L.E. Kavraki, and M.Y. Vardi. Impact of workspace decompositions on discrete search leading continuous exploration (DSLX) motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[134] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Int. Journal of Robotics Research*, 8(11-12):1448–1465, 2009.

[135] M. Quigley, B.P. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *Proc. of the ICRA Workshop on Open Source Software*, 2009.

[136] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[137] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of the Symposium on Foundations of Computer Science*, 1979.

[138] M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[139] A. Riefert, J. Müller, M. Sauer, W. Burgard, and B. Becker. Identification of critical variables using an FPGA-based fault injection framework. In *Proc. of the Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TUZ)*, 2013.

[140] A. Riefert, J. Müller, M. Sauer, W. Burgard, and B. Becker. Identification of critical variables using an FPGA-based fault injection framework. In *Proc. of the IEEE VLSI Test Symposium (VTS)*, 2013.

[141] A. Rottmann. *Approaches to Online Reinforcement Learning for Miniature Airships*. PhD thesis, University of Freiburg, Department of Computer Science, 2012.

[142] A. Rottmann, C. Plagemann, P Hilgers, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[143] A. Rottmann, M. Sippel, T. Zitterell, W. Burgard, L.M. Reindl, and C. Scholl. Towards an experimental autonomous blimp platform. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2007.

[144] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[145] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation – robot motion with uncertainty. In *Proc. of the AAAI Fall Symposium: Planning with POMDPs*, 1998.

[146] N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research (JAIR)*, 23(1):1–40, 2005.

[147] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, second edition, 2003.

[148] A.J. Rutkowski, M.M. Miller, R.D. Quinn, and M.A. Willis. Egomotion estimation with optic flow and air velocity sensors. *Biological Cybernetics*, 104(6): 351–367, 2011.

[149] H. Saiki, T. Fukao, T. Urakubo, and T. Kohno. Hovering control of outdoor blimp robots based on path following. In *Proc. of the IEEE Int. Conf. on Control Applications (CCA)*, 2010.

[150] G. Sánchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics Research*, volume 6. Springer Verlag, 2003.

[151] M. Sauer, V. Tomashevich, J. Müller, M. Lewis, A. Spilla, I. Polian, B. Becker, and W. Burgard. An FPGA-based framework for run-time injection and analysis of soft errors in microprocessors. In *Proc. of the IEEE Int. On-Line Testing Symposium (IOLTS)*, 2011.

[152] C. Schroeter, A. Koenig, H.-J. Boehme, and H.-M. Gross. Multi-sensor Monte Carlo-localization combining omni-vision and sonar range sensors. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2005.

[153] *SDP600 Series – Differential Pressure Sensors*. Sensirion AG, Stäfa, Switzerland, 2012. URL http://www.sensirion.com/en/products/differential-pressure-sensors/differential-pressure-sensor-sdp600-series/.

[154] S. Shen, M. Nathan, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[155] R. Simmons and D. James. *Inter Process Communication: A Reference Manual (for IPC version 3.9)*, 2011. URL http://www.cs.cmu.edu/afs/cs/project/TCA/www/ipc/.

[156] M. Sippel, A. Abduhl-Majeed, W. Kuntz, and L.M. Reindl. Enhancing accuracy of an indoor radar by the implementation of a quaternion- and unscented Kalman filter-based lightweight, planar, strapdown IMU. In *Proc. of the European Navigation Conf. (ENC-GNSS)*, 2008.

[157] F. Sittel, J. Müller, and W. Burgard. Computing velocities and accelerations from a pose time sequence in three-dimensional space. Technical Report 272, University of Freiburg, Department of Computer Science, 2013.

[158] A. Spilla, I. Polian, J. Müller, M. Lewis, V. Tomashevich, B. Becker, and W. Burgard. Run-time soft error injection and testing of a microprocessor using FPGAs. In *Proc. of the Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TUZ)*, 2011.

[159] M. Sridharan, G. Kuhlmann, and P. Stone. Practical vision-based Monte Carlo localization on a legged robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[160] 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.

[161] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3D using attitude and noisy vision sensors. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[162] H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular SLAM: Why filter? In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[163] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. Journal of Robotics Research*, 21(4):311–330, 2002.

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

[165] S. Thrun. Particle filters in robotics. In *Proc. of the 17th Annual Conf. on Uncertainty in AI (UAI)*, 2002.

[166] S. Thrun. Learning occupancy grid maps with forward sensor models. *Journal of Autonomous Robots*, 15(2):111–127, 2003.

[167] S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distribution. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2000.

[168] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

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

[170] D.H. Titterton and J.L. Weston. *Strapdown Inertial Navigation Technology*, volume 17 of *IEE radar, sonar, navigation and avionics series*. Institution of Electrical Engineers, second edition, 2004.

[171] H. Tokutake, S. Sunada, and J. Fujinaga. Attitude control of a small UAV using a flow sensor system. *Journal of System Design and Dynamics*, 5(1):1–16, 2011.

[172] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[173] K.I. Tsianos, I.A. Şucan, and L.E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1(1):2–11, 2007.

[174] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[175] R. Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, University of Stellenbosch, 2004.

[176] L. Wasserman. *All of Nonparametric Statistics*. Springer Verlag, 2006.

[177] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[178] J. Wendeberg, J. Müller, C. Schindelhauer, and W. Burgard. Robust tracking of a mobile beacon using time differences of arrival with simultaneous calibration of receiver positions. In *Proc. of the Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*, 2012.

[179] R.J. Woods, S. Avadhanula, E. Steltz, M. Seeman, J. Entwistle, A. Bachrach, G. Barrows, S. Sanders, and R.S. Fearing. An autonomous palm-sized gliding micro air vehicle. *IEEE Robotics & Automation Magazine*, 14(2):82–91, 2007.

[180] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[181] Y. Yang, S. Pandya, J. Chen, J. Engel, N. Chen, and C. Liu. Micromachined hot-wire boundary layer flow imaging array. In *Proc. of CANEUS Conf.*, 2006.

[182] T.N. Yap and R. Shelton. Simultaneous learning of motion and sensor model parameters for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[183] J.C. Zufferey, A. Guanella, A. Beyeler, and D. Floreano. Flying over the reality gap: From simulated to real indoor airships. *Journal of Autonomous Robots*, 21 (3):243–254, 2006.