Stochastic Filtering on Mobile Devices in Complex Dynamic Environments

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Stefan Tasse

Dortmund

2014

Tag der mündlichen Prüfung: 27.03.2014

Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter: Prof. Dr.-Ing. Uwe Schwiegelshohn
(Technische Universität Dortmund)
Prof. Dr. sc. nat. Hans-Dieter Burkhard
(Humboldt-Universität zu Berlin)

Acknowledgments

As any long-term project, this dissertation would have been impossible without the support of many other people. I therefore thank all those who helped by giving me their patience, constructive criticism, and valuable advice.

My foremost thanks go to Prof. Dr.-Ing. Uwe Schwiegelshohn for providing me with the opportunity to do research and work at the Robotics Research Institute at the TU Dortmund University. He originally brought me into contact with robotics, made my interest grow into enthusiasm, and allowed me the freedom to follow my own research goals. Furthermore, I am thankful to Prof. Dr. sc. nat. Hans-Dieter Burkhard, who agreed to advise and review my dissertation. His work and support for robotics research and the RoboCup initiative have been very important for me personally and for the German robotics research community in general.

Essential for this productive and inspiring environment have always been my co-workers and fellow researchers at the institute. I would like to thank them all for the great cooperation, for those many professional discussions, and for the friendly and personal atmosphere. My special thanks is to Sören Kerner, Matthias Hofmann, Oliver Urbann, Alexander Papaspyrou, Christian Grimme and Alexander Föllig. I also thank all my students, whose enthusiasm in the course of project groups and theses rewarded all our efforts and kept the RoboCup project successful.

Finally, I would like to thank my friends and family, foremost of all my wife Christiane, for the never-ending moral support and patience. Without you this work would not have been finished.

This dissertation is based upon the following publications. Those are decisively contributed to by the author, and supported by the fellow co-authors, who provided extensive assistance in creating and verifying the implementations and evaluations. Papers and journals published up to 2012 are written under the birth name Czarnetzki, and from then on under the married name Tasse. At the time of this dissertation's completion, all works are reviewed and published.

S. Czarnetzki, S. Kerner, and O. Urbann. Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems - Humanoid Soccer Robots*, 57(8):839 – 845, 2009.

S. Czarnetzki, S. Kerner, and D. Klagges. Combining Key Frame Based Motion Design with Controlled Movement Execution. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary (editors), *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 58–68. Springer Berlin / Heidelberg, 2010.

S. Czarnetzki, S. Kerner, and O. Urbann. Applying Dynamic Walking Control for Biped Robots. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary (editors), *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 69–80. Springer Berlin / Heidelberg, 2010.

S. Czarnetzki and C. Rohde. Handling Heterogeneous Information Sources for Multi-Robot Sensor Fusion. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010)*, pages 133 – 138, Salt Lake City, Utah, September 2010.

B. Künne, J. Eggert, and S. Czarnetzki. Condition Monitoring in Intralogistic Systems by the Utilization of Mobile Measurement Units. In *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL 2010)*, Hong Kong, China, August 2010. IEEE.

S. Czarnetzki, M. Hegele, and S. Kerner. Odometry Correction for Humanoid Robots Using Optical Sensors. In Javier Ruiz-del Solar, Eric Chown, and Paul Plöger (editors), *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 48–59. Springer Berlin / Heidelberg, 2011.

S. Czarnetzki, S. Kerner, and M. Kruse. Real-Time Active Vision by Entropy Minimization Applied to Localization. In Javier Ruiz-del Solar, Eric Chown, and Paul Plöger (editors), *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 266–277. Springer Berlin / Heidelberg, 2011.

S. Czarnetzki, S. Kerner, and P. Szcypior. Learning from Demonstration - Automatic Generation of Extended Behavior Networks for Autonomous Robots from an Expert's Demonstration. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 394–400. INSTICC Press, 2011.

D. Hauschildt, S. Kerner, S. Tasse, and O. Urbann. Multi Body Kalman Filtering with Articulation Constraints for Humanoid Robot Pose and Motion Estimation. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli (editors), *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 415–426. Springer Berlin / Heidelberg, 2012.

G. Jochmann, S. Kerner, S. Tasse, and O. Urbann. Efficient Multi-Hypotheses Unscented Kalman Filtering for Robust Localization. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli (editors), *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 222–233. Springer Berlin / Heidelberg, 2012.

O. Urbann, S. Kerner, and S. Tasse. Rigid and Soft Body Simulation Featuring Realistic Walk Behaviour. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli (editors), *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 126–136. Springer Berlin / Heidelberg, 2012.

S. Tasse, M. Hofmann, and O. Urbann. On Sensor Model Design Choices for Humanoid Robot Localization. In Xiaoping Chen, Peter Stone, LuisEnrique Sucar, and Tijn Zant (editors), *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 380–390. Springer Berlin Heidelberg, 2013.

S. Tasse, M. Hofmann, and O. Urbann. SLAM in the Dynamic Context of Robot Soccer Games. In Xiaoping Chen, Peter Stone, LuisEnrique Sucar, and Tijn Zant (editors), *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 368–379. Springer Berlin Heidelberg, 2013.

Abstract

Gathering information, especially about the immediately surrounding world, is a central aspect of any smart device, whether it is a robot, a partially autonomous vehicle, or a mobile handheld device. The consequential use of electrical sensors always implies the need to filter the imperfect sensor data output in order to gain reliable information. While the challenge of perception and cognition in machines is not a new one, new technology constantly opens up new possibilities and challenges. This is stressed further by the advent of cheap sensor technology and the possibility to use a multitude of small sensors, with the simultaneous constraint of limited resources on mobile, battery-powered computing devices.

In this work, stochastic methods are used to filter sensor data, which is gathered by mobile devices, to model the devices' location and eventually also relevant parts of their dynamic environment. This is done with a focus on online algorithms and computation on these mobile devices themselves, which implies limited available processing power and the necessity for computational efficiency. This dissertation's purpose is to impart a better understanding about the conception and design of stochastic filtering solutions, to propose localization algorithms beyond the current state of the art, and to show the use of simultaneous localization and mapping algorithms in the context of cooperatively estimating the surrounding world of a team of robots in a fast changing, dynamic environment. To achieve these goals, the concepts are depicted in multiple application scenarios, design choices and their implications systematically cover all aspects of sensing and estimation, and the proposed systems are evaluated in real-world experiments on humanoid robots and other mobile devices.

Contents

Chapter 1

Introduction

An idea about one's whereabouts and about the immediate surrounding environment is natural for every person. This comes instinctively and provides the means for moving in and interacting with the world. Such knowledge is also desirable for many technical devices to perform their function more effectively or to provide new functionality, either in automation, service, or in ubiquitous computing applications which support our everyday life.

Current technology provides various means of sensing specific properties in the world. Sifting through the vast amounts of sensor data and inferring general truths about the state of the world is, however, still a problem in which human brains are superior to even the most advanced machine. The challenge of perception and cognition in machines is the manipulation of the sensory input's mathematical representations to find conclusive insights based on noisy, redundant, and sometimes contradictory sensor readings.

In the course of this work, stochastic methods are applied to solve this challenge, which is considered as a problem of estimating physical properties of mobile devices and their surrounding environment based on sensor information. Special focus is on online solutions for this problem, e.g. to provide the best estimation at the time of sensing, in contrast to computing the optimal result based on a complete set of previously recorded sensor data. State of the art methods are reviewed and supplemented, and findings and improvements are illustrated in general terms as well as in several different concrete application scenarios.

1.1 Why Stochastic Modeling?

For some readers the use of stochastic theory might not be obvious when working with sensor data. After all, uncertainty in sensor readings is primarily undesirable. If sensor readings contain a large amount of uncertainty, an engineer will always try to use better, more accurate sensors to eliminate uncertainty and to directly gain a precise, detailed and complete model of the world.

However, as long as sensors are not perfect, their measurements are not perfect either. Just using these plain measurements without further knowledge of the sensor's properties will sometimes still give results within an acceptable margin of error. Using stochastic expressions to formulate the knowledge about how this specific sensor is "not perfect", though, can often improve the results. The following example illustrates this concept.

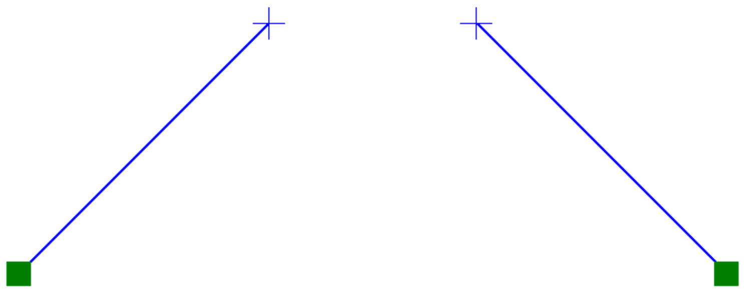
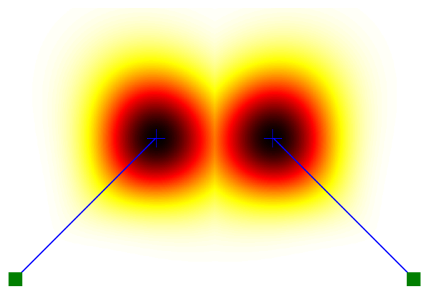
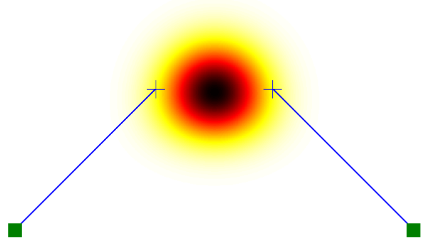


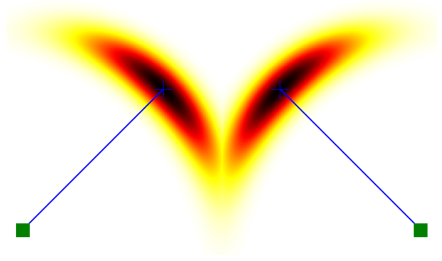
Figure 1.1: Raw sensor measurements (blue) of two sensors (green squares).

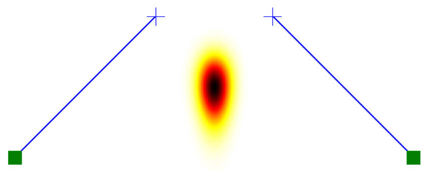
Figure 1.1 shows two sensors, depicted by green squares, which measure the angle and distance to an object. Assuming that both sensors measure the same object at the same time, the measurements obviously contain errors, since the positions as indicated by each sensor separately do not coincide. If no further information is given, the usual practice of combining both measurements is to calculate their average. But if something is known about the sensing process and its imperfections, this knowledge about the involved statistic uncertainties of the sensing process can be exploited to estimate the most likely origin of the measurement.

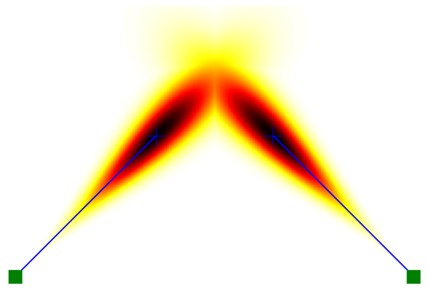
Figure 1.2 illustrates how different the result of a sensor fusion can look for different sensing processes. For normally distributed errors in the measurement domain of sensed angle and distance, figures 1.2a, 1.2c and 1.2e show the position likelihood of the sensed objects without further conditions. Figures 1.2b, 1.2d and 1.2f show the corresponding (normalized) likelihood under the assumption that the measurements of both sensors originate from the same object. For equal uncertainties in both angle and distance measurements as in figure 1.2a, the fusion of both measurements as shown in figure 1.2b corresponds to an average of the Cartesian coordinates of both measurements. If the measurement process is more precise in sensing the angle than the distance, or the other way around, then the object's most likely position based on these two measurements differs significantly, as demonstrated in figures 1.2d and 1.2f. This is relevant for many real sensors: Sonar sensors for example measure the distance more precisely than the angle to


(a) Separate measurement probabilities for sensors with equal uncertainties in angle and distance.


(b) Combined measurement probability for sensors with equal uncertainties in angle and distance.


(c) Separate measurement probabilities for sensors with uncertain angle observation.


(d) Combined measurement probability for sensors with uncertain angle observation.


(e) Separate measurement probabilities for sensors with uncertain distance observation.

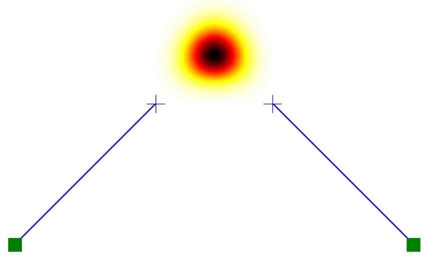

(f) Combined measurement probability for sensors with uncertain distance observation.

Figure 1.2: Different uncertainties of a sensing process and their impact on the sensor fusion result.

the reflecting object. When recognizing an object by means of image processing on the other hand, the originating angle with respect to the camera's optical axis can be calculated very accurately, while the distance is either unknown or can only be estimated roughly, e.g. by comparing the objects real and measured size.

This utilization of the information contained in a measurement's uncertainty is not only important when fusing multiple measurements made by different sensors of one object. The same applies for the integration of sensor data gathered over time. Therefore tasks necessitating the processing of vast amounts of sensor data often involve stochastic filtering to achieve reliable results even if the involved sensors are far from perfect.

1.2 Goals

In this work, stochastic methods are used to filter sensor data, which is gathered by mobile devices, to model the devices' location and eventually also relevant parts of their dynamic environment. This is done with a focus on online algorithms and computation on these mobile devices themselves, which implies limited available processing power and the necessity for computational efficiency. This dissertation's purpose is structured along the following three goals.

The first goal is to impart a better understanding about the conception and design of stochastic filtering solutions. At least in the robotics domain, stochastic filters are well known, standard literature like [86, 78] covers the common algorithms, and straight forward applications often bear acceptable, but not optimal results. A number of design choices are always part of an implementation process, some of which are discussed frequently in literature, while others are generally neglected or only mentioned briefly. As a consequence, these implementations rarely function to their full potential, and sometimes might even be dismissed prematurely for unsatisfactory estimation outputs, while small, seemingly negligible changes would completely change and significantly improve the system's behavior. This work covers several different design choices with respect to a filter's state space and update processes, and explains their influence on a filter's computational efficiency and its estimation quality. This implies an importance of some design aspects over others, which in some cases is contrary to the coverage in common literature. By highlighting these findings and their implications and providing general concepts and guidelines for the modeling of proprioceptive and exteroceptive processes, this work aids future engineers and researchers in the design of appropriate estimation systems and the optimal handling of sparse and noisy sensor data.

The proposal of localization algorithms beyond the current state of the art is this work's second goal. In the application scenario of smartphone localization for location-based services, a probabilistic robotics formulation of the problem is novel and challenging, insofar that locomotion information about intention as well as execution lies within the user and are thus, unlike in other robotic applications, inaccessible to an estimation algorithm by any direct means. This is overcome by exploiting knowledge about biped walking motion generation to infer the underlying walking motion from acceleration measurements. Consequently, suitable motion and sensor models and a localization system are implemented to prove the superiority of this concept over conventional indoor localization techniques for smartphones. For the application scenario of humanoid soccer robots, where Kalman and particle filters are already common, a novel approach to Gaussian mixture filtering is presented, which utilizes techniques from particle filtering in a multi-hypotheses Kalman system to incorporate valuable aspects of both filter strategies. The resulting solution is superior to common Kalman filters in its ability of fast re-localization and its representation of multi-modal belief distributions, and outperforms state of the art particle filters in localization accuracy and computational efficiency.

The task of simultaneous localization and mapping (SLAM), i.e. localization in a previously unmapped, typically static environment, is a very active field of research. It is this work's third goal to show the benefit of applying SLAM concepts also to complex dynamic environments, and to cooperatively estimate the localization of a team of robots as well as to track other dynamic objects and agents in a unified estimation state. The main challenge consists of managing the corresponding algorithms' complexity on embedded robotic platforms with very limited computational capability. This is achieved by an appropriate handling of the different information sources' heterogeneity and a careful application of a number of approximations, which lower the solution's complexity and at the same time simplify the distributed modeling among cooperating robots.

In summary, this dissertation aims to provide a solid stochastic and algorithmic basis for the design of estimation systems as well as several specific novel solutions. To achieve this, the concepts are depicted in multiple application scenarios, design choices and their implications systematically cover all aspects of sensing and estimation, and the proposed systems are evaluated in real-world experiments on mobile devices and robots.

1.3 Structure

In the course of this work, various estimation problems are covered in the context of three different applications: humanoid robotics with robot soccer as a benchmark task, mobile measurement units on intralogistic conveyor belt systems, and smartphone localization for location-based services. These three scenarios are briefly summarized in chapter 2 as a motivation for the introduced methods and concepts. Subsequent findings and conclusions prove applicable to many fields involving mobile devices in general, among these robotics applications. As such, the benefit is shown of depicting such application fields in a robotics context and modeling them as stochastic estimation problems, even if robotic aspects such as autonomous decision making and actuation are not present in all of them. This first part is concluded in chapter 3 with the introduction of several common algorithms from the field of probabilistic robotics, especially the Bayes filter concept and its most common implementations, the Kalman filter and the particle filter. This chapter represents the general part of this thesis' related work. More specific related solutions and publications are briefly introduced in each following chapter to provide the context for each topic.

Throughout the rest of the dissertation from chapter 4 to chapter 8, own work is presented exclusively. On the top level, this work is structured by covering all relevant aspects of stochastic modeling with increasing complexity of the presented scenarios. Chapter 4 covers the proprioception of robots, i.e. all inward perceptions like those resulting in the knowledge of one's own body configuration, relative positioning of body parts, and the dynamics thereof. Chapter 5 is complementary to this by covering exteroceptive aspects, i.e. outward oriented perception. Concepts from both previous chapters are applied in chapter 6, where a probabilistic smartphone localization solution is derived for indoor scenarios, for which no similar stochastic approaches exist yet. Chapter 7 then develops a localization solution for a robotics context with the aim of surpassing existing state of the art algorithms. Finally, in chapter 8, the stochastic modeling task is extended from localization of single mobile devices in static environments to multiple communicating agents in a dynamic environment. The thesis is concluded in chapter 9.

Chapter 2

Application Scenarios

In the course of this dissertation, the presented concepts are illustrated using several application scenarios. In general, those concepts are depicted in robotic contexts, even if robotic aspects — such as autonomous decision making and actuation — are not present in two out of three cases, or merely in the very limited form of user interaction in case of location-based ubiquitous computing. However, it will be shown how tasks in the following three areas, namely humanoid robots, intralogistic systems, and ubiquitous computing for location-based services, benefit from a probabilistic robotics viewpoint.

2.1 Humanoid Robots

The research field of autonomous mobile robots provides the main application scenario for this work. Previously, the majority of robots were build for industrial applications, production or transporting material, in clearly defined environments such as assembly line factories. Common robotic arms are mostly preprogrammed, statically fixed to the floor, and only able to adapt in a very limited way to changes in their designated working area. This contrasts with the mobile and intelligent robot known from science fiction books and movies, which often appears in humanoid form and possesses strength, dexterity, and cognitive abilities beyond those of humans.

Mobile robots are the focus of current research and development with the trend towards smaller, lighter, faster and more autonomous systems. In recent years, some first models already found their way into the consumer market. One popular example are autonomous robotic vacuum cleaners in private households. More complicated systems already exist for office environments or military applications [79]. Those however are only used as telepresence systems, as precise and reliable autonomous understanding of complex environments and situations in general is still beyond the capabilities of current robots and an active research problem.

The ultimate challenge of autonomous mobile robots is the development of robots with similar (or even surpassing) physical and mental capabilities as humans. There are two main motivations behind building humanoid robots, i.e. robots with human physique: On the one hand, as indoor or office environments are made for humans, robots with humanoid form and the necessary control over their body would obviously be able to navigate in those environments as well. This also includes for example stairs, which present insurmountable obstacles for rolling robots. On the other hand, robots have a higher acceptance by people the more human they look, at least up to a certain point of similarity.

In the following, the RoboCup Standard Platform League will be described as the context of humanoid robotics research at the Robotics Research Institute of TU Dortmund University, and to describe the scenarios in which algorithms are evaluated in later chapters.

2.1.1 The RoboCup Standard Platform League

Several robotic contests have been established over the last years to foster research and development. Among those are the following: The European Land Robot Trials (*ELROB*) focus on short-term realizable robot systems for reconnaissance, surveillance and security tasks. The first two DARPA Challenges, namely the *DARPA Grand Challenge* and *Urban Challenge* were aiming at autonomous vehicles for off-road and urban driving, respectively. The *DARPA Robotics Challenge* instead focuses on humanoid robots for disaster relief and rescue operations. A strictly civilian contest concentrating equally on fundamental research and applications of autonomous robots is the *Robot World Cup Initiative*, in short *RoboCup*.

RoboCup aims at forwarding research into artificial intelligence and robotics by setting a plain and generally understandable task. Choosing soccer as a challenge for artificial intelligence in contrast to logical problems like chess necessitates the integration of various technologies and sub tasks, such as multi agent cooperation in dynamic environments, processing sensor data and choosing and executing actions, all together in systems under real-time constraints. The competitive character additionally results in continuous adaptation and sophistication of the teams playing against each other. Additional tracks for search and rescue as well as service and industrial applications are meant to transfer basic research results from soccer robotics into real-world application scenarios.

The RoboCup consists of different leagues focusing on different aspects of the overall problems, ranging from tactics and strategy in the simulation leagues to basic skills and hardware development in hardware leagues. In the *Standard Platform League* (SPL) the humanoid robots of type Nao by Aldebaran Robotics are used. As all teams work with the same robot models, the focus is on developing software which utilizes the given hardware in the most optimal way possible. In contrast to other leagues where each team has different robots, published code by high ranked teams can easily be incorporated into other teams' code. This custom of sharing research results — not only by way of publication but also in form of code releases — applies further pressure towards improvement and introducing and integrating new ideas.

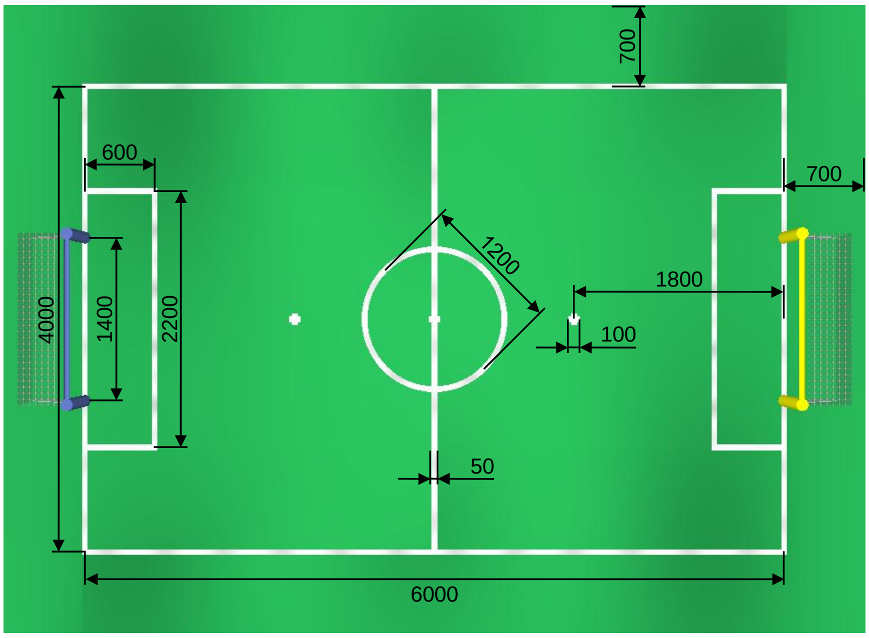


Figure 2.1: Dimensions of an SPL field given in millimeter.

Figures 2.1 and 2.2 illustrate the scenario of RoboCup Standard Platform League soccer games. While the fields are specified as given in figure 2.1 and known to the robots, the surrounding is generally unknown and may change during the game. As can be seen in figure 2.2, there is no separation between field and audience. As a consequence, many false positive observations arise and influence the robot's perception and model of the world. It is of importance to note that those observations are not just occurring randomly in the environment, but can also be generated systematically from specific points, e.g. a person in blue jeans standing at the border of the field and blue goal posts may be indistinguishable from a robot's perspective and field of view. The consequences of this and corresponding solutions will be covered in detail in chapter 7.

2.1.2 Nao: A Commercial Humanoid Robot Platform

This section gives a short overview of the Nao robot, as it is used for the evaluation of various algorithms in the course of this work. Especially chap-

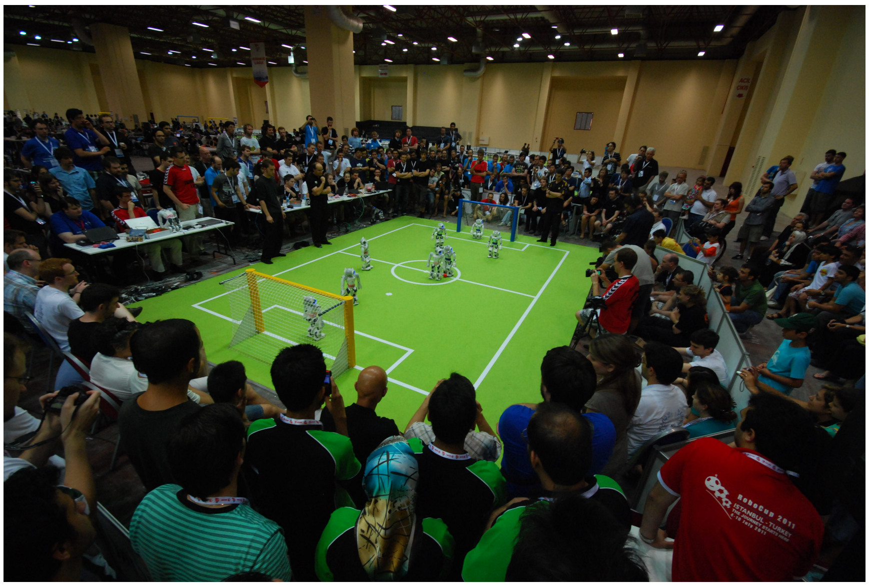


Figure 2.2: SPL game during RoboCup 2011, without any separation between field and audience.

ters 4 and 5 refer to the Nao's characteristics directly, but overall this also has an impact on design decisions throughout this work.

The Nao is a humanoid robot platform developed by Aldebaran Robotics and intended for teaching, research and entertainment. It replaced Sony's 4-legged robot Aibo as the standard robot in the SPL in 2008[1]. The Nao model repeatedly received upgrades and currently different versions are available. While some algorithms presented in this work have been applied to various Nao versions, all evaluations as well as the following description refer to the NaoV3 RoboCup Edition model which has been used in RoboCup 2010, and in RoboCup 2011 with only minor modifications[2].

This Nao model contains an x86 AMD Geode 500MHz CPU and 256MB SDRAM. Its main source of information about the surrounding world are two cameras in its head, which do not have an overlapping field of view and can only be used one at a time. The Nao also contains sonar sensors in its chest with an effective cone of 60°and a detection range from 0.25 m up to 2.55 m. However, due to the wide detection cone, these sensors will detect the ground at around 0.7 m distance when the robot is standing straight, which therefore limits the effective range for obstacle detection. This ground detection distance also varies depending on the exact torso orientation and height. Additionally, the arm movement needs to be confined to stay out of the sonar detection cone. Due to those restrictions and the high noise ratio

[1]Aibo and Nao competitions have been held in parallel at RoboCup 2008

[2]Later usage of the term *Nao* refers to the NaoV3 RoboCup Edition.

the sonar sensors are only used to verify the distance of visually detected close obstacles.

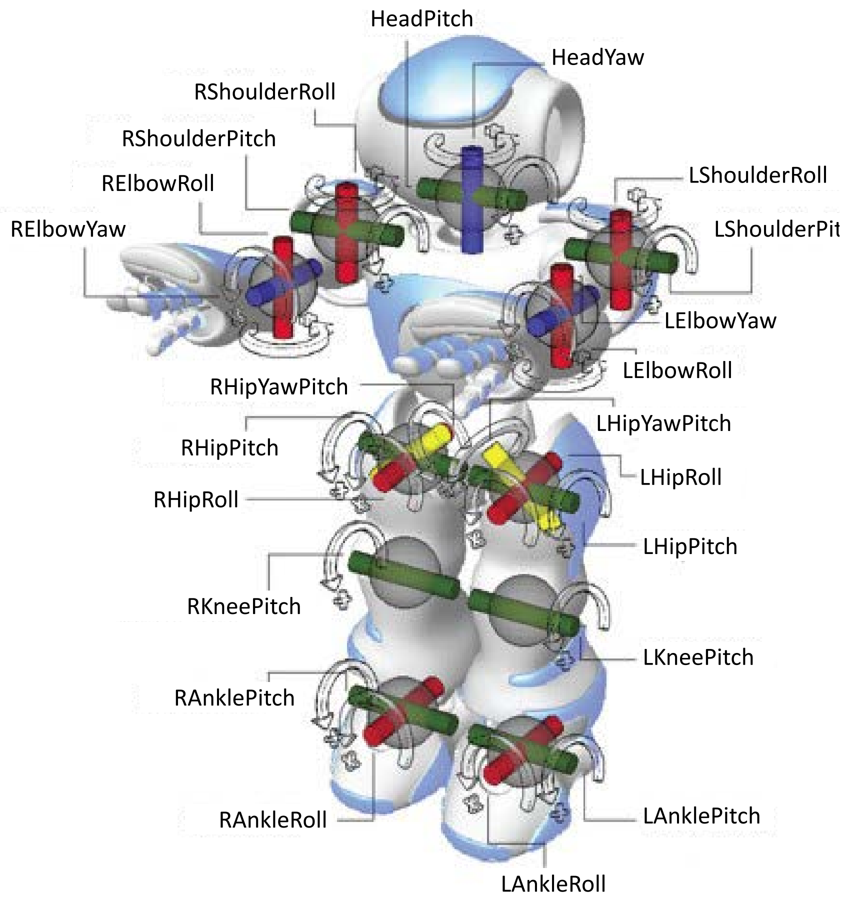


Figure 2.3: The degrees of freedom of the Nao RoboCup Edition (copyright by Aldebaran Robotics).

Nao's 21 degrees of freedom are visualized in figure 2.3. It has to be noted that the left and right *HipYawPitch* joints (shown in yellow in figure 2.3) are physically coupled and controlled by a single motor. This allows the narrow waist and graceful body, but at the same time complicates the robot's kinematics. Consequently, both feet cannot achieve arbitrary positions and rotations simultaneously, but the workspaces of both feet depend on each other. This further complicates precise movement execution and odometry measurements, which will be addressed in chapter 4.

2.1.3 Robotic Software Framework

The algorithms proposed in later chapters are implemented as modules in a robot software framework used to control the Nao. This framework is based on the German Team framework [73, 72], which has been adapted to the Nao [8, 15, 16].

The middleware NaoQi is provided by the manufacturer Aldebaran Robotics as a way to control the Nao using high level commands, but also as the sole interface to the Nao's sensors and actuators by means of the Device Communication Manager (DCM). Part of the software framework runs as a module in NaoQi and exchanges sensor data and actuator commands via shared memory with the actual robot control program. The cameras are accessed using standard video for Linux drivers. The control program itself consists of three processes, as visualized in figure 2.4. The motion process reads sensor data and generates actuator outputs at 100 Hz, and is responsible for the generation and execution of stable walking motions. The cognition process contains the analysis of information about the external world, e.g. by means of image processing and stochastic modeling, and the behavior decision making [58, 12]. A third process allows the exchange of debug information with external devices, which is not used during competition games.

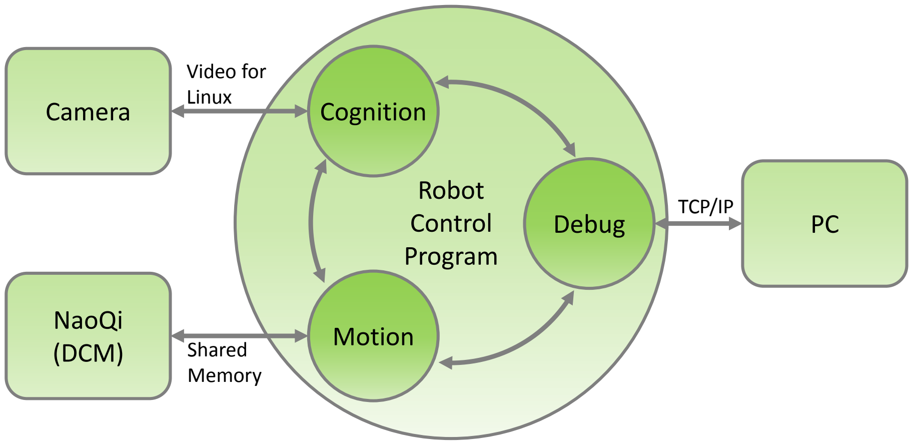


Figure 2.4: The software framework used on the Nao.

Inside the processes like motion and cognition the information flow is organized in modules. Each module requires and provides certain representations. Such representations are data structures containing for example the images, raw sensor data, various image processing or filtering results, motion control commands, and finally actuator requests. Using a debug connection, it is possible to switch between different module configurations at runtime and to visualize or even modify the representations.

Part of the framework is the simulation software SimRobot [56]. It is based on a solid-state physics simulation with extensions for realistic image generation, including rolling shutter effects, motion blur, etc., and is equipped with Nao models. The robots are simulated according to the specifications of the Nao's physical properties as provided by Aldebaran Robotics. The simulation allows testing and debugging of parts of the robot control software, or even the interaction of a whole team of robots in a controlled environment. This reduces hardware wear and shortens development cycles. In a limited way, it also allows experiments with algorithms, which do not meet the runtime constraints of the Nao's limited processing power.

2.2 Intralogistic Systems

[3] Intralogistics refers to the internal flow of material and information at a given site, e.g. a warehouse, factory, or distribution center. The materials or products are commonly handled using conveyor systems (see figure 2.5 for one example of a roller conveyor system). Intralogistic systems are generally modular so that they can be assembled to the specific need and requirements at hand. However, this flexibility usually ends with the construction of the system, which thereafter operates in a static configuration with little or no redundancy due to reasons of cost efficiency. A failure of one transportation segment can thus often lead to a down-time of a large portion of the site, resulting in significant economic loss.

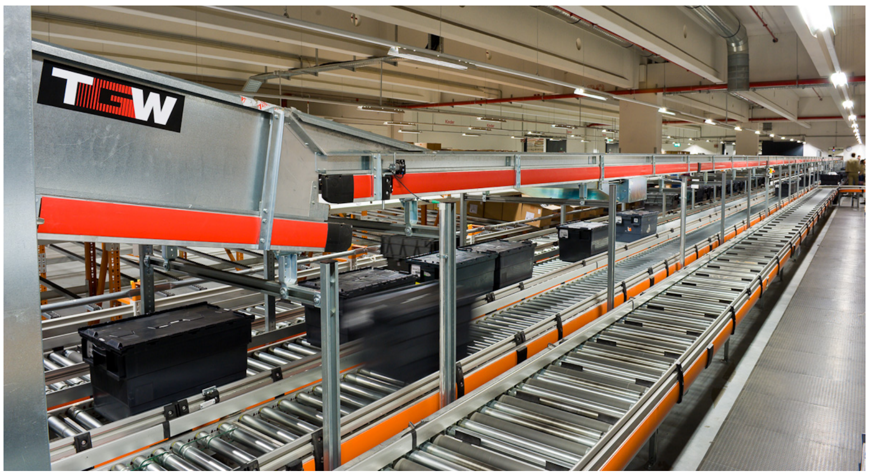


Figure 2.5: Accumulation roller conveyor for the transport of cartons and totes in distribution centers (by Kay-Uwe Rosseburg, available under a Creative Commons Attribution Licence 3.0 at `http://commons.wikimedia.org/wiki/File:Accumulation_Roller_Conveyor_for_Cartons_Totes.jpg`).

Such unplanned down-times can be avoided by anticipating failures and scheduling maintenance at the least critical times. For state-of-the-art in-

[3]Parts of this section are based on an article in the Proceedings of the IEEE International Conference on Automation and Logistics 2010. The corresponding publication, see [53], is available at `http://ieeexplore.ieee.org`.

tralogistic systems with high load and usage rates it is therefore important not just to monitor the handling and transportation processes themselves, but the condition of the entire system as well. As requirements with regard to reliability and availability gain in importance, demands arise concerning the application of modern maintenance concepts.

2.2.1 Condition Monitoring for Logistics on Demand

The following application scenario originated from within the scope of the collaborative research centre 696 "Logistics on Demand" which was established at the TU Dortmund University in July 2006 and financially supported by the German Research Foundation (DFG). Its goal is to allow a more efficient design and organization of intralogistic systems by developing analytical methods and technologies based on actual requirements. This will help rather inflexible current intralogistic systems, with regard to ineffectiveness in purchasing, operating and maintenance, to transition into systems able to meet the fast changing demands like "just in time" and "just in sequence" production concepts, and utilize the exponential increase of data and availability of new delivering methods.

"Logistics on Demand" strives to achieve this by adapting both operation and maintenance of intralogistic systems to current and anticipated future demand. Transportation systems can adjust by allowing certain parameters, which currently are fixed when installing a system based on the maximum expected load, to vary depending on actual load and usage statistics [23]. This can reduce the wear especially on failure relevant mechatronic components, and potentially also decrease the energy consumption of the material handling systems. To anticipate the maintenance demand it is necessary to monitor not only the transported products but also the transportation system itself in detail and in a cost-efficient, preferably automated way. As complex intralogistic systems are characterized by long conveying distances and huge dimensions, it is not economical to equip the whole system with stationary sensors in the necessary density. [24], [54] and [55] discuss some components with strong influence on the overall condition of an intralogistic system and also explain the dedicated techniques which allow to monitor these components. Some examples are rollers, electric drives or friction belts. New maintenance concepts based on the system's condition and requirements are therefore supposed to utilize both stationary and mobile measurement units. The latter travel on the conveying system without the need of an own locomotion mechanism and can collect measurements of huge parts of the transportation system with minimal effort for the system's operator.

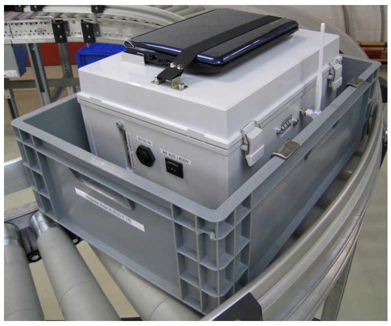
The data gathered by the mobile measurement units has to be correlated with reliable location information to be useful. Track and trace applications in the field of logistics commonly apply light barriers, bar-codes, RFID-

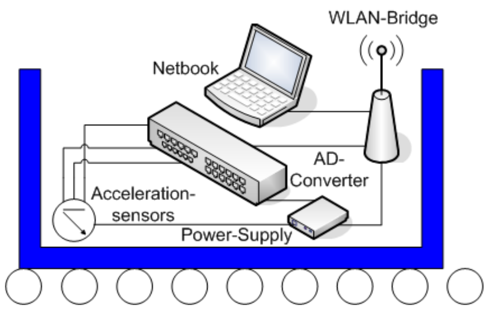
Chips [48] or GPS-Receivers. These techniques, however, are better suited for logistic processes between plants or the shipping of goods, rather than to the intralogistic material handling and the gathering of precise location information. The latter is required for identifying specific parts and potential failure areas of the handling system for maintenance purposes.

Of the broader scope of the "Logistics on Demand" project, this work only addresses the task of localizing mobile measurement units and mapping potential failure areas as an application oriented scenario for the use of stochastic modeling. This includes and extends the work presented in [53].

2.2.2 Mobile Sensor Unit

A prototypical setup is shown in figure 2.6a. A regular load container with a base area of 600 mm by 400 mm is used as a chassis for transportation on the conveyor. It is equipped with a 3-dimensional acceleration sensor in order to measure in horizontal, vertical and cross direction. The measured accelerations are the result of the forces acting upon the container. These include a regular part consisting of gravity, accelerations due to controlled changes of vertical and horizontal speed, and centrifugal accelerations in curving segments. Additional accelerations may occur due to unplanned or undesired behavior of the container, such as speed changes or bumps. Those can be caused by slippage, or by mechanical wear or mis-alignment between different segments or parts. The identification of these latter ones is an indication for potential failure areas.


(a) Mobile unit in container.



(b) Concept of the measurement unit.

Figure 2.6: Mobile measurement unit [53].

The mobile unit participates in the transportation process between the regular material containers on the conveying system. While traveling, it permanently collects data and evaluates it in batches as presented in [53], so that reporting is available in regular intervals. Section 4.1 also presents efficient online algorithms performing the same tasks, so that the evaluation can be done in real-time. To be independent from unreliable Wi-Fi signals and external devices, a regular netbook is attached to the equipment so that the sensor readings can be gathered directly (cp. figure 2.6b). The data is processed and after it is evaluated only the results are transmitted to a receiver once a strong Wi-Fi signal is present. The utilization of the mobile measurement unit is clarified by the example of a roller conveyor which is a central element of most intralogistic systems. Figure 2.7 shows the experimental setup which is used for evaluation purposes.

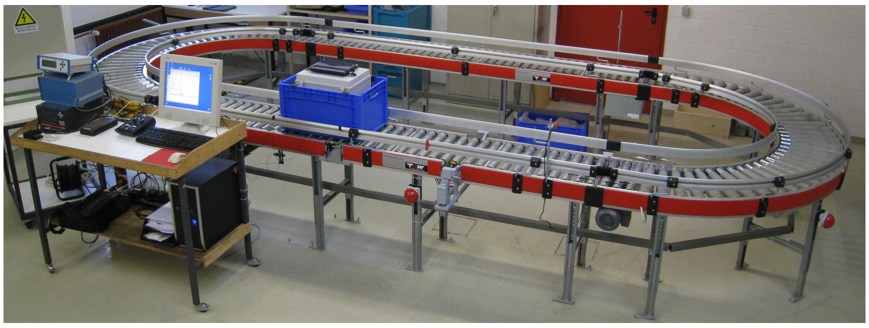


Figure 2.7: Roller conveyor in an experimental setup [53].

The described layout of the mobile measurement unit is the one used for the experiments in [53] and for the illustration and evaluation of the algorithms presented in section 4.1. It should be noted though, that the same functionality can be provided by much more compact devices such as regular smartphones. In fact, most recent smartphones contain gyroscopes as well as acceleration sensors and therefore superior sensory equipment as well as equal or superior computational capabilities compared to the mobile measurement unit used here. Therefore the intralogistic system condition monitoring described here can be implemented exclusively using affordable components, which signifies its practical relevance.

2.3 Location-based Services

Over the last years, the demand for precise localization has extended beyond robotics research and military applications. Modern life is permeated by ubiquitous computing and location-based services. Nowadays, navigation systems in cars are common and reliably pick the fastest routes to desired destinations based on current traffic statistics. The introduction of smartphones with localization capabilities gives rise to a multitude of new applications. Some of those are just known applications in new contexts, like routing systems for pedestrians using public transport, but many new ones are appearing out of the omnipresence of these devices and their vast developer community. Popular examples are location-based games, which integrate the player's location into the context of the game, and travel applications, which augment the smartphone's live video stream with labels for local sights, places of interest, and nearby stores and restaurants.

The widespread availability of satellite-based navigation makes positioning in open areas possible with high accuracy and precision. This positional accuracy deteriorates in urban environments due to signal reflections off the concrete or even loss of part of the satellites' signals. To compensate this loss of accuracy, some techniques are already implemented in most smartphones, using cell tower and Wi-Fi signal strength for example. But existing methods still do not guarantee the spacial resolution desired by many providers of location-based services. In this context, the lack of positioning information inside buildings is even more prominent.

2.3.1 Indoor Smartphone Positioning

Indoor localization presents a harder problem than the outdoor case due to two reasons: To begin with, signal strength from outdoor sources such as satellites or cell towers are either blocked completely by the building's steel and concrete structure, or can be received only with interference and after several reflections off walls and ceilings, which significantly compromises methods relying on measured signal strength or a signal's time of travel. But even if a similar accuracy as outdoors could be achieved indoors, this would often not be enough. The same position uncertainty, e.g. a precision of around 15 m, still allows to reliably tell the correct street corner and is thus sufficient for most outdoors routing needs, but covers several different hallways, rooms, and even floors inside a building, hence rendering an indoor service unable to provide any information about the specific room or even the user's position inside the room.

Assuming indoor positioning mechanisms with an accuracy as low as a single meter, several interesting applications would become possible, ranging from museum guides, routing systems for large airports, to a multitude of commercial uses. The availability of such mechanisms without the need for additional hardware beyond common smartphone technology is especially appealing, as many people already invested in smartphones voluntarily, and new software can be easily distributed commercially using the existing build-in stores.

Location-based services for retail trade will be illustrated as an example. In this context, several stores already provide smartphone applications for the management of shopping lists. These are stored online and can be updated remotely from a home computer while another person is already in the store. A superior shopping experience is provided to the customers by coupling this shopping list with a functionality for scanning and adding articles, and finally bypassing queues at counters by electronically paying the already registered products. Such applications already exist, as does the desire to extend them with exact positioning information inside stores.

Knowing the precise location and context a customer is acting within at any given time in the shopping center allows for a number of use case scenarios:

- *Context-aware Offerings* summarize the ability of actively notifying the client application on a customer's smartphone when approaching a specific location or acting in a certain context. By combining the location with information about articles in the cart and on the shopping list, fitting offerings can be made based on customer profiles or even possible recipes, e.g. offering a certain white wine near the liquor section to go with the already purchased fish. This is similar to offerings in online shops, in which these are already widely accepted and appreciated.
- *Advertisement Impact Measurement* allows to directly quantify the impact of placing display stands or giveaway booths for certain brands at certain places, whereas currently this can only be evaluated using the increase in sales for this specific article.
- *Audience-specific Routing*, based on the costumer's shopping list and his profile, could lead him through the store in a way, which may not be the shortest one possible, but still more time-efficient than the customer's own wandering, while at the same time passing all offerings which the store company's profile deems relevant.

To enable such indoor location-based services, the localization methods needs to provide high-resolution indoor position information, which are up-to-date and continuously verified. Furthermore, the system needs to be scalable, i.e. it must be possible to roll out to a potentially large customer basis, in a cost-effective way. This implies that no additional hardware must be needed on the customers' side, and the effort of calibration and customization of new stores needs to be kept at a minimum. These requirements and restrictions will be addressed later for the development of a suitable localization system.

2.3.2 Mainstream Technologies for Localization

This section reviews different means of localization and their performance, starting from well-established technologies. Modern smartphones already provide means for localizing themselves; a capability that is leveraged by many location-based services applications, though mostly for outdoor use so far.

2.3.2.1 Satellite-based Navigation

The localization technology used by most of these devices and services is satellite-based, mostly on the Global Positioning System (GPS) which has been developed since 1973 by the United States of America and came fully operational in 1994 [64], available from the start for civilian use with selective availability and lower resolution, and since 2000 without restrictions [61]. Similar systems have been developed by other nations [36], namely the Russian GLObal NAvigation Satellite System (GLONASS), which has been opened to civilian use in 2007, or are currently under development, such as the European Galileo positioning system or Indian and Chinese versions thereof.

In outside scenarios, the Global Positioning System (GPS) is the means of choice for most applications. For faster initialization, the position fix is often seeded with rough position estimates from surrounding cell tower signals. In open areas with direct satellite visibility, the accuracy of GPS can be in the magnitude of few meters [93]. In urban areas, however, signal reflections, the multi-path problem[4], and also the loss of weaker satellite signals usually impairs the quality drastically [67].

2.3.2.2 Wi-Fi-based Localization

Another ubiquitous mechanism that receives more and more attention is the localization through Wi-Fi access points in the proximity of the device: With the focus on urban environments, they reach a very high density, thus providing a small-tiled mesh of potential signal sources. Given an up-to-date database, Wi-Fi localization has a good potential for higher resolutions than GPS.

Generally, two different methods can be distinguished when inferring the position from radio signal strength [59, 3, 80, 33]: the *propagation-based* approach and *fingerprinting*. Propagation-based approaches use radio propagation models, either empirical or theoretical ones, to calculate distances from the received transmission energy. A multilateration algorithm can then use the resulting distances to different sources for calculating the most likely receiver position. Similar to GPS, which itself utilizes a time difference of arrival multilateration, such methods provide good results as long as the path between sender and receiver is reasonably free. In indoor environments, however, a direct line of sight is rarely the case. Instead the signal is affected by constructive or destructive interference, obstructed, partially absorbed, or reflected, which results in significant deviations from the rather abstract mathematical propagation models, and eventually leads to decreased reliability and precision of the position estimate.

Instead of using mathematical models for the strength of signals, fingerprinting methods rely on building a comprehensive database of sample measurements at different known locations consisting of all received signals and their strengths. A new measurement at an unknown location

[4]That is, several possible paths are close enough to each other that a deterioration of accuracy makes them indistinguishable from each other.

is then compared to the ones in the database and the nearest neighbors in signal-space are used to infer the current location. Various techniques have been proposed, ranging from simply picking the single nearest neighbor [19] to different interpolation methods among different numbers of nearest neighbors [59, 74]. Many methods only consider the similarity in signal space [19, 59], some also take the spacial distribution of the database samples into account [74], e.g. by finding the smallest polygon out of a given number of nearest neighbors and using its centroid as the localization result. Averaging the signal strength over time and over signals known to originate from the same access point naturally improves the reliability, but an accuracy of around 2m can only be achieved in ideal situations of strong and stable signals from three or more access points, otherwise dropping to merely room precision or worse [59].

Overall, it shows that the required accuracy cannot be achieved with traditional approaches only, necessitating to prospect for other options. To cope with this issue, chapter 6 formulates the problem in a more general way using stochastic filtering mechanisms.

2.4 Common Task Formulation

The three described application scenarios, while vastly different in purpose and utilized hardware, nevertheless all require solutions to a common task, namely the tracking and localization of mobile devices. Furthermore, this information needs to be available at the time of operation, not later after extensive processing. So the required algorithms must provide the best possible result at each time based on all data accessible so far, and must be efficient enough to run on-line, i.e. to process each new data instance in the time till the next one is available.

As an input for all localization systems serves a variety of sensors, which differ in hardware implementation, measured physical property and purpose. In the following chapters, these will be classified into sensors measuring properties of the mobile device's own kinematic and dynamic state and into ones measuring properties of the device's surrounding, external environment. The former ones are called proprioceptive sensors, such as inertia measurement units or joint angle sensors. The latter ones are exteroceptive sensors, such as cameras or Wi-Fi receivers.

The scope and complexity of the task differs for each application scenario. Localization, i.e. finding the device's own position, is the major challenge in the location-based service and intralogistic scenarios. The latter one only briefly covers aspects of mapping static environment characteristics. The robot soccer scenario, on the other hand, also requires the tracking of a number of dynamic elements in each robot's environment additionally to calculating its own location therein. It is possible, though, to formulate these tasks using the same framework of a stochastic estimation problem. So understanding the applied filtering algorithms and the proprioceptive and exteroceptive aspects of the different scenarios allows to transfer findings from one to another, and thus also to successfully apply them to other possible engineering problems.

Chapter 3

Filtering Algorithms

This chapter covers basic algorithmic solutions to hard problems in the robotics domain based on probabilistic methods. As such, it does not present own work, but rather gives an overview about concepts and algorithms which will be applied and extended in later chapters. The terminology in this one and in later chapters, as well as the overview provided here, will follow [86] by Thrun, Burgard, and Fox.

3.1 Basic Concepts

The inference of knowledge about the world based on sensor information is the challenge, which will be addressed for all three application scenarios presented in chapter 2. A complete description of the world is of course infinite, but only a subset of it is relevant for a specific task. This part of the world or the environment is characterized by its *state*, which contains information about all its relevant aspects. For this state x of relevant physical properties to be susceptible of stochastic estimation, it is assumed to be described at each time t by a vector of real numbers:

$$x_t \in \mathbb{R}^n. \tag{3.1}$$

If only the agent's position is of interest, this is called a *localization* problem. The location in a 2-dimensional environment is then typically described by the robot's *pose*: a vector $(p_x, p_y, p_\theta)^T$ consisting of coordinates p_x and p_y along the two spacial axes and an orientation angle p_θ. In this case, the information which is used to localize in an environment is usually given in form of a map m.

The generalized case is called *world modeling* and typically also involves the positions and states of other static and dynamic objects in the environment. When addressed separately, those elements of the map will be denoted m_i; among other real valued properties, such a model of an object may contain spacial coordinates $(m_{i,x}, m_{i,y})^T$ or velocity components

$(m_{i,v_x}, m_{i,v_y})^T$. Henceforth, the terms *world* and *environment* will be used synonymously to address those aspects which are important for the task at hand. In this context, modeling the world is therefore equivalent to inferring its state as accurately and precisely as possible.

The approaches based on these concepts are applicable to very general scenarios. For the illustrations and examples in this chapter though, a robotics context as described in section 2.1 will be used.

Learning about the world is possible through information sources in form of electronic sensors, which unfortunately do not observe the state directly, but measure different quantities in the environment. Additionally, the environment may be influenced by own actions. Even if their intent is known, their outcome is rarely deterministic. Together with a-priori knowledge about relations and causalities of objects and processes, those are the available information sources for the estimation of the environment's state.

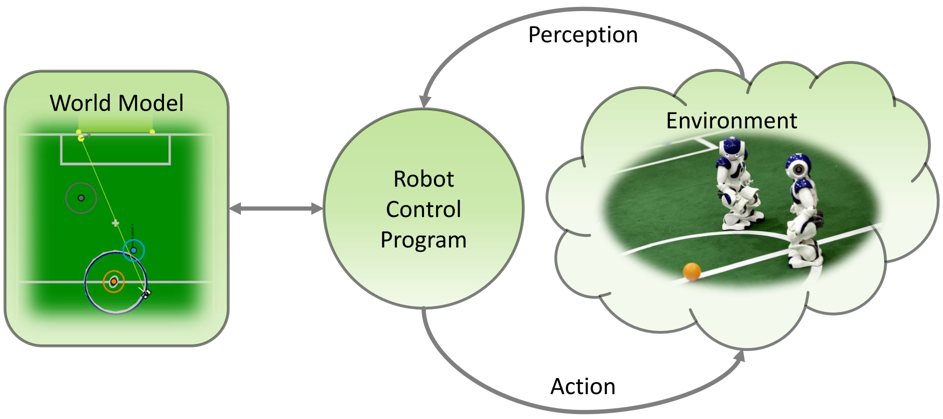


Figure 3.1: Robot environment interaction.

The relations between robot and world are visualized in figure 3.1. Commonly, robot software is designed based on this control loop: The robot observes its environment, integrates this new information into its internal model of the world, updates behavior decisions based on this internal model, and executes the corresponding actions. Executing actions mostly concerns motion execution, while perception summarizes various means of sensor or image processing. Both are active fields of research, and certain aspects will be addressed in chapters 4 and 5, respectively. The main concern of this work, however, focuses on world modeling, so other aspects are only covered insofar as they are relevant for the design of world modeling systems.

After its initialization, the world model is frequently updated using two kinds of information: those about the robot's actions and those about its perception. The former is called *process update* or *motion update*, and the latter is called *sensor update* or *measurement update*. Perception information comes in the form of new sensor readings or image processing results.

Such measurement data at time t will be denoted z_t, or $z_{t_1:t_2}$ for the set of all measurements taken between time t_1 and t_2 for $t_1 \leq t_2$. In the following, each measurement is a vector of k real numbers:

$$z_t \in \mathbb{R}^k. \tag{3.2}$$

Information about the robot's actions or change of state, in general called *control data* and denoted u_t, is also given in form of a vector as in equation 3.3, but can originate from different sources.

$$u_t \in \mathbb{R}^q \tag{3.3}$$

If a robot's locomotion system is controlled using speed vector inputs, for example, this speed control data can be used directly. Such motion commands, however, are rarely executed perfectly, but their effect is sometimes easy to measure. For wheeled robots an obvious possibility is the use of wheel encoders to measure the executed rotation instead of using the motor commands directly. Such information concerning the change of state is called *odometry* measurement and commonly used equivalently to control data [86, 78].

In general, odometry "refers to the use of data from the actuators (wheels, treads, etc.) to estimate the overall motion of the vehicle" [22, page 477]. The exclusive use of odometry to estimate a robot's pose is called *dead reckoning*, which corresponds to only executing the above mentioned motion update. An error between the dead reckoning estimate and the true path of the robot is unavoidable due to various causes, such as variations between commanded and executed motor movements, inaccurate or erroneous modeling of the robot's own properties and those of its environment, and resulting effects like slippage which are difficult to measure. *Position tracking* therefore incorporates additional sensor information to correct those errors and to improve this estimate of the robot's path. While position tracking implies a known position at one point in time, which is tracked afterwards, the problem of finding one's position without any knowledge about initial positioning is called *global localization.*

Those different modes of the localization problem frequently occur in common realistic application scenarios. A localization problem may start with global localization, and proceeds as position tracking or *local localization* once a position hypothesis is sufficiently validated. In case the tracking fails, re-localization needs to be performed similar to initial global localization. The *kidnapped robot problem* describes the worst-case scenario in which a robot, which is tracking its position, is "teleported" without its knowledge, i.e. instantaneously placed at another position, and subsequently needs to register that its previous position estimate has become invalid, and then to re-localize. Note that this is actually more difficult than the global localization problem, as initially the robot wrongly assumes that is knows where it is. This problem occurs only in limited form in real applications, but serves as a good benchmark for re-localization after failed position tracking attempts[1].

Localization is only one application of state estimation in the robotics domain. *Mapping* refers to the estimation of an environment's map, based on sensor data gathered from known locations. The more complicated problem of mapping with a mobile robot, which does not know its true location but has to localize itself at the same time while building the map, is commonly known as *simultaneous localization and mapping* (SLAM). This topic is covered in more detail in section 3.5. In the following, the localization example will be used to introduce general concepts and algorithms related to state estimation.

An important concept is that of the robot's *belief*, abbreviated as $bel(x)$, which reflects the robot's current knowledge about the world's state x. In probabilistic robotics, this belief is expressed through conditional probability distributions. These assign a probability to all hypotheses, i.e. to all possible configurations in state space. So the belief about the state x_t at time t is a posterior probability based upon all information available so far, which consists of all measurements and control data as explained above and given in equation 3.4.

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{3.4}$$

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) \tag{3.5}$$

Equation 3.5 describes the probability before incorporating the most recent measurement z_t. This is called the *prediction*, since $\overline{bel}(x_t)$ predicts the state at time t based on the belief at time $t-1$ and the current control data. Calculating $\overline{bel}(x_t)$ is done in the motion update. Without loss of generality, this assumes that the robot first executes u_t and then makes the observation z_t. Calculating $bel(x_t)$ from $\overline{bel}(x_t)$ by incorporating z_t is called *correction* and done in the sensor update. Together, the motion and sensor updates describe the mathematical basis of the modeling aspect in the iterative process visualized in figure 3.1.

[1]In robot soccer scenarios the kidnapped robot problem actually occurs in its true form due to incompletely executed wrong referee decisions: An assistant referee may pick up the wrong robot for a penalty, start to carry it away, but then put it back on the field once the mistake is pointed out.

3.2 Recursive Bayes Filtering

This section explains the mathematical principles and assumptions which allow to calculate the beliefs and update steps defined above. Sections 3.3 and 3.4 in turn describe different approximations for the belief in equation 3.4 and the resulting practical filtering implementations.

For a robot to actually compute a belief during its operation, it is undesirable for this computation to depend on all previously acquired information. This is due to the limited memory available to store all incoming data, as well as the increasing computational cost of the belief's calculation for each new time step, if it would depend on the continuously growing input data. The *Markov assumption* is the key to overcoming this problem. It states that "past and future data are independent if one knows the current state x_t" [86, page 33]. This assumes the state x_{t-1} to be *complete*, i.e. that neither past states $x_{1:t-2}$, measurements $z_{1:t-1}$, nor control data $u_{1:t-1}$ carry any additional information useful for the prediction of a future state x_t. Consequences of this assumption are the following two simplifications. First, the prediction $\overline{bel}(x_t)$ in equation 3.6 now only depends on the previous belief $bel(x_{t-1})$ and the current control data u_t. Second, in the measurement process as expressed in equation 3.7, measurements only depend on the state of the environment at that time, since no previous data carries additional information about that state.

$$p(x_t|x_{1:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{3.6}$$
$$p(z_t|x_{1:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \tag{3.7}$$

The resulting dependencies are visualized in the *hidden Markov model* (HMM) in figure 3.2.

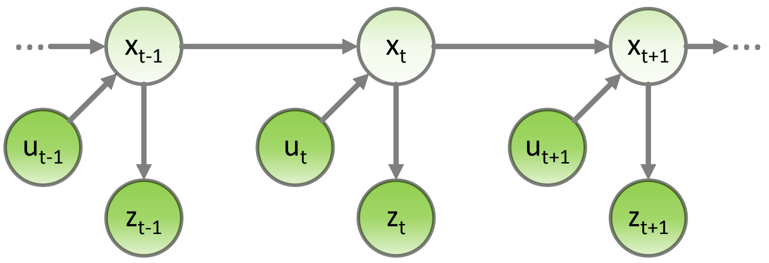


Figure 3.2: Hidden Markov Model of the Bayesian estimation process.

These assumptions allow the recursive computation of the prediction $\overline{bel}(x_t)$ based on the previous belief $bel(x_{t-1})$ and the current control data u_t as follows: Starting from equation 3.8 (cp. equation 3.5), the theorem of total probability gives equation 3.9. Equation 3.10 results from the Markov assumption expressed in equation 3.6 for the first factor, and the conditional independence of the state at time $t-1$ from the future control data u_t in the second factor. The definition in equation 3.4 finally gives equation 3.11.

$$\begin{align}
\overline{bel}(x_t) &= p(x_t|z_{1:t-1}, u_{1:t}) \tag{3.8}\\
&= \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})\ p(x_{t-1}|z_{1:t-1}, u_{1:t})\ dx_{t-1} \tag{3.9}\\
&= \int p(x_t|x_{t-1}, u_t)\ p(x_{t-1}|z_{1:t-1}, u_{1:t-1})\ dx_{t-1} \tag{3.10}\\
&= \int p(x_t|x_{t-1}, u_t)\ bel(x_{t-1})\ dx_{t-1} \tag{3.11}
\end{align}$$

Similarly, the Bayes rule applied on the conditional probability in equation 3.12 (cp. equation 3.4) results in equation 3.13 and in equation 3.14 when abbreviating the normalization factor as η. The same conditional independence of x_{t-1} from u_t as described above, and the assumption of the complete state with respect to observations as in equation 3.7 give equation 3.15, and the definition in equation 3.5 results in equation 3.16.

$$\begin{align}
bel(x_t) &= p(x_t|z_{1:t}, u_{1:t}) \tag{3.12}\\
&= \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \tag{3.13}\\
&= \eta\ p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t}) \tag{3.14}\\
&= \eta\ p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t-1}) \tag{3.15}\\
&= \eta\ p(z_t|x_t)\ \overline{bel}(x_t) \tag{3.16}
\end{align}$$

The resulting two equations 3.11 and 3.16 define the core of the recursive Bayes filtering algorithm. Equation 3.11 defines the motion update and incorporates motion control data and information about the state change from time $t-1$ to t. Equation 3.16 corrects the prediction using the sensor information provided by the measurement z_t. More details on the background and mathematical derivation can be found in [86, section 2.4.3].

An exact computation of the belief according to equations 3.11 and 3.16 is only possible for very special cases, such as simple reasoning on discrete state spaces. In general, further assumptions about state transition probabilities and posterior distributions need to be made to achieve tractable computational complexity. Practical solutions always represent a trade-off between the accuracy of the approximation and its computational efficiency. A further consideration is the ease of implementation, which often has a significant influence on an approach's popularity and success.

Two different classes of Bayes filter implementations for continuous state spaces will be applied repeatedly in the course of this work, namely the Kalman filter and the particle filter. The following sections 3.3 and 3.4 will introduce both concepts, respectively.

In addition to the choice of filter implementation, the probability distributions in equations 3.11 and 3.16 play important roles in the actual estimation system. Given a belief representation and filter algorithm, these probabilities are the central points to apply expert background knowledge to customize the filter for a specific application. In this context, the expression $p(x_t|u_t, x_{t-1})$ in equation 3.11, or more specifically the process specifying this probability distribution for each specific situation, is often referred to as the *motion model.* It provides the means to express the system's locomotion characteristics and intrinsic uncertainty, and to address eventual systematic tendencies, such as a known drift to one side while moving forwards. Chapter 4 covers several aspects concerning motion models and proprioception. Analogically, the probability distribution $p(z_t|x_t)$ in equation 3.16 results from the *sensor model.* The influence of expert knowledge about the sensing process on estimation quality will be one of the aspects addressed in chapter 5.

3.3 Kalman Filter

This section describes the Kalman filter (KF) [46], which is the most established representative of the class of Gaussian filters and is used in various contexts throughout this work. As the name suggests, Gaussian techniques describe the belief as a multivariate normal distribution characterized by their first two moments, the mean μ and covariance P as in equation 3.17.

$$\begin{aligned} bel(x_t) &= N(\mu_t, P_t) \\ &= \frac{1}{\sqrt{2\pi|P_t|}}\; e^{(-1/2(x-\mu_t)^T P_t^{-1}(x-\mu_t))} \end{aligned} \tag{3.17}$$

For a state vector x of dimensionality n, the Gaussian's first moment μ is also a vector of dimension n, and P is a quadratic, symmetric and positive semidefinite $n \times n$ matrix. Naturally, the Kalman filter is exclusively applied in the context of continuous state spaces. It is not suited for discrete or hybrid estimation problems.

The commitment to restrict the representation of belief states to their first two moments yields certain consequences. Foremost, this only allows to describe unimodal probability distributions, i.e. beliefs which only have one maximum. When considering the classification of different localization scenarios in section 3.1 for example, this is almost incompatible with global localization problems, but fits well in case of position tracking. If the approximation is sufficient, however, this representation allows a very efficient computation of the belief in closed form.

The central idea is that a linear transformation of any Gaussian distributed random variable results in another Gaussian distributed random variable. Certain properties of the estimation problem's target system are therefore assumed in order to maintain a Gaussian belief throughout the updates steps:

1. The initial belief $bel(x_0)$ must be Gaussian.
2. The motion model in form of the state transition probability $p(x_t|x_{t-1}, u_t)$ in equation 3.11 must be a linear function, with added Gaussian noise.
3. The sensor model in form of the measurement probability $p(z_t|x_t)$ in equation 3.16 must be a linear function, with added Gaussian noise.

As a linear transformation of a Gaussian distribution results in another Gaussian distribution, such a system can be described by the state transition in equation 3.18 and a measurement function in equation 3.19, and maintain the assumption of Gaussian state and measurement representations throughout the process. Both transformations are linear, but non-deterministic on account of the added Gaussian noise vectors ϵ_t and δ_t, which model the uncertainty involved in both processes and have covariance matrices R_t and Q_t, respectively, and zero means.

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{3.18}$$
$$z_t = C_t x_t + \delta_t \tag{3.19}$$

The Kalman filter estimates the state of such a system by means of the following recursive equations.

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \tag{3.20}$$
$$\overline{P}_t = A_t P_{t-1} A_t^T + R_t \tag{3.21}$$
$$K_t = \overline{P}_t C_t^T (C_t \overline{P}_t C_t^T + Q_t)^{-1} \tag{3.22}$$
$$\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t) \tag{3.23}$$
$$P_t = (I - K_t C_t)\overline{P}_t \tag{3.24}$$

Equations 3.20 and 3.21 perform the motion update on the previous belief state $bel(x_{t-1}) = N(\mu_{t-1}, P_{t-1})$ to calculate the prediction $\overline{bel}(x_t) = N(\overline{\mu}_t, \overline{P}_t)$. Equations 3.23 and 3.24 compute the sensor update and the new belief state $bel(x_t) = N(\mu_t, P_t)$ using the Kalman gain K_t in equation 3.22. This Kalman gain controls how the state estimate is changed by the difference between expected and real measurement. I denotes the identity matrix. A derivation of the Kalman equations can be found in [86, section 3.2.4], as well as a proof of its optimality in case of linear systems as described here. The Kalman filter's computational complexity is lower bounded by approximately $O(k^{2.4})$ for k-dimensional measurements, due to the matrix inversion in equation 3.22, and at least $O(n^2)$ an n-dimensional state space due to the matrix multiplications [86, page 43].

The assumption of linearity of the involved transformations is a crucial aspect of the Kalman filter concept. However, practically relevant estimation problems are rarely linear. Kalman filters nevertheless enjoy ongoing popularity, as they are well understood and still provide good approximations in many scenarios by applying linearization of the non-linear transformations around the state estimate's current mean. Two Kalman approaches for non-linear systems are commonly used and will be described in the following sections, namely the Extended Kalman filter (EKF) in section 3.3.1 and the Unscented Kalman filter (UKF) in section 3.3.2. They differ mainly in the method employed for linearization and the algorithmic and numerical consequences. Both are applied in the course of this work, and the differences in their influence on the estimation quality are evaluated and compared to other relevant factors in section 5.1.

3.3.1 Extended Kalman Filter

The Extended Kalman filter relaxes the strict assumptions that process and measurement model need to be linear transformations as in equations 3.18 and 3.19, and allows general differentiable functions instead:

$$x_t = f(x_{t-1}, u_t) + \epsilon_t \tag{3.25}$$
$$z_t = h(x_t) + \delta_t. \tag{3.26}$$

Equations 3.25 and 3.26 can be used directly to calculate state and measurement predictions for a given belief[2]. For the unknown noise terms ϵ_t and δ_t, their mean is substituted, which is zero as defined above. The modification of equations 3.20 and 3.23 results in equations 3.27 and 3.30. For the predictions and correction of the covariance estimate however, the state transition matrix A_t in equations 3.18 and 3.21 and the measurement transformation matrix C_t in equations 3.19, 3.22 and 3.24 are replaced by Jacobian matrices of the partial derivatives of the functions $f(x_{t-1}, u_t)$ and $h(\overline{x}_t)$, respectively.

The Extended Kalman filter equations are then given analogous to equa-

[2]Note that the measurement function $h(x_t)$ in equation 3.26 implicitly uses knowledge about the environment to predict expected observations. The explicit inclusion of the environment map m as a function parameter, i.e. as $h(x_t, m)$, is omitted here.

tions 3.20 to 3.24,

$$\overline{\mu}_t = f(\mu_{t-1}, u_t) \tag{3.27}$$

$$\overline{P}_t = F_t P_{t-1} F_t^T + R_t \tag{3.28}$$

$$K_t = \overline{P}_t H_t^T (H_t \overline{P}_t H_t^T + Q_t)^{-1} \tag{3.29}$$

$$\mu_t = \overline{\mu}_t + K_t \, (z_t - h(\overline{\mu}_t)) \tag{3.30}$$

$$P_t = (I - K_t H_t) \overline{P}_t \tag{3.31}$$

where F_t and H_t are the process and measurement Jacobian matrices as defined in equations 3.32 and 3.33, with indices i and j denoting the corresponding component of each element.

$$F_t = \left[\frac{\partial f_i(\mu_{t-1}, u_t)}{\partial \mu_{t-1,j}} \right]_{(i,j)} \tag{3.32}$$

$$H_t = \left[\frac{\partial h_i(\overline{\mu}_t)}{\partial \overline{\mu}_{t,j}} \right]_{(i,j)} \tag{3.33}$$

In effect, the Extended Kalman filter constructs linear approximations of the functions h and f by applying a first order Taylor expansion, i.e. using the functions' values and slopes at the current state estimate. In contrast to the classic Kalman filter for linear systems, this application to non-linear systems is not formally optimal. The quality of the approximation, and therefore of the resulting estimation system, depends on how good this linearization approximates the true non-linear functions around the current state estimate. For many systems, this approximation is sufficient for small time steps, which together with its simplicity and computational efficiency explains the Extended Kalman filter's great popularity as a tool for state estimation in various application scenarios.

3.3.2 Unscented Kalman Filter

An alternative to the Extended Kalman filter comes in form of the Unscented Kalman filter (UKF) [42]. The development of the UKF is motivated by a number of potential disadvantages of the EKF as argued in [43]:

1. If the error propagation is not approximated well by a linear function, the linearization using the Taylor expansion around the mean may also produce only a poor approximation of the true error propagation.

2. There are state spaces in which Jacobian matrices do not exist or are not meaningful in every case, e.g. around discontinuities or singularities.[3]

[3] A practical example for such a system is the localization of transport cases as motivated in section 2.2 and derived in section 4.1.

3. The derivation of the Jacobian matrices can result in several pages of dense algebra[4], and the subsequent conversion to code can be difficult and error prone. Such human coding errors will degrade the estimation performance, but may not change the system's behavior completely in most cases, so that identification and debugging is difficult at best, especially in the absence of any measure of expected performance.

Instead of using the Taylor expansion to determine the first two moments analytically, the UKF avoids an explicit linearization by transforming the Gaussian using a deterministic sampling process called *unscented transform*. Its basic idea is to represent a Gaussian $N(\mu, P)$ by a set of weighted points, called *sigma points*, which are chosen to have the mean μ and covariance P. As illustrated in figure 3.3, the non-linear function is then applied to each sigma point separately, and the transformed Gaussian is extracted from the mean and covariance of the transformed points.

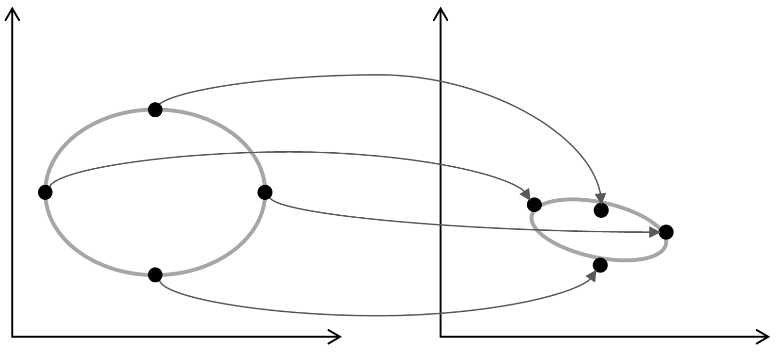


Figure 3.3: Illustration of the unscented transform applied to a Gaussian distribution.

The minimum number of sigma points necessary to capture an n-dimensional Gaussian distribution's mean and covariance is $n+1$, as shown in [43]. The more robust standard approach uses $2n+1$ sigma points, as described in [42], which are placed at the mean and symmetrically along the main axes of the covariance, which are given by the columns of its matrix square root. The n-dimensional Gaussian $N(\mu, P)$ is then represented by the following set $\mathcal{X}$ of sigma points $\mathcal{X}^{[i]}$:

$$\begin{aligned}
\mathcal{X}^{[0]} &= \mu \\
\mathcal{X}^{[i]} &= \mu + (\sqrt{(n+\lambda)P})_i && \text{for } i = 1, \ldots, n \\
\mathcal{X}^{[i]} &= \mu - (\sqrt{(n+\lambda)P})_{i-n} && \text{for } i = n+1, \ldots, 2n
\end{aligned}$$

[4]See appendix B for an example.

and corresponding weights $\mathcal{W}^{[i]}$:

$$\begin{aligned}
\mathcal{W}^{[0]} &= \lambda/(n+\lambda) \\
\mathcal{W}^{[i]} &= 1/2(n+\lambda) && \text{for } i = 1,\dots,2n.
\end{aligned}$$

with λ as a scaling parameter determining the spread of the sigma points[5].

For a non-linear system, which can be described by equations 3.25 and 3.26, the unscented Kalman filter is given by equations 3.34 to 3.43, where $\mathcal{X}_t = \{\mathcal{X}_t^{[i]}\}$ and $\overline{\mathcal{X}}_t = \{\overline{\mathcal{X}}_t^{[i]}\}$ are the sigma point sets of $N(\mu_t, P_t)$ and $N(\overline{\mu}_t, \overline{P}_t)$, respectively.

$$\overline{\mathcal{X}}_t^* = f(\mathcal{X}_{t-1}, u_t) \tag{3.34}$$

$$\overline{\mu}_t = \sum_{i=0}^{2n} \mathcal{W}^{[i]} \overline{\mathcal{X}}_t^{*[i]} \tag{3.35}$$

$$\overline{P}_t = R_t + \sum_{i=0}^{2n} \mathcal{W}^{[i]} (\overline{\mathcal{X}}_t^{*[i]} - \overline{\mu}_t)(\overline{\mathcal{X}}_t^{*[i]} - \overline{\mu}_t)^T \tag{3.36}$$

$$\overline{\mathcal{Z}}_t = h(\overline{\mathcal{X}}_t) \tag{3.37}$$

$$\overline{z}_t = \sum_{i=0}^{2n} \mathcal{W}^{i} \overline{\mathcal{Z}}_t^{[i]} \tag{3.38}$$

$$S_t = Q_t + \sum_{i=0}^{2n} \mathcal{W}^{[i]} (\overline{\mathcal{Z}}_t^{[i]} - \overline{z}_t)(\overline{\mathcal{Z}}_t^{[i]} - \overline{z}_t)^T \tag{3.39}$$

$$\overline{P}_t^{x,z} = \sum_{i=0}^{2n} \mathcal{W}^{[i]} (\overline{\mathcal{X}}_t^{[i]} - \overline{\mu}_t)(\overline{\mathcal{Z}}_t^{[i]} - \overline{z}_t)^T \tag{3.40}$$

$$K_t = \overline{P}_t^{x,z} S_t^{-1} \tag{3.41}$$

$$\mu_t = \overline{\mu}_t + K_t\,(z_t - \overline{z}_t) \tag{3.42}$$

$$P_t = \overline{P}_t - K_t \overline{P}_t^{x,z} K_t^T \tag{3.43}$$

In this formulation, equations 3.34 to 3.36 describe the motion update, and equations 3.37 to 3.43 the different parts necessary for the sensor update. This way, it is obvious how to use part of the algorithm for only one of the updates in case the other one only involves linear transformations and can be handled more efficiently using the classic Kalman updates described in equations 3.20 to 3.24. An equivalent, more efficient implementation uses the sigma point set $\overline{\mathcal{X}}_t^*$ directly for the sensor update instead of extracting $\overline{\mu}_t$ and $\overline{P}_t$, and recalculating $\overline{\mathcal{X}}_t$ from $N(\overline{\mu}_t, \overline{P}_t)$.

A drawback of the UKF is that it is more prone to numerical instability compared to the EKF, which is caused by the Cholesky decomposition used for the matrix square root in the sigma point computation. This, however,

[5] $(A)_i$ denotes the ith column of matrix A.

is countered by the algebraically equivalent, but numerically more stable square root formulation of the UKF [90]. Similar square root variants also exist for the EKF.

An additional flexibility is allowed in the UKF formulation in [43], where the noise term is no longer assumed to be additive, but is an additional argument of the non-linear transformation instead, i.e. changing the assumed models in equations 3.25 and 3.26 to those of equations 3.44 and 3.45, respectively. In this case, the involved state and covariances can be augmented by the noise terms' means and covariances, and sigma points are generated using the augmented Gaussian distributions.

$$x_t = f'(x_{t-1}, u_t, \epsilon_t) \tag{3.44}$$

$$z_t = h'(x_t, \delta_t) \tag{3.45}$$

Due to this argumentation, the UKF is often described as superior to the EKF in terms of estimation quality, which can also be shown using the popular showcase of a tracking task with measurements in polar coordinates [42, 43]. The improvement of an UKF over an EKF in practical applications, especially in contrast to other filter design choices, will be covered in more detail in section 5.1 using the example of humanoid robot localization.

3.4 Particle Filter

An alternative approach to the implementation of the Bayes filter concept is the particle filter. Particle filters are generally considered to be easy to understand and implement, quickly provide reasonably good results, and are widely used in the robotics domain. Based on [86], this section only gives a brief overview on this topic and on certain techniques, which enable them to work very efficiently, i.e. with only a very low number of particles. This work does not extend particle filtering algorithms themselves, but uses existing state of the art particle filters as a basis on several occasions: Section 5.2 uses active sensing to optimize filtering results, and chapter 7 applies techniques, which were originally developed in the particle filtering domain, to multi-hypotheses Kalman filtering and uses an existing state of the art particle filter for the evaluation and comparison of the estimation quality and computational performance.

3.4.1 Basic Algorithm

The particle filter is another implementation of the Bayes filter concept. While the Kalman filter represents the belief $bel(x_t)$ in the parametric form of a Gaussian distribution, the particle filter uses a set of particles $\mathcal{X}_t =$

$\{\mathcal{X}_t^{[i]}\}$, which are sampled from this belief distribution. In this, particles are similar to sigma points of the UKF in the previous section, as the terminology suggests, but while sigma points are sampled deterministically assuming a specific probability distribution, i.e. a Gaussian, particles are drawn randomly from the belief distribution without this assumption of an underlying parametric form. Therefore, the particle distribution only approximates the original probability distribution, but at the same time it is possible to represent a much wider range of distributions, including asymmetrical or multi-modal ones.

Each particle $\mathcal{X}_t^{[i]}$ represents a hypothesis for the true world state at time t. As the particle set $\mathcal{X}_t$ approximates the belief, the likelihood of a hypothesis to be included in this set is proportional to $bel(x_t)$:

$$\mathcal{X}_t^{[i]} \propto p(x_t|z_{1:t}, u_{1:t}). \tag{3.46}$$

Therefore, the density of particles in an area of the state space intuitively correlates to the likelihood of the true state being in this region. Of course, this is only asymptotically true for an infinite number of particles, but the difference can be neglected as long as the particle set is sufficiently large. The necessary size, however, depends on the properties and dimensionality of the estimation problem.

In general, the number of particles needs to increase exponentially with the estimation problems dimension [86, page 437]. This becomes clear by the following reasoning: Consider a certain number of particles which cover all relevant hypotheses in n dimensions. If another dimension is added to the estimation problem, and this dimension for itself allows for 10 likely hypotheses, then for each particle in the n-dimensional state space there are 10 new particles needed to cover the $(n+1)$-dimensional state space. On the other hand, the computations necessary for each particle are comparatively simple, so particle filters are easy to implement and offer an efficient solution for many low-dimensional estimation tasks.

The basic idea of all particle filter variants is given in algorithm 1, which again implements the Bayes filter concept introduced in section 3.2. Its input are last time step's belief $bel(x_{t-1})$ in form of the last particle set $\mathcal{X}_{t-1}$, the new control u_t, and sensor data z_t. The motion update can be found in line 4. The resulting particles $\overline{\mathcal{X}}_t$ are distributed according to $\overline{bel}(x_t)$ (cp. equation 3.11). Line 5 implements the sensor update[6]. To ensure that the new particle distribution $\mathcal{X}_t$ reflects the posteriori distribution $bel(x_t) = \eta\ p(z_t|x_t)\ \overline{bel}(x_t)$ (cp. equation 3.16), the particles are resampled in lines 9 to 13 according to the importance weights $\mathcal{W}_t^{[j]}$ calculated in the

[6] Note that just a hypothesis' likelihood, but not its state space coordinates, change with the availability of new sensor information. This will become important later in SLAM contexts in section 3.5 and chapter 8, and likewise in one example in section 4.1.

sensor update. This central element of the particle filter algorithm is called *resampling* or *importance sampling.*

Algorithm 1: Particle filter algorithm, based on the *Sampling Importance Resampling* (SIR) filter [75][86, page 98].

```
   Input: $\mathcal{X}_{t-1}$, $u_t$, $z_t$
1  $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$;
2  $\mathcal{W}_t = \emptyset$;
3  for $i = 1 \ldots |\mathcal{X}_{t-1}|$ do
4      sample $\overline{\mathcal{X}}_t^{[i]} \propto p(x_t|\mathcal{X}_{t-1}^{[i]}, u_t)$ ;
5      $\mathcal{W}_t^{[i]} = p(z_t|\mathcal{X}_t^{[i]})$ ;
6      Add $\overline{\mathcal{X}}_t^{[i]}$ to $\overline{\mathcal{X}}_t$;
7      Add $\mathcal{W}_t^{[i]}$ to $\mathcal{W}_t$;
8  end
9  for $i = 1 \ldots |\overline{\mathcal{X}}_t|$ do
10     draw $\overline{\mathcal{X}}_t^{[j]} \in \overline{\mathcal{X}}_t$ with probability $\propto \mathcal{W}_t^{[j]}$;
11     $\mathcal{X}_t^{[i]} = \overline{\mathcal{X}}_t^{[j]}$;
12     Add $\mathcal{X}_t^{[i]}$ to $\mathcal{X}_t$ ;
13 end
14 return $\mathcal{X}_t$
```

An implementation would be possible without the resampling step by initializing the importance weights with 1 and updating them multiplicatively:

$$\mathcal{W}_t'^{[i]} = \mathcal{W}_{t-1}'^{[i]} \, p(z_t|\mathcal{X}_t^{[i]}). \tag{3.47}$$

This original version of the particle filter is called *Sequential Importance Sampling* (SIS). Such a filter however would require substantially more particles, as many of them would end up in low probability regions of the state space, i.e. with weights near zero. The resampling step instead focuses particles on regions of high probability, thus improving its efficient utilization of the limited number of particles.

For many real applications it is often important not only to know the localization belief distribution, but also to output a single discrete localization result for higher level modules to work with. In simple scenarios, a weighted average of all particles can provide a reasonable result. Regarding the possible multi-modal nature of this belief representation, it is usually reasonable to employ clustering techniques to extract a maximum likelihood estimate from the particle set based on the highest local particle density.

3.4.2 Sampling Variance and Particle Diversity

The non-deterministic nature of the sampling also possesses drawbacks in form of the variance or deviation introduced by the sampling process, i.e. the slight differences between repeated approximations of an identical source probability distribution. This is in fact amplified by the repeated resampling step during each filter update. It can be illustrated using the example of a robot, which is temporarily stationary and only makes observation producing the same weights for all particles. In this example, $bel(x_t)$ should be identical to $bel(x_{t-1})$. In the basic implementation presented in algorithm 1, no new particles would be introduced in any of the update steps. However, the resampling would occasionally miss to reproduce one of the particles. Over time, this process unavoidably leads to an unreasonable decay of the particle set's diversity.

Two different techniques are possible and commonly employed to counter sample set deprivation. The first one is minimizing the sampling variation itself, e.g. by using low variance sampling methods [86, page 110]. The second one compensates the effects of degrading particle diversity by artificially inserting additional diversity during the resampling step, e.g. by adding new particles at random positions in state space [26, 32].

The last measure presents a certain problem for particle filters with a low number of particles, as regularly replacing even only 10 particles by random ones will seriously affect the filter stability if the complete particle set consists of only 100 particles. At the same time, 10 random particles have only a very low likelihood of turning up in relevant poses if large portions of the state space are made up of low probability area.

A solution for this problem is to restrict the probability distribution from which those additional random particles are drawn. *Sensor resetting* [57] is a technique which samples these additional particles from the last measurements, i.e. directly from the distribution $p(x|z_t)$. At the same time, this elegant solution is also able to handle localization losses and the kidnapped robot problem introduced in section 3.1. Instead of randomly sampling $p(x|z_t)$, a simplified but efficient implementation of sensor resetting is to add particles deterministically at maximum likelihood positions calculated from the last measurements, i.e. by multiangulation or multilateration in case of different landmark observations. Those additional particles obviously result in a deviation from the true posterior distribution. A formally correct version of the sensor resetting idea has been implemented by [87] as *Mixture Monte Carlo* localization, which generates a certain number of particles by sampling from the sensor model and weights those particles using the motion model. Thus, for an infinite number of particles, the belief still converges to the true posterior.

3.4.3 Influence of False Positives

An important aspect for localization in real-world scenarios is the robustness not only to noisy measurements, but also to *false positives.* False positive perceptions are those observations which the perception process incorrectly registers as measurements of a known feature in the environment; instead, they do not correspond to any feature in the robot's map. These are usually caused by sensor errors or random clutter in the environment.

The classic Kalman filter described in section 3.3 does not include an explicit formulation for false positive observations. The usual compensation in scenarios with expected false positives is to enlarge the measurement noise. Of course, this reduces the Kalman filter's reactivity not only for false positives, but also for correct measurements.

For particle filters, the sensor model $p(z_t|x_t)$ used in algorithm 1 in line 5 (or its adaptation in equation 3.47) is not restricted to Gaussian measurement noise. A common choice is to model two components, one for correct observations and one for false positives, as in equation 3.48 with k being the measurement's dimensionality. The first one is usually implemented as a Gaussian for the deviation of the actual measurement z_t from the expected measurement $h(x_t)$ of the corresponding landmark. The probability for false positive observations is a uniform ϵ_0.

$$p(z_t|x_t) = (1-\epsilon_0)\left(\frac{1}{\sqrt{(2\pi)^k\,|Q_t|}}e^{-\frac{1}{2}(z_t-h(x_t))^T Q_t^{-1}(z_t-h(x_t))}\right)+\epsilon_0 \quad (3.48)$$

For small particle numbers, however, few false positive observations or even just high noise levels in the measurements can still decrease the weights of particles around the true position enough to loose the coverage of that area after resampling. This might happen if a random sample receives a favoring weight while most others' weights are only updated with a likelihood close to the false positive probability ϵ_0. In this case, a classic low variance resampler would instantly replace a number of those "low weight" samples with the random "good" one.

Techniques like *temporal smoothing* and *lazy resampling* strive to prevent such behavior [65]. Temporal smoothing calculates a particle's weight as a product of separate weight factors for each observation class, and smooths these separate weights over time by capping the allowed de- or increase per single measurement. When a given class is not observed, the corresponding weight ages to 1 to loose influence on the overall weight product. Lazy resampling in turn is a resampling strategy which restricts how often a certain particle is allowed to be duplicated in a single resampling step. A more detailed derivation and evaluation can be found in [65, 66].

Overall, state of the art particle filters employing the techniques briefly presented in this and the previous section are able to perform well even with relatively little particles, e.g. with only 100 particles in a RoboCup SPL scenario (cp. section 2.1). Compared to classic Kalman filters, they show an increased robustness to false positives and high sensor noise. Such a filter is the basis for active sensing algorithms presented in section 5.2, and provides the baseline for an evaluation of a different approach's localization quality in chapter 7.

3.5 Algorithms for Simultaneous Localization and Mapping

Localization in a known environment is used as the state estimation example in the previous sections. The Bayes filter concept and algorithms, however, are applicable beyond this low-dimensional pose estimation task. In the case that a robot does not know anything about its environment prior to its deployment, it also has to build a map of the world along with finding its position in it. As mentioned in section 3.1, this is commonly called *simultaneous localization and mapping* (SLAM). The inclusion of additional mapping variables considerably increases the state's dimension n and therefore the estimation problem's computational load according to the chosen algorithm's complexity.

SLAM is an ongoing research challenge, both for its computational and conceptual complexity, as well as for the compelling idea of robots, which can perform given tasks in every new environment without any prior knowledge about it. Out of the various approaches to and forms of the general SLAM problem, only a subset is introduced here as a background to chapter 8. In particular, only the *online SLAM problem* is considered, i.e. continuously estimating the best guess of an environment map and a robot's position therein during operation. Furthermore, maps in this context consist of a number of discrete features, which may be ambiguous but allow the computation of correspondence likelihoods.

In the following sections 3.5.1 and 3.5.2, two popular SLAM approaches are introduced, which are based on the Kalman and particle filter algorithms covered in sections 3.3 and 3.4, respectively.

3.5.1 EKF-SLAM

The EKF-SLAM is a straight forward application of the Kalman filter to the SLAM problem[7]. The state is extended beyond the pose $(p_x, p_y, p_\theta)^T$ to also include models for each landmark, e.g. consisting of the landmark's position estimate $(m_{i,x}, m_{i,y})$ and its signature s_i which is used to identify it among other landmarks. It is possible, that a landmark's appearance can be perceived without any uncertainty, or that it falls into a clearly separable category which does not need to be modeled. In these cases, the signature might be omitted from the state and filed separately without associated uncertainty.

[7]The idea of applying Extended Kalman filters to the SLAM problem predates the emergence of Unscented Kalman filters. The EKF-SLAM is therefore still predominant in literature, even if the same concept can naturally be implemented using the unscented transform, too. Eventual differences between EKF and UKF implementations of the same concept are not discussed here any further.

In general, for N landmarks with positions $(m_{i,x}, m_{i,y})^T$ the state x is extended to

$$x = (p_x, p_y, p_\theta, m_{1,x}, m_{1,y}, s_1, m_{2,x}, m_{2,y}, s_2, \dots, m_{N,x}, m_{N,y}, s_N)^T. \tag{3.49}$$

When operating in a totally unknown environment, a robot initially starts with the state $x_0 = (p_x, p_y, p_\theta)^T = (0, 0, 0)^T$ and includes additional landmarks when first perceiving them.

The working principle of the EKF-SLAM algorithm is illustrated in figure 3.4. The true path, which the robot traveled so far, is indicated by a black dashed line. Red circles mark the positions of eight previously unknown point landmarks in the environment. For each time frame, the uncertainty ellipses visualize the robot's pose and the subset of those landmarks, which are included in the current state at that time. If an observation is made in a time frame, it is represented by a red line between the robot's and the observed landmark's positions. The accumulated motion execution uncertainty propagates into the landmark position's model. As the robot progresses, it only observes new landmarks at first. That means, the first landmark is observed twice, then only the second one, then only the third one, and so on. Only after observing the eighth landmark and taking the third turn, the robot finally observes the first landmark again.

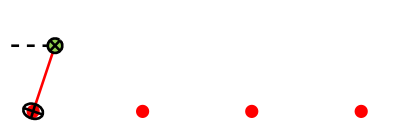
The overall uncertainty in the modeled state therefore builds up, until it can be reduced by observing one of the landmarks again which have been mapped earlier. Figures 3.4e and 3.4f present the state before and after such a new observation of an earlier observed landmark. Their comparison shows the refinement not only of the current pose estimate, but also of the estimate of all landmarks which have been observed previously.

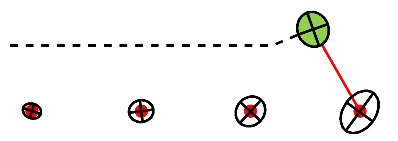
The correction of the whole map along with the robot's current pose estimate is possible through the modeling of the estimate's uncertainty in a joint covariance matrix. The drawback of this, however, is the growth of the state's dimensionality linear in the number of landmarks, and the resulting squared growth of necessary memory space and computational complexity. Therefore, the application of Kalman filters becomes unfeasible for SLAM problems with a large number of landmarks, but offers a simple and reliable solution for moderately sized maps.

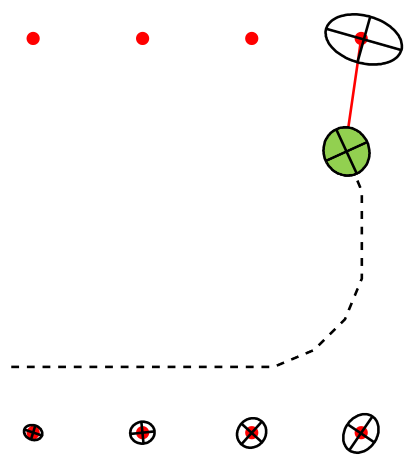
3.5.2 FastSLAM

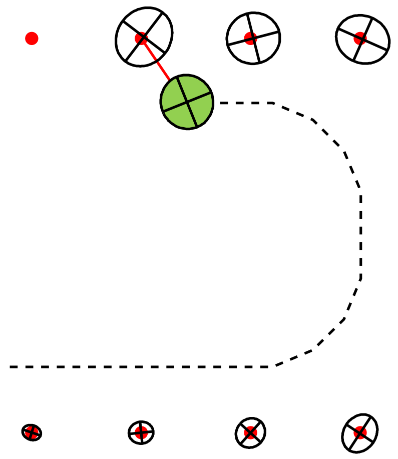
In contrast to the previous section, the SLAM task cannot be addressed with particle filters directly, as the higher dimensionality of the SLAM problem would require too many particles to cover all relevant hypotheses[8]. Instead it is possible to exploit the stochastic characteristics of the SLAM problem with known correspondences to achieve a more efficient implementation.

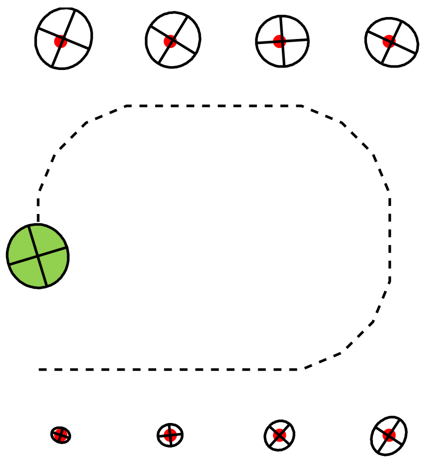
[8]The necessary number of particles increases exponentially with the dimension of the estimation task, as previously mentioned in section 3.4.1.


(a) First landmark observation.


(b) Mapping of the forth landmark.


(c) Uncorrected pose uncertainty propagates into landmark models.


(d) Pose uncertainty grows as long as only new landmarks are observed.


(e) Just before closing the loop.

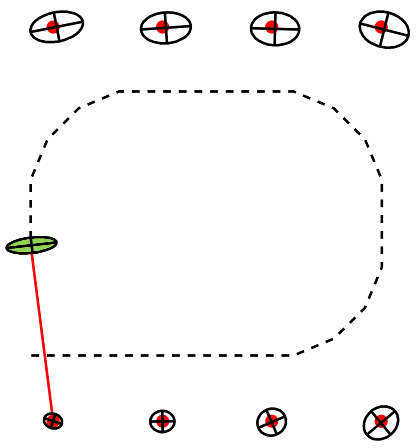

(f) After closing the loop.

Figure 3.4: Illustration of the EKF-SLAM algorithm at work.

When examining the full SLAM posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$, i.e. the estimation of the complete map and the robot's whole path in it, not all aspects of the state $x = (p_x, p_y, p_\theta, m_{1,x}, m_{1,y}, s_1, m_{2,x}, m_{2,y}, s_2, \ldots, m_{N,x}, m_{N,y}, s_N)^T$ (cp. equation 3.49) depend upon each other. Assuming that the origins of all measurements of a certain landmark are known, then this landmark's position can obviously be estimated independently of all other landmarks. Consequently, if the robot's full path $(p_x, p_y, p_\theta)^T_{i:t}$ is known, then all landmark positions $(m_{i,x}, m_{i,y})^T$ can be estimated independent of each other. This allows to express the SLAM posterior in a factored form as in equation 3.50.

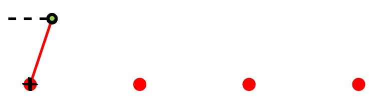
$$p(x_{1:t}|z_{1:t}, u_{1:t}) = p((p_x, p_y, p_\theta)^T_{i:t}|z_{1:t}, u_{1:t}) \prod_{i=1}^{N} p((m_{i,x}, m_{i,y})^T|z_{1:t}, u_{1:t}) \tag{3.50}$$

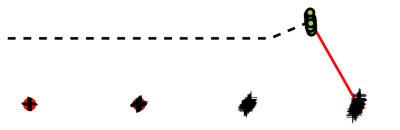
This is relevant in the particle filter context. As explained earlier in section 3.4.1, only a particle's likelihood, but not its state space coordinates, change with the availability of new sensor information. Thus, each particle represents a hypothesis not only for the current pose, but for the robot's path up to this point, i.e. $p((p_x, p_y, p_\theta)^T_{i:t}|z_{1:t}, u_{1:t})$. As a consequence, it is possible to model these path hypotheses using a particle filter, with each particle additionally modeling all landmarks using separate filters. This general technique is known as a *Rao-Blackwellized particle filter*.

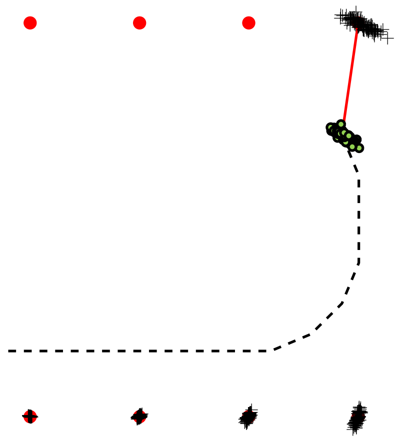
The FastSLAM algorithm [63],[86, chapter 13] applies this idea of factoring the posterior to SLAM, resulting in a particle filter which models the robot's localization and path hypotheses, and in which each particle carries a number of Gaussians to model landmark positions and their current uncertainty. Each new measurement is used in two ways, first to update the particle's likelihood, and then to refine the particle's model of the belonging landmark. The latter is identical to a Kalman filter's sensor update step (cp. equations 3.22 to 3.24). As neither the robot's motion nor the passing of time affects the landmarks' positions, their models do not need a motion update.

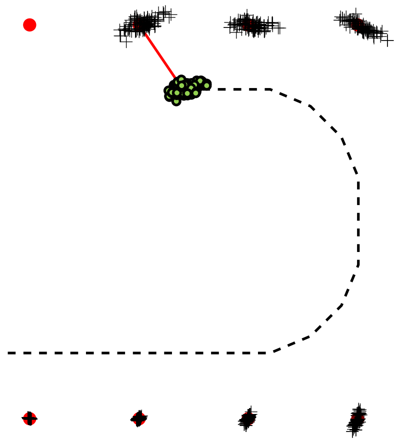
Figure 3.5 illustrates the working of a FastSLAM filter in a scenario identical to and with the same input data as in the one in figure 3.4. Particles are represented as green circles, and the means of their landmark models as black crosses. As in figure 3.4, the figures 3.5e and 3.5f show the SLAM estimation before and after the observation of a landmark which has been mapped earlier in the experiment. With the collapse of the particle set, also the possible landmark positions are reduced to the ones which are closest the the real positions.

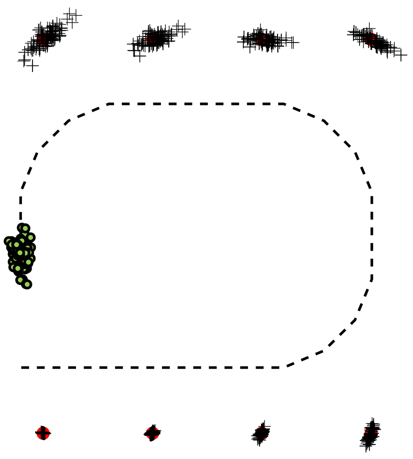
Two aspects highlight the computational efficiency of the FastSLAM algorithm: Firstly, its particle filter part only covers the localization aspect of


(a) First landmark observation.


(b) Mapping of the forth landmark.


(c) Uncorrected pose uncertainty propagates into landmark models.


(d) Pose uncertainty grows as long as only new landmarks are observed.


(e) Just before closing the loop.

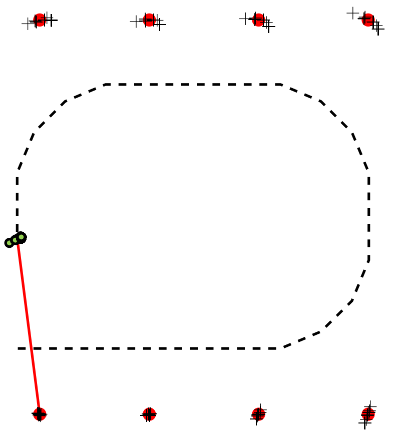

(f) After closing the loop.

Figure 3.5: Illustration of the FastSLAM algorithm at work.

the filter, therefore much fewer particles are necessary compared to applying it to the full state including all landmark dimensions. Basically, those are only as many particles as would be necessary to handle the equivalent localization problem without the simultaneous mapping task. Secondly, the state's representation grows only linearly with the inclusion of new landmarks into the map. This latter point makes the FastSLAM approach especially important for large maps.

Chapter 4

Proprioception

Proprioception is the sensing of one's own body. This includes the awareness about its configuration and its kinematic and dynamic state, i.e. where each body part is relative to the others, which forces and moments are applied, and how these result in movements relative to the surrounding world. In human beings, proprioception is partly a conscious, but mostly an unconscious process. One example is the tendency to adjust the head orientation to bring the eyes on a level with the horizon, independent of the torso's tilt.

In a robotics context, proprioception can be achieved by measuring forces, rotational velocities, pressures, and joint angles, and then calculating either analytically or probabilistically the robot's kinematic and dynamic configuration, orientation and position. In chapter 3 a reasonable sense of the own posture and movement are a prerequisite for Bayes filter localization. This is provided in form of odometry, which serves as an input to the motion update, and in the mapping between measurement and state space in the sensor model. Thus, knowledge about the proprioception process and its inherent uncertainties and shortcoming is essential for the proper design of localization algorithms.

This chapter provides an insight to the place of proprioception in the Bayes filter theme, thus aiding the understandability of further chapters. Section 4.1 presents the structurally simplest possible example of a device with zero degrees of mobility and a single 3-dimensional accelerometer as its only information source for the task of localizing on an intralogistic transportation system in the application scenario described in section 2.2. Even this setup offers different possible design choices and interpretations of the proprioceptive measurements as either part of the control input or a measurable part of the state.

The significantly more complex proprioception processes involved in humanoid robots are analyzed in section 4.2. Motion generation in humanoid robots, biped walking and the difficulties of measuring biped odometry are covered briefly to establish the causes of the humanoid proprioception uncertainties. The properties established here will form the basis for various methods applied throughout this work, namely for sensor model design in chapter 5 and as a motivation for design choices in chapters 7 and 8.

4.1 Intralogistic Tracking

[1] This section uses the scenario described in section 2.2 to illustrate different ways to interpret proprioceptive information in the Bayes filter theme. For this, the application is analyzed from a robotics viewpoint, though the mobile device in question lacks any actuation like a normal robotic system. A single proprioceptive sensor presents its sole information source to perform position tracking.

4.1.1 Problem Analysis

The illustration of an example environment to localize in is given in section 2.2 and repeated here in figure 4.1 for convenience (cp. figure 2.7). As the mobile sensor unit can only move along a very restricted space, i.e. along the roller conveyor, it makes sense to reflect this by restricting the state space accordingly. Instead of using a full metric state space, a majority of which is inaccessible, a topological formulation offers a more condensed representation.

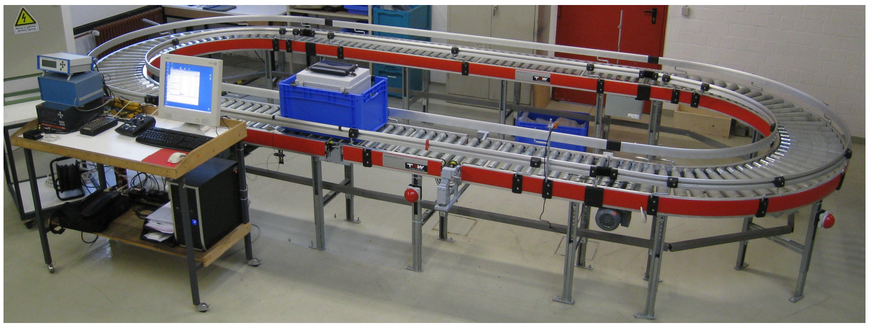


Figure 4.1: Roller conveyor in an experimental setup (cp. figure 2.7).

The state space will be represented as a directed graph consisting of edges and nodes. An edge describes a segment of the system's track with additional data attached to it. A node represents the connection between the segments and has no physical expansion. In complex systems a node can connect more than two modules which will apply to junctions, sorters or pushers.

[1]Parts of this section are based on an article in the Proceedings of the IEEE International Conference on Automation and Logistics 2010. The corresponding publication, see [53], is available at `http://ieeexplore.ieee.org`.

Each segment has a number of properties: The unique identification number of the segment, its length and the reciprocal of the radius of the segment, which can also be understood as the curvature. A straight segment has an infinite radius which causes the curvature to become zero. Considering a curve segment, the length of the segment refers to the mean circular path between the outer and inner border of the segment. Intralogistic systems can be arranged as an open configuration or a closed configuration. In an open configuration the containers arrive on the system at a source and leave the system at a sink though more than one source or sink can exist in such a configuration. The main characteristic of this kind of configuration is that one container which is traveling from a source to a sink never passes one and the same segment twice.

A different configuration is called closed or loop configuration. The handled containers arrive at and leave the system at sources and sinks as well but because of the loop-based topology of the system a container can pass the same segment more than once. Figure 4.2 shows a simple layout of such a roller conveyor with a loop configuration which is composed of four segments. Accordingly, table 4.1 shows its tabular representation, which will be used for the solution explained next.

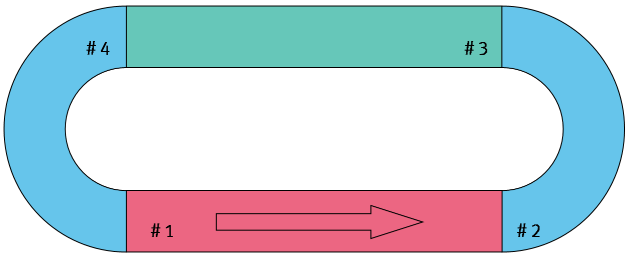


Figure 4.2: Example configuration of a roller conveyor.

ID #	Length [m]	Curvature c_i [1/m]	Description
1	3.125	0	Straight
2	3.142	1	Curve
3	3.125	0	Straight
4	3.142	1	Curve

Table 4.1: Tabular representation of the configuration in figure 4.2.

For the mapping of disturbances or areas of possible malfunctioning on the conveyor track, it is necessary for the mobile sensor unit to know its position in the track's map. While creating this map itself from sensor data is possible to a certain degree, in this context, very precise information is commonly available, e.g. in form of printed building plans or even as electronic design layouts. It can therefore be assumed that a-priori knowledge in the form of a given map as described above is available. This assumption reduces the task of localization to 1-dimensional positions on a sequence of map segments and decisions at eventual intersections.

In easy scenarios the trivial approach would consist of using the measured sensor data directly to infer the unit's movement and based on this its position. This approach is known as dead reckoning (cp. section 3.1). In case of an acceleration sensor the measured acceleration in forward direction can be integrated once to yield the velocity and again to get the unit's position. This procedure is obviously error-prone, since the integration of sensor noise produces a continuously increasing error in the position estimation. Additionally it is necessary for the sensor to be calibrated perfectly due to the same reason. Systematic error introduced by imperfect calibration produces significant drift in the estimation result. For a system to be both cheap and reliable it needs to be robust against those deficiencies.

The next section will present a probabilistic formulation to solve the localization problem as an offline batch computation on a complete sensor data set, as done in the context of the collaborative research centre 696 "Logistics on Demand", as well as the necessary complexity reduction to allow an efficient computation of the resulting optimization problem [53]. A Bayes filter formulation to solve the localization task is described in section 4.1.3, and implementations of the different filters presented in chapter 3 are discussed. As online algorithms, these solutions have the advantages not to grow exponentially in complexity relative to the duration of a localization test run, and to be able to provide results in real-time. Section 4.1.4 presents a brief consideration on the suitability of these filter implementations for the original task of mapping disturbances and potential failure areas on conveyor tracks.

4.1.2 Offline Formulation

A stochastic measurement model can be defined as the conditional probability $p\left(z_t|x_t, m\right)$ of a measurement z_t at time t under the assumption of a position x_t in a map m. A complete representation of the state of the mobile sensor unit is given by $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$ with the unit's position y_t along the track, velocity and acceleration, respectively. Note that a state x_t can be calculated once x_0, $\ddot{y}_{1:t}$ and potential choices at intersections are known. The unit is externally propelled, exercises no own control over its motion, and no friction needs to be taken into account, so discretizing the kinematic motion equation results in the process model given in equation 4.1.

$$
\begin{aligned}
x_t = \begin{pmatrix} y_t \\ \dot{y}_t \\ \ddot{y}_t \end{pmatrix} &= f(x_{t-1}, \epsilon_t) \\
&= \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} x_{t-1} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \epsilon_t \qquad (4.1)
\end{aligned}
$$

Given a position y_t and a map m as described in section 4.1.1, the map segment m_i on which the sensor unit is located at time t can be identified. Knowledge of the sensor's characteristics allows to infer $p\left(z_t|x_t, m\right)$: Measurements of an acceleration sensor positioned at the unit's center are generated according to

$$
\begin{aligned}
z_t = \begin{pmatrix} a_{1,t} \\ a_{2,t} \\ a_{3,t} \end{pmatrix} &= h(x_t, m, \delta_t) \\
&= \Omega \left[\begin{pmatrix} \ddot{y}_t \\ \dot{y}_t^2 / r_i \\ g \end{pmatrix} + \delta_t \right] \\
&= \Omega \left[\begin{pmatrix} \ddot{y}_t \\ \dot{y}_t^2 c_i \\ g \end{pmatrix} + \delta_t \right] \qquad (4.2)
\end{aligned}
$$

with the sensor's rotation relative to the unit's coordinate frame given as the rotation matrix Ω and added measurement noise δ_t in the form of a multi-variate Gaussian with zero mean and covariance Q_t. As mentioned in section 4.1.1, map segments m_i are characterized by their curvature c_i rather than their radius r_i, simply because it is easier to implement using a curvature of zero for straight lines than an infinite number as its radius.

Similar to the derivation of equation 3.16 in section 3.2, the offline estimation of the mobile sensor unit's states at all times $t \in \{0, .., T\}$ means to find the most likely states given all sensor measurements and the map, i.e. the maximum in equation 4.3.

$$
\begin{aligned}
&p\left(x_{0:T}|z_{0:T}, m\right) \qquad (4.3) \\
&= \frac{p\left(z_{0:T}|x_{0:T}, m\right) \cdot p\left(x_{0:T}|m\right)}{p\left(z_{0:T}|m\right)} \\
&= \eta' \cdot p\left(z_{0:T}|x_{0:T}, m\right) \cdot p\left(x_{0:T}|m\right) \\
&= \eta' \cdot p\left(z_{0:T}|x_{0:T}, m\right) \cdot p\left(x_0|m\right) \cdot \prod_{i=1}^{T} p\left(x_i|x_{i-1}, m\right)
\end{aligned}
$$

Let a hypothesis $\mathcal{H}$ be the tuple $(x_0, \ddot{y}_{1:T})$. No a priori knowledge about the unit's movement is available. To simplify the problem and its computational complexity, all hypotheses are assumed to be equally probable, as long as no measurements are available. Equivalently, both the initial state x_0 and all $\ddot{y}_t$ shall be independent and uniformly distributed, so that the states $x_{0:T}(\mathcal{H})$ resulting from the application of the noise-free process model, i.e. the motion equations, are equally probable a-priori.

This simplifies equation 4.3 to

$$p\left(\mathcal{H}|z_{0:T}, m\right) = p\left(x_{0:T}(\mathcal{H})|z_{0:T}, m\right) = \eta'' \cdot p\left(z_{0:T}|x_{0:T}(\mathcal{H}), m\right). \tag{4.4}$$

Note that this assumption is generally not true and neglects knowledge about regularities in the mobile sensor unit's motion that are commonly available in real applications. At the same time the amount of data gathered even in short test scenarios quickly exceeds the limit where an optimal solution can still be computed efficiently. Both problems are addressed in the next section.

4.1.2.1 Complexity Reduction for Efficient Optimization

The approach presented here represents both the reduction of computational complexity and a way to introduce a priori knowledge about likely state sequences. Conveyor tracks typically consist of segments that are run with approximately constant speed and short periods of constant acceleration or deceleration, either due to speed changes between map segments or due to irregularities causing the unit to temporarily jam or bump into a wall.

In case of state estimation as an offline problem, it is beneficial to identify such segments of approximately constant acceleration. This is done in a preprocessing step based on the gathered sensor data. As can be seen in equation 4.2, changes in sensor readings are mainly caused by variations in $\ddot{y}_t$, by r_i due to transition from one map segment i to another one, and by measurement noise δ_t. Changes caused by increasing velocity $\dot{y}_t$ are comparatively small for high sampling rates. Low-pass filtering of z_t and subsequent derivation results in a signal representing the change caused mainly by changes in $\ddot{y}_t$ or r_i. Thresholding and local non-maxima suppression allows to identify points t_j, $j \in \{1, .., N\}$ of maximal change in the filtered sensor readings. Intervals $\mathcal{S}_j = [t_{j-1}, t_j]$ between those points consequently contain only insignificant change and can be assumed to correspond to segments of constant acceleration $\ddot{y}_t$. Note that this does not represent an attempt to identify transitions between map segments, even if those transitions often introduce additional fragment boundaries. Additional fragmentation of intervals, which already corresponded to segments of constant acceleration, obviously do not violate the constant acceleration criteria. On the other hand, map segment transitions may also cause periods of acceleration change depending on the segments' properties. While these multiple segmentations might be a problem for any heuristic for map segment transition finding, this only introduces a beneficial increase in resolution at regions of high interest where the constant acceleration assumption might introduce approximation error.

For intervals $\mathcal{S}_j$ with $j \in \{1, .., N+1\}$ and $t_0 = 0$ and $t_{N+1} = T$ the hypothesis $\mathcal{H}$ reduces to the more compact form of $(x_0, \overline{\ddot{y}}_{1:N+1})$ with the constant accelerations $\overline{\ddot{y}}_j$ on each segment $\mathcal{S}_j$. This allows the integration of a priori knowledge about the conveyor track's regularities while still resulting in equation 4.4 when assuming the $\overline{\ddot{y}}_j$ to be independent and uniformly distributed. The advantage is in the restriction of the search space to a more likely subset of hypotheses instead of explicitly computing the probability of each state in the sequence with full resolution as in equation 4.3.

The hypothesis $\mathcal{H}_{opt}$, which maximizes the probability in equation 4.4, can now be found by minimizing the (squared) difference between the measured sensor values $z_{0:T}$ and the expected sensor readings $\widehat{z}_{0:T}$ for a given hypothesis:

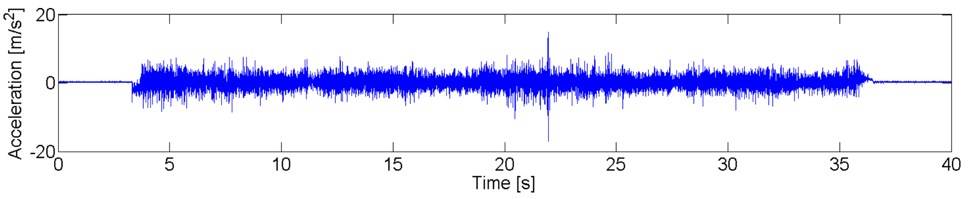
$$\begin{aligned}\mathcal{H}_{opt} &= \arg\min_{\mathcal{H}} \sum_{t=0}^{T} \left[z_t - \widehat{z}_t(\mathcal{H})\right]^2 \\ &= \arg\min_{\mathcal{H}} \sum_{t=0}^{T} \left[z_t - h\left(x_t(\mathcal{H}), m, 0\right)\right]^2 \end{aligned} \tag{4.5}$$

The optimization problem posed in equation 4.5 can be solved using any kind of non-linear optimization technique. If required, it is possible to generate an initial hypothesis by using the average sensor reading in forward direction. If the resulting dead reckoning state estimate diverges too much to allow finding the global optimum, it is possible to iteratively generate a solution using only a small subset of sequent segments at a time. Note that for unknown or imperfect sensor calibration, the rotation matrix Ψ from equation 4.2 and also potential sensor specific gains or offsets may be included into the hypothesis $\mathcal{H}$ and therefore into the estimation process. The resulting estimation algorithm is robust to sensor noise, errors the sensor calibration or deviations in the exact map specification.

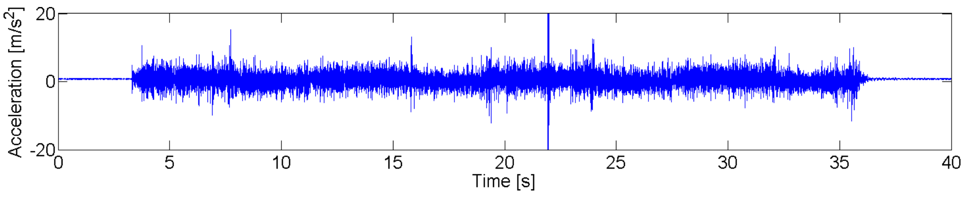
4.1.2.2 Evaluation

The offline solution for the position tracking task is tested in an experimental setup as described in figures 4.1 and 4.2. The measurements show significant random noise as well as systematic errors due to imperfect sensor calibration. Furthermore, the map specification given in table 4.1 is only approximate. All this is done deliberately to prove the algorithm's robustness in real world scenarios. Its performance under those circumstances is shown and compared to a dead reckoning approach.

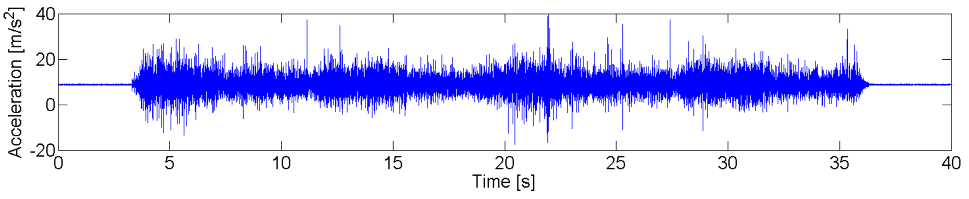
The raw sensor measurements gathered by the mobile sensor unit's acceleration sensor with 1 kHz in a 40 sec test run are visualized in figure 4.3, where axis 1 points in direction of the track, axis 2 is horizontally perpendicular to axis 1, and axis 3 is vertical, forming a right hand coordinate system. Even while the measurement unit is stationary, the sensor noise is significant. But when being transported along the track, it can be seen that the sensor noise increases to an order of magnitude above the actual signal. Also note the disturbance around the 22 second mark, which is caused manually as part of the experimental setup. Detecting this is trivial, but the central task is the estimation of the exact location corresponding to this time frame as described earlier.



(a) Raw data (axis 1).



(b) Raw data (axis 2).



(c) Raw data (axis 3).

Figure 4.3: The raw data measured during the experiment.

For this experiment the sensor is deliberately calibrated imperfectly. No rotation relative to the unit's coordinate frame is determined and no gains are applied. The sensor values' offsets are calculated from the instant of the sensor data when the unit is still stationary. Consequently the velocity inferred from this data directly is subject to significant drift as can be seen in figure 4.4. A comparison of figures 4.4 and 4.5a suggests that the sensor measures a slight acceleration when entering a curved segment and an according deceleration when leaving, but also a constant acceleration while being in the curved segment. The former can be explained by the sensor not traveling on the center of the curved segment but further outside, either due to a translated placement of the sensor with respect to the center of the mobile unit or due to a translation of the unit itself due to centrifugal force and some freedom on the broader track, both resulting in increased velocity. While this is physically plausible and expected, the additional constant acceleration is caused by measuring part of the centripetal acceleration due to imperfect calibration of the sensor's rotation. This causes part of the centrifugal acceleration to be interpreted as an acceleration along the track, thus increasing the sensor unit's estimated velocity. As mentioned above and visualized in figure 4.4 this is fatal to non-robust estimation algorithms such as dead reckoning.

Figure 4.5 presents the result of low-pass filtering the sensor data (in blue), followed by maximum detection in the derived signal as described in the previous section. The accepted maxima after applying a threshold are marked (in green) in figures 4.5c and 4.5d with positive and negative impulses of magnitude 1 while the rejected ones have a smaller magnitude. The former ones combined are also visualized as segment borders (in red) in figures 4.5a and 4.5b.

Given this segment information and a state estimate, the expected noise-free sensor readings according to the measurement model of equation 4.2 can be calculated. These expected measurements are visualized in black together with the filtered sensor data. In this experimental setup, the starting point lies close to the beginning of segment 1, i.e. on the first straight segment. This prior information is provided to the algorithm. The expected measurements according to this starting point and the dead reckoning estimate are displayed in figure 4.6.

The result after applying the optimization described above is visualized accordingly in figure 4.7. The improved correlation of sensor measurements to the map data and consequently the expected sensor data is obvious in figure 4.7b. The mobile sensor unit's stop position is also closely matched but for a small remaining velocity after the last deceleration phase causing the points labeled 12 and 13 to drift further along the track. This could be corrected by assuming the end velocity to be known, i.e. to be zero in this case, which however is not desirable for a general setup and thus not done here.

The presented test run shows the applicability of the approach under conditions of uncertainty, such as maps which are known only approximately, significant sensor noise and even systematical errors in the measurements as due to incorrect sensor calibration. However, the introduction of segments of constant acceleration, while based upon reasonable observations of the behavior of conveyor systems, still represents an approximation. While avoiding this assumption would generally be desirable, the resulting increase

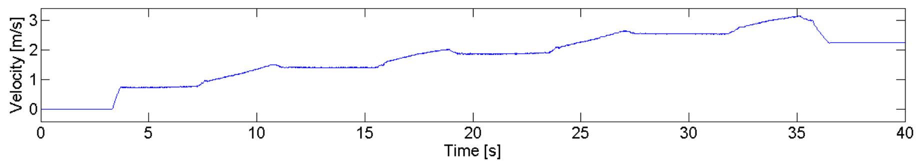
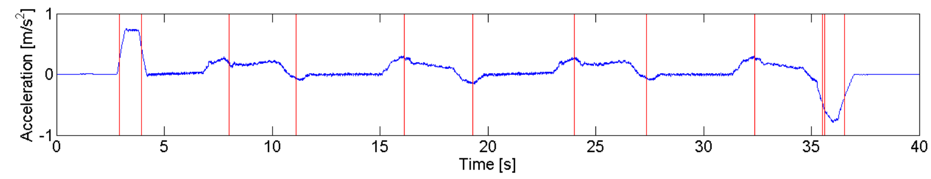
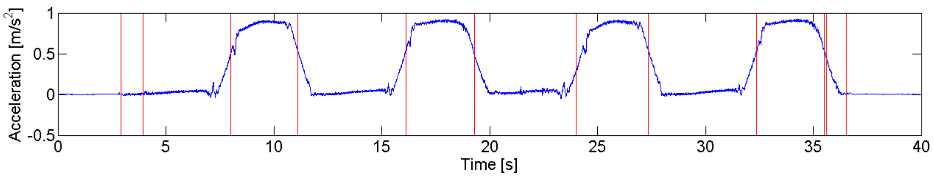


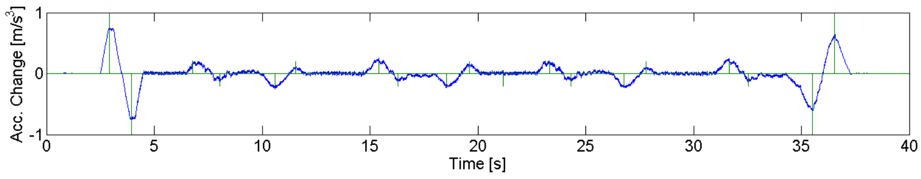
Figure 4.4: Velocities generated from raw data after offset correction.



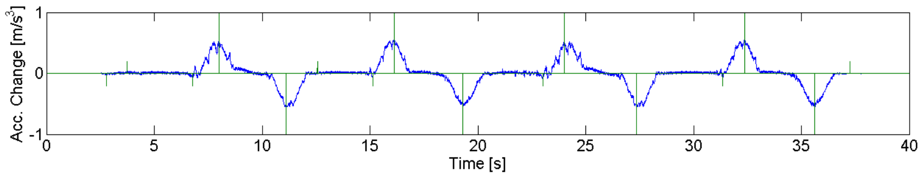
(a) Filtered data (axis 1).



(b) Filtered data (axis 2).

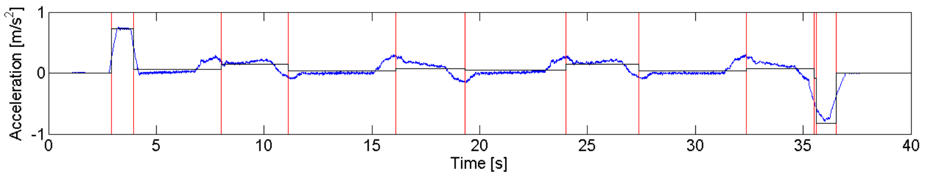


(c) Derived filtered signal with local maxima (axis 1).

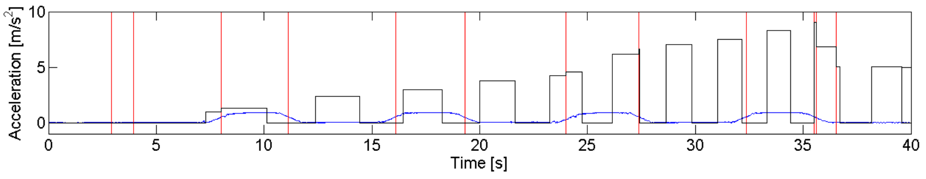


(d) Derived filtered signal with local maxima (axis 2).

Figure 4.5: The low-pass filtered sensor data with segmentation information computed in a preprocessing step.

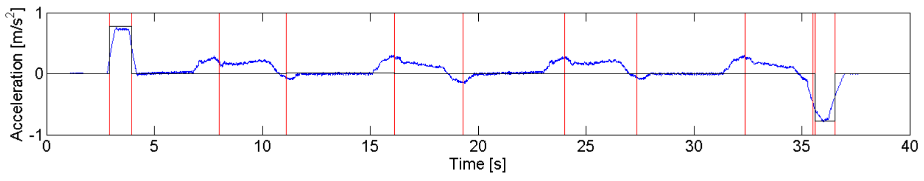


(a) Filtered data and expected measurements (axis 1).

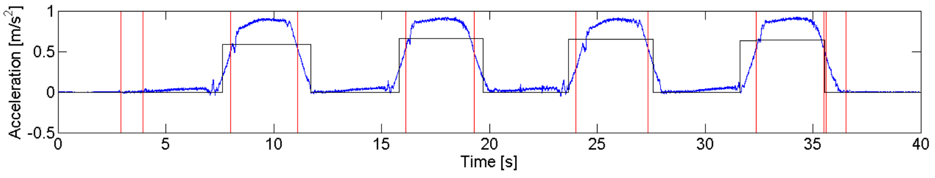


(b) Filtered data and expected measurements (axis 2).

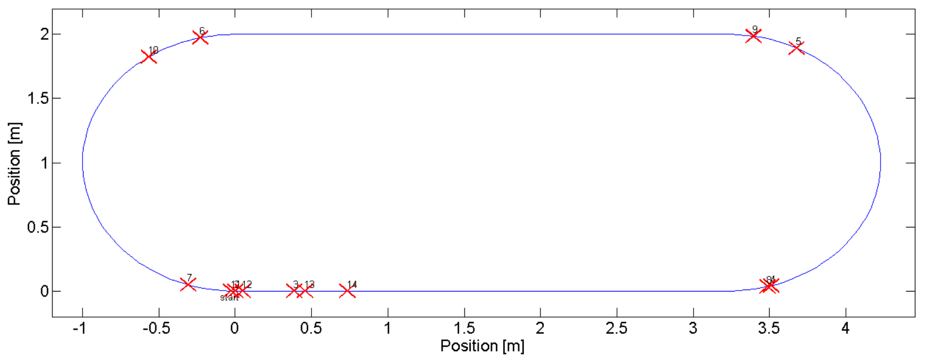
Figure 4.6: Filtered and expected sensor readings for the dead reckoning estimate.



(a) Filtered data and expected measurements (axis 1).



(b) Filtered data and expected measurements (axis 2).



(c) Map and positions of segment borders.

Figure 4.7: Filtered and expected sensor readings for the optimization result.

of complexity renders the optimization problem unsolvable in any reasonable time, even for a short test sequence like the one presented here, which only consists of 40 seconds of data. For realistic scenarios, measurement units could travel for hours on circular systems, continuously checking for potential failure areas.

4.1.3 Bayes Filter Implementation

The formulation as an online-problem for Bayes filtering is generally preferred, because such a solution's complexity does not depend on the length of the operation time and results are available directly during the time of operation. This allows a nearly continuous operation in intralogistic systems with closed configurations, and the possibility to flag potential problems as soon as they occur.

Additionally, a formulation with this simple example illustrates two relevant points: The first one, which is covered in section 4.1.3.1, is a certain degree of freedom in the interpretation of part of the input data as either control or measurement information. This can be exploited to reduce implementation complexity without having to accept any approximation. Secondly, section 4.1.3.2 illustrates differences in Kalman formulations, namely EKS versus UKF, which go beyond mere linearization accuracy as depicted in many common references (e.g. [42, 90, 43, 86]), but actually involve the utilization of additional map information. This provides both context and contrast to the analysis in section 5.1.

In the following, the scenario is slightly simplified by assuming the same roughly correct calibration of the sensor's orientation with respect to the mobile unit as in the offline solution's evaluation above. The purpose of this section is not to provide quantitative comparisons, but to illustrate conceptual choices and advantages.

4.1.3.1 Particle Filter Formulation

When applying a particle filter to this tracking problem, the first design choices are the form and dimensionality of the state space, and which sensor input to interpret as a measurement z_t and which as control information u_t. The formulation of equations 4.1 and 4.2 represent the choice of a 3-dimensional state space $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$ without the existence of any control information u_t, using the raw sensor data as z_t.

For a calibrated sensor output z_t, it is possible to simplify the mapping expressed in equation 4.2, as the rotation Ψ is an identity. Thus, $a_{1,t}$ is the acceleration component in forward direction, $a_{2,t}$ in sideways direction, and the vertical component $a_{3,t}$ of the sensor data does not carry any relevant information in the planar scenario presented above and can be neglected. In this case it is possible to interpret the sensor data in the following two ways:

1. Every input is measurement data: $z'_t = (a_{1,t}, a_{2,t})^T$. There is no control data u_t.
2. Only the sideways acceleration measurement is sensor data: $z''_t = a_{2,t}$. The forwards acceleration measurement is interpreted as odometry information: $u_t = a_{1,t}$.

In the first case, the motion model is identical to equation 4.1 and the sensor model changes only slightly to equation 4.6.

$$\begin{aligned} z'_t = \begin{pmatrix} a_{1,t} \\ a_{2,t} \end{pmatrix} &= h'(x_t, m, \delta_t) \\ &= \begin{pmatrix} \ddot{y}_t \\ \dot{y}_t^2 c_i \end{pmatrix} + \delta'_t \end{aligned} \tag{4.6}$$

In the second case, expressed in equations 4.7 and 4.8, the mobile sensor unit's acceleration is not a property of the state anymore, which reduces to $x'_t = (y_t, \dot{y}_t)$, and the motion and sensor models change accordingly.

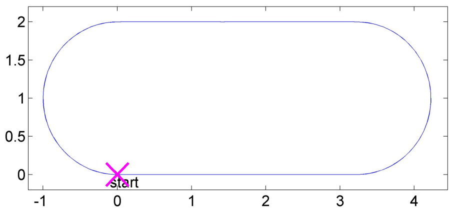
$$\begin{aligned} x'_t = \begin{pmatrix} y_t \\ \dot{y}_t \end{pmatrix} &= f'(x'_{t-1}, u_t, \epsilon'_t) \\ &= \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} x'_{t-1} + \begin{pmatrix} 0 \\ \Delta t \end{pmatrix} u_t + \begin{pmatrix} 0 \\ \Delta t \end{pmatrix} \epsilon'_t \end{aligned} \tag{4.7}$$

$$\begin{aligned} z''_t = a_{2,t} &= h''(x'_t, m, \delta_t) \\ &= \dot{y}_t^2 c_i + \delta''_t \end{aligned} \tag{4.8}$$

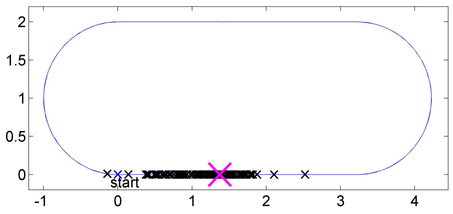
Both interpretations are valid and yield the same conceptual outcome. In fact, the offline formulation in section 4.1.2, with the hypothesis representation $\mathcal{H}$ as the tuple $(x_0, \ddot{y}_{1:T})$, can be interpreted as finding the errors corrupting the odometry information $u_{1:T}$, which would otherwise be identical to the unit's true acceleration $\ddot{y}_{1:T}$. The design difference resulting from those different interpretations has significant consequences though, which will be covered in more detail below.

Note that ϵ'_t is chosen as an acceleration noise (instead of an additive velocity noise) and multiplied with Δt in equation 4.7, so that it is in the same units and range as ϵ_t in equation 4.1. This simplifies the comparisons done later on.

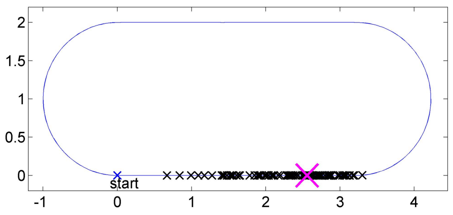
Figure 4.8 demonstrates the working of a particle filter for tracking a mobile sensor unit. Particles are depicted as black crosses, their mean by a larger purple cross. Figure 4.8b shows the spread of different hypotheses after the initial acceleration phase, each representing different positions (as visualized) as well as different velocities. The hypotheses representing



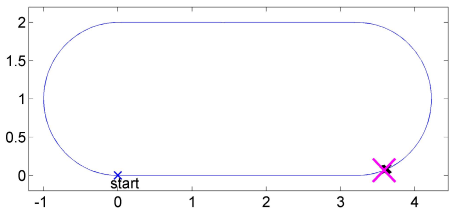
(a) Initial particle distribution.



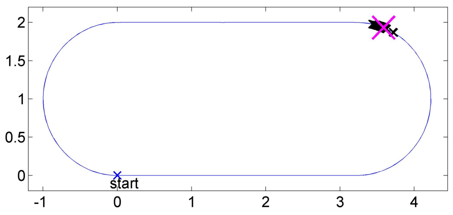
(b) Particle distribution after 5 seconds.



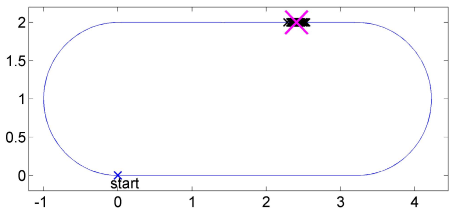
(c) Particle distribution after 6.5 seconds.



(d) Particle distribution after 8 seconds.



(e) Particle distribution after 10.5 seconds.



(f) Particle distribution after 12 seconds.

Figure 4.8: Tracking a mobile sensor unit using a particle filter. Particles are shown in black, their mean in purple.

higher velocities and therefore faster advancing positions receive decreasing weights when they enter the second map segment without measuring the correct corresponding centripetal acceleration in figure 4.8c. Similarly, the hypotheses in the remaining distribution, which is close to the true position and velocity, receive superior likelihood weights once the mobile sensor unit enters the curved segment in figure 4.8d, while the slower particles still remaining on the straight segment receive low weights, consequently collapsing the particle distribution to a small region in state space. As the curvature allows to track the velocity with higher certainty, also the position uncertainty along the curve grows only slightly (cp. figure 4.8e). Once the unit enters the next straight map segment, the particle distribution starts to expand again, albeit with a good initial guess of both position and velocity (cp. figure 4.8f).

In general, the particle filter is always able to recover as long as a single particle remains on or close to the unit's true position in state space. This is still true for slight systematic errors in the sensor data, as hypotheses with wrong velocities and resulting positions might temporarily fit the measurement information better along a single map segment, but the additional information provided by the transition from one segment to another is able to correct the estimate, as long as not all particles around the system's true state are removed during the resampling step.

As explained above, a particle filter can be designed either using $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$ or $x'_t = (y_t, \dot{y}_t)$ as its state space. Each particle can be considered as a hypothesis for a point in state space to represent the current "true" system state. Assume a set of different hypotheses is supposed to cover all combinations of 10 different positions and 10 different velocities, resulting in 100 different hypotheses. As described in section 3.4, only a particle's weight is changed by the measurement update, but not its position in state space. In order to allow sudden changes in acceleration, those changes therefore need to be introduced during the process update as *random* variations by process noise. Thus a significant amount of particles would receive a change in $\ddot{y}_t$ during the process update, which would receive low weights in the following sensor updates and ultimately be lost during the resampling in one of the following time steps. Consequently, the particle set needs to be sufficiently large to compensate the regular loss of particles due to the necessary high process noise in the acceleration component. Following the example above, an additional 10 acceleration choices per combination of position and velocity would result in 1000 particles.

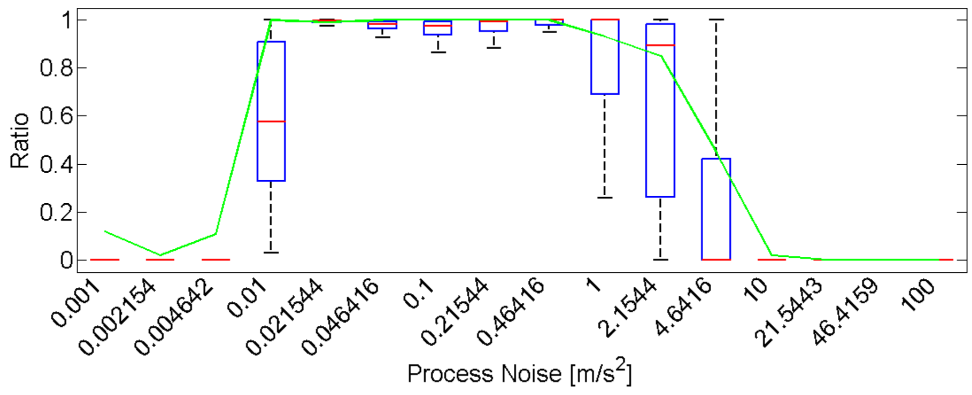
In general, designing the process model as in equation 4.7 updates the state with control data $u_t = a_{1,t}$, which can be expected to be closer to the true acceleration as a normally distributed acceleration change ϵ_t. In effect, abrupt changes in acceleration are encoded in the forward component $a_{1,t}$ of the accelerometer reading. In the absence of other information about the application's system, the process noise ϵ'_t in equation 4.7, which is necessary to compensate for the noise corruption of the odometry information, can be chosen much lower than the process noise ϵ_t, which also needs to account for true changes in acceleration. In other words, the process noise ϵ'_t in equation 4.7, which is necessary to compensate for the noise corruption of the odometry information, could be chosen lower than the process noise ϵ_t, which also needs to account for true changes in acceleration.

In this scenario however, another factor functions as a counterweight for this: The offline solution's complexity reduction presented in section 4.1.2.1 exploited the fact that the acceleration changes only at a few distinct points, and stays constant most of the time. This fact allows to parameterize the process noise ϵ_t to a low level, one actually lower than the real measurement noise $a_{1,t}$. This enables setting ϵ_t even lower than ϵ'_t, thus allowing to cover the 3rd dimension of the state space less dense, while relying on the filter's tracking functionality to iteratively adjust the estimate to eventual acceleration changes.

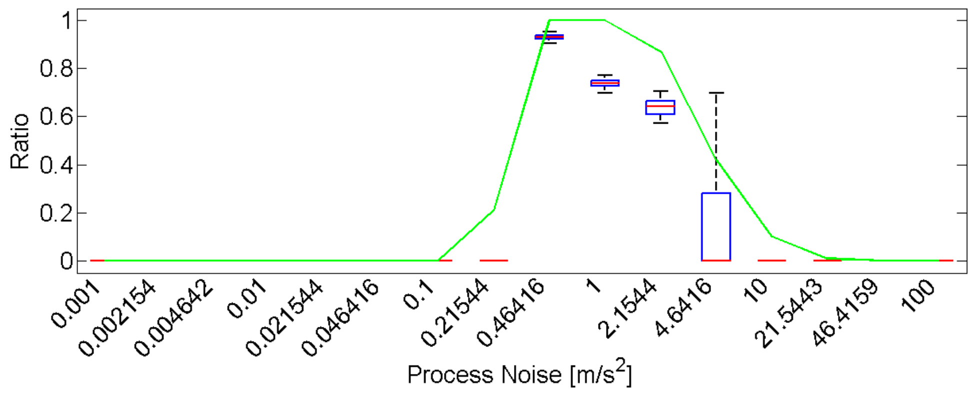
These effects with respect to the particle number and the necessary process noise values are empirically validated and visualized in figures 4.9 and 4.10. In these and the following experiments, each combination of parameters is evaluated in 100 independent particle filter computations using the recorded sensor data of a 25 m test run. The parameter range is evaluated using logarithmically spaced sample values. The ratio of particles in each test, which finish in a 1 m vicinity to the sensor unit's true final position, is recorded and visualized as box plots. The green line additionally shows the ratio of test runs per parameter combination that end with particles inside the valid region. This is done as an indication of successful and failed tracking runs, as a small number of particles close to the true position are enough to maintain these true hypotheses and continue localizing, while the tracking ultimately failed otherwise.

Figure 4.9 shows localization tries for both approaches with 1000 particles, which is a sufficiently large number, with varying process noise. It can be seen that the necessary process noise, as predicted, is significantly smaller for the 3-dimensional state space version compared to the 2-dimensional one. On the other hand, the former one also corrects the acceleration component directly in the sensor update, while the 2-dimensional particle filter only uses the lateral acceleration component as explained above. This is the reason why the 3-dimensional filter is also robust for a wider range of process noise specifications.

To evaluate the filter performance with respect to the number of used particles, a suitable standard deviation for the process noise is picked for each filter version according to the findings in figure 4.9. With the fixed process noise parameters of $\epsilon_t = 0.03 m/s^2$ and $\epsilon'_t = 0.5 m/s^2$, the filters are evaluated with varying particle numbers. As can be seen in figure 4.10, the 2-dimensional filter version can achieve comparable tracking success rates for a lower number of particles: A success rate of above 90% is possible with



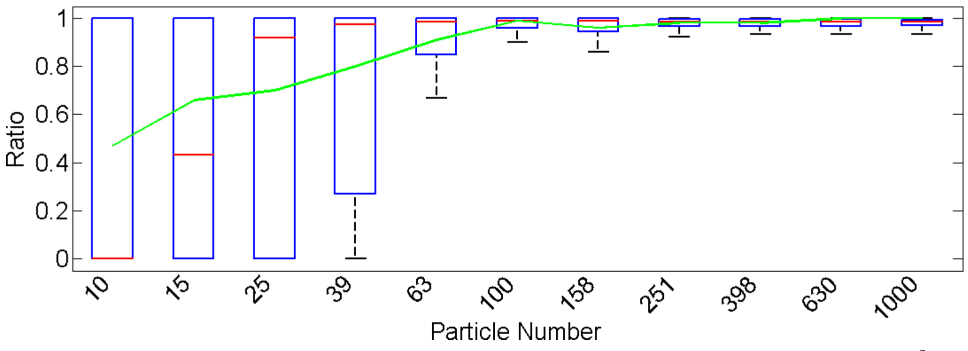
(a) Particle filter with 3-dimensional state space $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$.



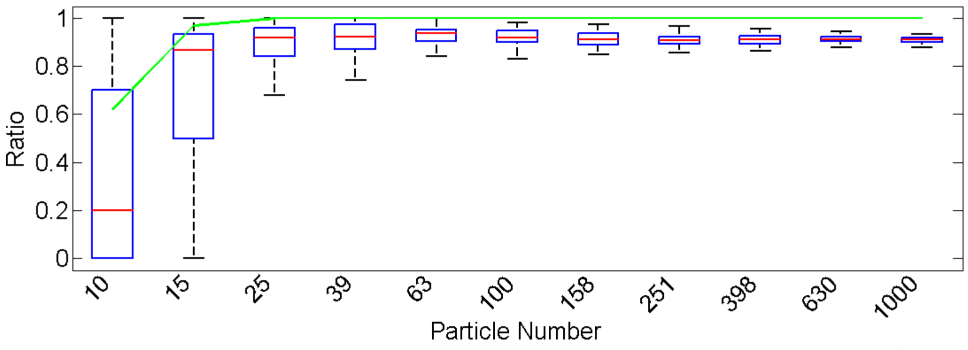
(b) Particle filter with 2-dimensional state space $x'_t = (y_t, \dot{y}_t)$.

Figure 4.9: Tracking success and ratio of particles in correct final area for filters with varying process noise and a fixed number of 1000 particles.

as little as 15 particles for the 2-dimensional filter, and with 25 particles and above the tracking is successful 100% of the time, while over 60 particles are needed to achieve 90% success for a 3-dimensional filter, and a guaranteed successful tracking requires a particle number an order of magnitude above this.



(a) Particle filter with 3-dimensional state space $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$ and $\epsilon_t = 0.03m/s^2$.



(b) Particle filter with 2-dimensional state space $x'_t = (y_t, \dot{y}_t)$ and $\epsilon'_t = 0.5m/s^2$.

Figure 4.10: Tracking success and ratio of particles in correct final area for filters with fixed process noise and varying particle numbers.

In summary, these results illustrate that even using the same filter technique, different interpretations and design choices, together with the corresponding parameterizations, have a huge impact on a filter's computational performance and robustness.

4.1.3.2 Kalman Formulation

For a Kalman filter, the difference between a dimensionality of 2 or 3 is negligible, as both run faster than real-time even in un-optimized implementations. For the sake of simplicity, the state space $x_t = (y_t, \dot{y}_t, \ddot{y}_t)$ and the process and measurement models in equations 4.1 and 4.6 are used in the following.

For the derivation of the measurement Jacobian H it is important to remember that the map segment m_i and its curvature c_i in equation 4.6 are identified using the position y_t, so the curvature can be seen as a function $c(y_t)$. This function, however, is defined section by section, so its derivative does not exist at the discontinuity of the transition point from one segment to another, and is zero otherwise.

While this is more than a hint that the design of an Extended Kalman filter for this problem is not advisable, one might argue that due to the discretization of time steps it is improbable to hit any exact border points. For all other cases, the measurement Jacobian would be given by:

$$\begin{aligned} H = \frac{\partial z'_t}{\partial x_t} &= \frac{\partial h'(x_t, m, 0)}{\partial x_t} \\ &= \begin{pmatrix} \frac{\partial a_{1,t}}{\partial y_t} & \frac{\partial a_{1,t}}{\partial \dot{y}_t} & \frac{\partial a_{1,t}}{\partial \ddot{y}_t} \\ \frac{\partial a_{2,t}}{\partial y_t} & \frac{\partial a_{2,t}}{\partial \dot{y}_t} & \frac{\partial a_{2,t}}{\partial \ddot{y}_t} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 2\dot{y}_t c_i & 0 \end{pmatrix}. \end{aligned} \tag{4.9}$$

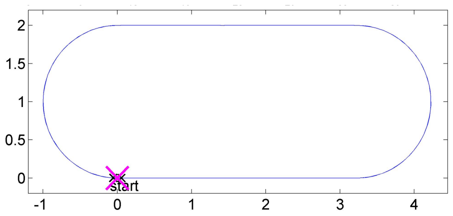
As can be seen in equation 4.9, both $\frac{\partial a_{1,t}}{\partial y_t}$ and $\frac{\partial a_{2,t}}{\partial y_t}$ are zero, so the measurements have no direct influence on the position estimate. The velocity can only be corrected on segments with a curvature $c_i \neq 0$, which also partially corrects the position component because of their dependence encoded in the covariance matrix. Most importantly, the transition point from one segment to another does not provide any special information, since the Extended Kalman filter only linearizes around the mean, and does not take the surrounding area in state space into account.

As a consequence, the estimation performance of such an Extended Kalman filter is obviously inferior to the particle filter presented above. Consider a position estimate shortly before the transition from curved to straight segment. As the centripetal force in the acceleration measurement is suddenly absent, thus lower as expected, the EKF sensor update would cause the estimate to decelerate, instead of accelerating to reach the next segment quicker. At the same time, measuring a centripetal acceleration on a straight segment would have no effect in any direction, independent of the current position either close after the last or close to the next curve, which could explain the measurement by a minor change in the position component on the track.

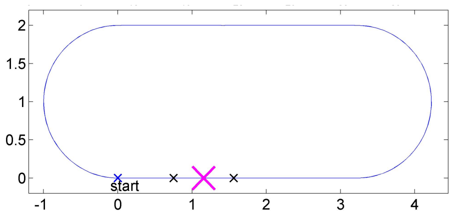
In contrast, the implementation as an Unscented Kalman filter does not need the computation of a Jacobian. Instead, sigma points are generated using the current state covariance, and the sensor model in equation 4.6 is sampled directly using these points. Consequently, the UKF behaves similarly to the particle filter, but uses a fixed number of deterministically sampled sigma points instead of a configurable number of randomly sampled particles. In case of a 3-dimensional state space as used here, the UKF uses 9 sigma points[2] generated as described in section 3.3.2. Figure 4.11

[2]Not all 9 sigma points are visible separately in figure 4.11, since they are spread

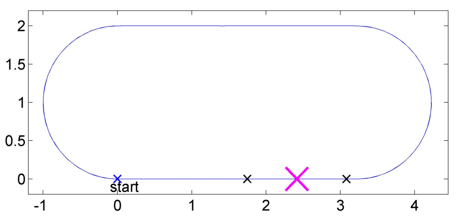
demonstrates the UKF position tracking at the same times as shown in figure 4.8. Each time sigma points are on two different segments at once, the belief is shifted to the segment which fits the measurement data best.



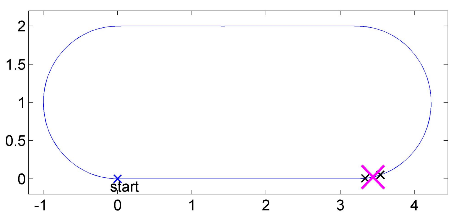
(a) Initial mean and sigma points.



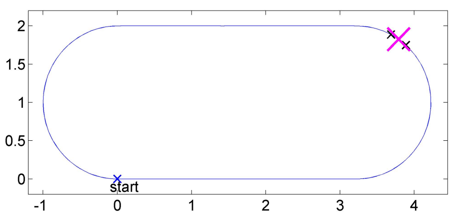
(b) Mean and sigma points after 5 seconds.



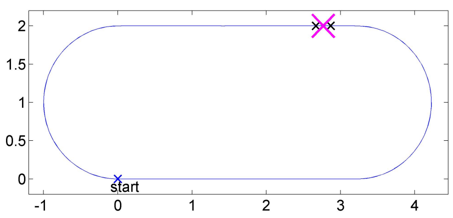
(c) Mean and sigma points after 6.5 seconds.



(d) Mean and sigma points after 8 seconds.



(e) Mean and sigma points after 10.5 seconds.



(f) Mean and sigma points after 12 seconds.

Figure 4.11: Tracking a mobile sensor unit using an Unscented Kalman filter. The current mean is shown in purple, the covariance is visualized in form of the calculated sigma points in black.

It can be concluded that the mobile sensor unit can be tracked successfully using an UKF as well as a particle filter, while the tracking usually fails using the EKF formulation with the measurement Jacobian in equation 4.9 once the sensor calibration is slightly incorrect as in all used demonstrations. The computational performance of the UKF is similar to the particle filter for the low number of used particles. However, as soon as the conveyor system becomes more complicated and includes intersections, tracking will no longer be possible using a uni-modal belief representation. In this case, the UKF would need to be extended into a multi-hypotheses tracking system,

across all 3 dimensions, and only the position component is visualized. This component is identical for several points due to the deterministic sampling along the main uncertainty directions computed by the covariance matrix decomposition.

while the necessary changes to the particle filter would be minor.

4.1.4 Considerations towards Mapping

Section 4.1.3 illustrates different ways to solve the position tacking problem online, i.e. during the sensor unit's travel across the conveyor track. This is desirable in general, especially in systems where such sensor units can be routed along cyclic paths, thus staying on track for extended periods of time while continuously reporting back to a central maintenance server. With regard to the original application purpose described in section 2.2, however, it is not only necessary to successfully track a sensor unit over extended periods of time, but to generate a precise mapping of detected disturbances and potential failure areas.

Let $t_{disturbance}$ be the time of detection of a potential failure area. The goal now is to get the best possible estimate of the sensor unit's position at $t_{disturbance}$. The trivial solution would be to simply record the belief $bel(x_{t_{disturbance}})$ or even only the single most likely position at that time. This, however, does not provide the overall best estimation of the disturbance position, since $bel(x_{t_{disturbance}})$ only incorporates past measurements, not future ones. One example would be a defective roll in the middle of a long straight segment between two adjoining curves in a roller conveyor. If the measurement unit overestimates its velocity when coming out of the curve, it would map the detected disturbance, its most likely position estimate at time $t_{disturbance}$, too far along the segment due to the aggregated position error. Both the velocity error as well as the position estimate would be corrected at a later time when nearing the next curve, but this information and the resulting correction is not yet available at the time of the disturbance.

The goal is to extend this correction at the later time backwards to the remembered position at time $t_{disturbance}$. For the UKF implementation in section 4.1.3.2 this would mean an extension of the state space to include the coordinate of the disturbance area, which would necessitate a large change in all involved components. The particle filter offers a simpler integration following the FastSLAM idea covered in section 3.5.2. As mentioned, all particles represent state hypotheses. Those are not changed during the sensor update, they just receive weights, which result in unlikely hypotheses being discarded eventually. A surviving hypothesis represents a plausible chain of motion updates which agrees with the measured sensor data. Attaching a history of measured disturbance positions to each particle will therefore allow a backtracking of the most likely path out of a later belief. In the example above, the estimation of the disturbance position would initially be identical to the particle distribution at that time, but would narrow down as soon as the particle distribution collapses at the transition to the next segment. Note that two particles can be at the same state space coordinates at this segment transition time, but might represent slightly different paths, and thus different mapped disturbance positions. As long as the disturbance is only measured once, these different mapped positions represent the remaining uncertainty of the mapping.

This application illustrates the possibility of factorizing the full simultaneous localization and mapping problem, as described previously in section 3.5. In case disturbances are measured multiple times, e.g. when repeatedly traveling along the same segment, it makes sense to employ separate estimators for each map entry, resulting in a FastSLAM implementation as presented in section 3.5.2. The likelihood of repeated observations of mapped features can then be used as part of the particle weight calculation to further improve the estimation result. Further aspects of simultaneous localization and mapping will be addressed later in chapter 8 for the dynamic environment of the robot soccer scenario.

4.2 Walking Robots

A large part of this work concentrates on humanoid robots. While its focus lies on localization and world modeling, the understanding of the specialties of humanoids compared to other robots forms a necessary basis for those. Two issues related to body design stand out: Sensor information is usually not omni-directional. Instead, exteroceptive sensors are mainly mounted in the head and can/must be moved independently from other body motion to achieve a wider field of awareness. This exteroceptive aspect will be covered in section 5.2.

An even more prominent aspect is the biped locomotion. Though motivated by nature in the hope of developing robots with navigation abilities equal to those of humans, biped walking still remains a research problem which is not yet solved completely. Even on flat terrain, current state of the art humanoid robots still experience problems when facing small or medium unplanned disturbances or contact with obstacles or other robots. Rugged terrain still presents a major challenge.

Overall, biped motion generation possesses special odometry uncertainties, which have to be taken into account by any localization algorithm designed for humanoid robots. Therefore, this section illustrates concepts and algorithms for motion generation, in order to point out inevitable uncertainties in the execution of walking motions and the ways to measure them. Section 4.2.1 presents the walking engine which is used in all experiments with Nao robots in the course of this work. This implementation is exemplary for state of the art biped walking, as an early version of it is the first closed-loop motion control implemented and employed on the Nao [13, 14], and subsequent developments by other research groups are based on the same principles [82, 31, 30]. Section 4.2.2 evaluates resulting consequences with respect to biped odometry.

4.2.1 Biped Walking Control for Humanoid Robots

[3] Gait planning for humanoid robots is fundamentally different from the path planning for simple robotic arms. The robot's center of mass is in motion all the time while the feet periodically interact with the ground in an unilateral way, meaning that there are only repulsive and no attractive forces between feet and ground. Therefore, the movement of the center of mass cannot be controlled directly, but is governed by its momentum and the eventual contact forces arising from ground interaction. These have to be carefully planned in order not to suffer from postural instability.

Section 4.2.1.1 gives a short introduction to well established criteria for evaluating robot's stability, as well as general strategies to create motions abiding such stability constraints. Section 4.2.1.2 outlines the algorithmic process and abstractions necessary to calculate the body's motion, i.e. how to move each single joint, from a general directive to move to a certain point in the environment. One part of this process, namely how to move the body's *center of mass* (CoM) to satisfy a previously derived stability criterion, will be detailed in section 4.2.1.3.

4.2.1.1 Stability of Biped Robots

Biped walking can be divided into different phases based on the feet's ground contact, as visualized in figure 4.12. The convex hull around the contact area between feet and ground is called the support polygon. A walking gait always maintains ground contact at any time. If a gait contains a certain ballistic phase without ground contact, it is called running instead of walking.

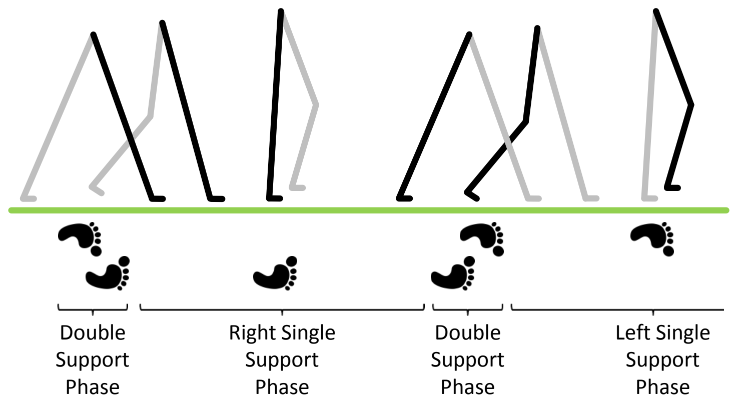


Figure 4.12: Biped gait phases.

[3]Parts of this section are based on an article in the Journal of Robotics and Autonomous Systems in 2009. The corresponding publication, see [13], is available at `http://www.sciencedirect.com`.

A robot's posture during these walking phases is called balanced and a gait is called statically stable, if the projection of the robot's center of mass to the ground lies within this support area [44]. This kind of gait however results in relatively low walking speeds. It is characterized by relatively large double support phases. Similarly, natural human gaits are normally not statically stable. Instead they typically consist of phases in which the projection of the center of mass leaves the support polygon, but in which the dynamics and the momentum of the body are used to keep the gait stable. Those gaits are called dynamically stable [44].

The concept of the zero moment point (ZMP) is useful for understanding dynamic stability and also for monitoring and controlling a walking robot [92]. The ZMP is the point on the ground where the tipping moment acting on the robot, due to gravity and inertia forces, equals zero. The tipping moment is defined as the component of the moment that is tangential to the supporting surface, i.e. the ground. The moment's component perpendicular to the ground may also cause the robot to rotate, but only in a way to change the robots direction without affecting its stability, and is therefore ignored in this context. For a stable posture, the ZMP has to be inside the support polygon. In case it leaves the polygon, the vertical reaction force necessary to keep the robot from tipping over cannot be exerted by the ground any longer, thus causing instability and fall.

In fact, following Vukobratovic's classical notation [91], the ZMP is only defined inside the support polygon. This coincides with the equivalence of this ZMP definition to the center of pressure (CoP), which naturally is not defined outside the boundaries of the robot's foot. If the ZMP is at the support polygon's edge, any additional moment would cause the robot to rotate around that edge, i.e. to tip over. Nevertheless, applying the criteria of zero tipping moment results in a point outside the support polygon in this case. Such a point has been proposed as the foot rotation indicator (FRI) point [29] or the fictitious ZMP (FZMP) [91]. In this so called fictitious case the distance to the support polygon is an indicator for the magnitude of the unbalanced moment which causes instability and therefore a useful measure for controlling the gait.

There are different approaches to generate dynamically stable walking motions for biped robots. One method is the periodical replaying of trajectories for the joint motions recorded in advance, which are then modified during the walk according to sensor measurements [49]. This strategy explicitly divides the problem into subproblems of planning and control. Another method is the realtime generation of trajectories based on the present state of the kinematic system and a given goal of the motion, where planning and control are managed in a unified system. Implementations of this approach differ in the kinematic models being used and the way the sensor feedback is handled. One group requires precise knowledge of the robot's dynamics, mass distribution and inertias of each link to generate motion patterns, mainly relying on the accuracy of the model for motion pattern generation [38, 40, 94]. A second group uses limited knowledge about a simplified model (total center of mass, total angular momentum, etc.) and relies on feedback control to achieve a stable motion [45, 95].The model used for this is often the inverted pendulum model. The latter approach is chosen here and will be detailed in the next sections.

4.2.1.2 Generating the Walking Motion Patterns

The motion generation takes place in the control program's motion process (see figure 2.4). Its input is either a target location or a velocity, as specified by the behavior module in the cognition process. Its output is a new set of target joint angles every 10 ms (see figure 2.3). These are forwarded to the Device Communication Manager in the NaoQi middleware, which in turn controls the joints using separate motor controllers, and sends feedback in form of measured joint angles and sensor information. Figure 4.13 illustrates the information flow and intermediate steps necessary for the online generation of target joint angles to achieve a stable walking motion along a desired trajectory.

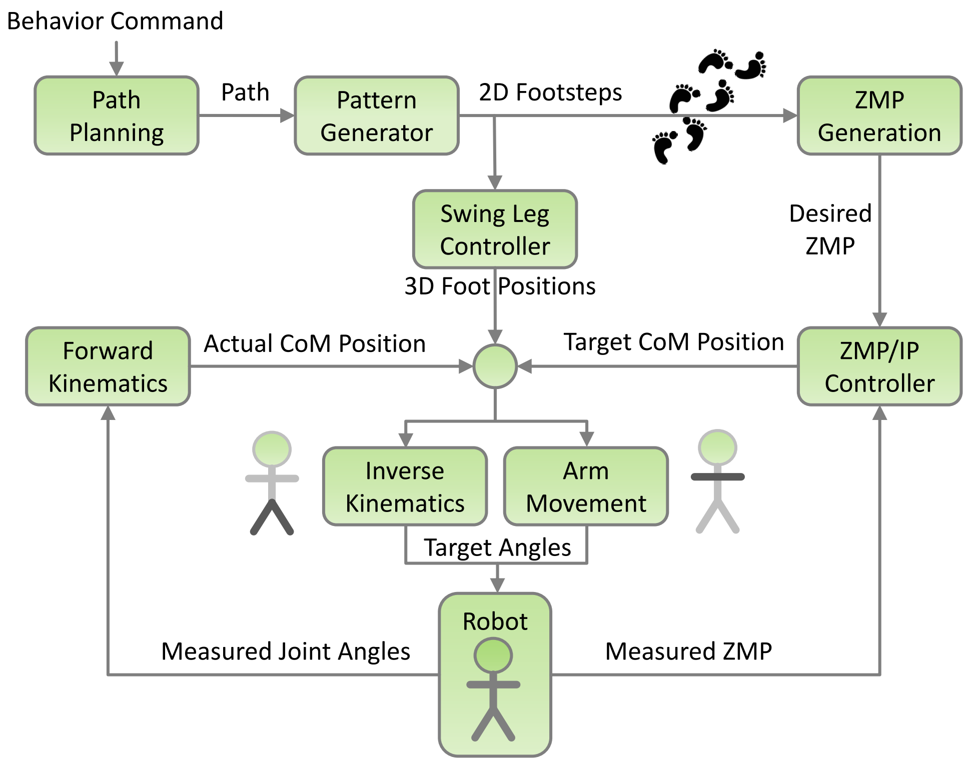


Figure 4.13: Information flow of motion generation.

As a first step, the behavior command is interpreted by a path planning module. Generating a small path segment from a velocity command is trivial, while generating a valid path to a target location also involves the navigation around other robots or obstacles. There are a number of different strategies for navigation tasks, such as the computationally intensive calculation of optimal paths in a discretized space, and efficient approximations using graph-based representations or potential-field-methods. A more detailed overview can be found in [60]. The path planning module used for evaluations in game situations is potential field based with adaptations for enhanced path stability for noisy obstacle information.

Given a desired path to follow, the pattern generator module defines a series of reachable footsteps along this path, together with the duration for travel from one step position to another. The 3-dimensional trajectory for the so called swing leg is generated by a separate module, while the information about feet on the ground provide the immediate future's support polygons. The support polygon information allows to define a ZMP trajectory, so that each motion which results in such a ZMP would be dynamically stable. The core of the walking algorithm is the ZMP/IP controller, which uses a short preview of the desired ZMP trajectory to generate a CoM motion consistent with those future ZMP positions. At the same time, current ZMP information measured by the robot, e.g. using pressure sensors in the feet, is used as control feedback to compensate disturbances like unexpected collisions or an uneven ground. The result is a CoM trajectory which allows a stable motion in the desired direction. The robot's current CoM relative to its own coordinate frame can be calculated by forward kinematics[4]. The three inputs, i.e. the desired CoM, the current offset between CoM and the own coordinate frame, and the planned 3D positions of both feet in this reference frame, are used to calculate the feet positions relative to the robot's torso. Knowing these, the inverse kinematics[5] can be solved to calculate the leg joint angles. Arm motions can be generated in parallel once the leg motions are known. In this implementation, arm motions are generated by a heuristic to counter torques originating from rapid leg accelerations. Head motions are not visualized explicitly in figure 4.13, as their interpolation can be computed directly from the behavior decision's current viewing direction request. This final set of target joint angles is then forwarded to the robot hardware, i.e. the NaoQi middleware in case of a Nao robot.

[4]Forward kinematics is the calculation of body part positions and orientations from joint angles.

[5]Inverse kinematics is the calculation of the joint angles which are necessary to reach a given end-effector position and orientation. For the existence of a (not necessarily unique) solution, the requested configuration must be inside the limbs workspace. In case of the Nao, this is not trivially done due to the combined hip joint for both legs (cp. section 2.1.2). However, when allowing one degree of freedom in the swing foot's orientation, the inverse kinematics for the Nao can even be solved analytically.

4.2.1.3 Stability Control

As mentioned in section 4.2.1.1 and indicated by the ZMP/IP controller module's name, a stable CoM motion is controlled by applying a constrained inverted pendulum model to the input ZMP trajectory. This module is a central part in the walking control pipeline described in the previous section, both for the stability of the generated walking motions and also for their precision and the handling of possible uncertainties. This section will focus on the model underlying the controller, which is relevant for this work. Further aspects of the controller's design are neglected here. They can be found in [13, 14], and in [10] concerning motions other than walking.

The main task of this module is computing the movement of the robot's body to achieve a given ZMP trajectory. To calculate this, a simplified model of the robot's dynamics is used, representing the body by its center of mass only. In the single support phase of the walk, only one foot has contact to the ground. Considering only this contact point and the center of mass, the resulting motion can be described as an inverted pendulum. Its height can be changed by contracting or extending the leg, therefore allowing further control over the CoM trajectory. Restricting the inverted pendulum such that the CoM only moves along an arbitrary defined plane results in simple linear dynamics called the 3D Linear Inverted Pendulum Mode (3D-LIPM) [45]. In a Cartesian coordinate system with the main axes y_1, y_2 and y_3, this plane is given by its normal vector $(k_{y_1}, k_{y_2}, -1)$ and its intersection with the vertical y_3-axis h_{CoM}, the height of the CoM.

For walking on an overall flat terrain the constraint plane is horizontal ($k_{y_1} = k_{y_2} = 0$) even if the ground itself is uneven. The global coordinate frame depicts the ground as the y_1-y_2-plane and the vertical direction as y_3. Let M be the mass of the pendulum, g the gravity acceleration and τ_{y_1} and τ_{y_2} the torques around y_1- and y_2-axis, then the pendulum's dynamics are given by

$$\ddot{y}_2 = \frac{g}{h_{CoM}} y_2 - \frac{1}{M h_{CoM}} \tau_{y_1} \tag{4.10}$$

$$\ddot{y}_1 = \frac{g}{h_{CoM}} y_1 + \frac{1}{M h_{CoM}} \tau_{y_2}. \tag{4.11}$$

It can be noted that even in the case of a sloped constraint plane, the same dynamics can be obtained applying certain further constraints covered in [45]. According to this model the position (p_{y_1}, p_{y_2}) of the ZMP on the floor can be easily calculated using

$$p_{y_1} = -\frac{\tau_{y_2}}{Mg} \tag{4.12}$$

$$p_{y_2} = \frac{\tau_{y_1}}{Mg}. \tag{4.13}$$

Substituting equations 4.12 and 4.13 into equations 4.10 and 4.11 yields the following:

$$p_{y_1} = y_1 - \frac{h_{CoM}}{g}\ddot{y}_1 \quad (4.14)$$

$$p_{y_2} = y_2 - \frac{h_{CoM}}{g}\ddot{y}_2. \quad (4.15)$$

It can be seen that for a constant height h_{CoM} of the constraint plane the ZMP position depends on the position and acceleration of the center of mass on this plane and the y_1- and y_2-components can be addressed separately.

It should be noted for clarification that the ZMP notion of the 3D-LIPM [45] does not take the limitation of the ZMP to an area inside the support polygon into account. Using equations 4.14 and 4.15 for planning and controlling stable walking may result in a fictitious ZMP lying outside the support polygon. As mentioned in section 4.2.1.1, this is an indicator for an unbalanced moment which causes instability. Since the mathematical notation of the 3D-LIPM does not involve any distinction based on the support polygon, the general term ZMP will be used further on.

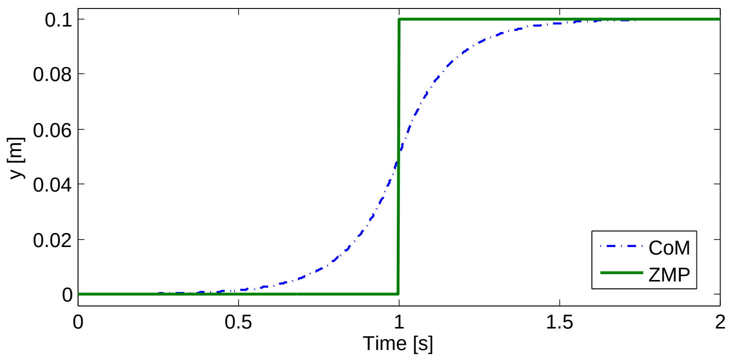


Figure 4.14: CoM motion necessary to achieve a sudden change of the ZMP location.

Consider a sideways shift of the ZMP from one foot to another, e.g. along direction y_2 without loss of generality. A movement of the center of mass, which would result in such a ZMP shift is visualized in figure 4.14. As can be seen, the CoM actually needs to start moving before the ZMP. This makes the necessity for a planning component obvious, in contrast to a controller which just has the current desired ZMP as an input value. In the ZMP/IP controller module, this is implemented using a preview controller, which gets not only the current, but also a certain set of future desired ZMP values as its input [13].

Movement of the robot's body to achieve a given ZMP trajectory is thus reduced to planning the CoM trajectory for each direction, resulting in two systems of lesser complexity whose state at every given time is naturally represented by $(y_i, \dot{y}_i, \ddot{y}_i)$ with $i \in \{1, 2\}$. The ZMP position p_{y_i} is both the target of the control algorithm as well the measurable output of the system. Equations 4.14 and 4.15 suggest that the state vector $(y_i, \dot{y}_i, p_{y_i})$ is an equivalent system representation. Choosing this one incorporates the control target into the system state and significantly simplifies further derivations of the controller.

The details of the controller's design are not of interest for this work, but can be found in [13]. Important for the understanding of biped odometry uncertainties are the underlying system assumptions, namely the 3D-LIPM, and the ZMP as the criterion for stability control. Specifically, the ZMP focuses on tipping moments to avoid falling, which is sufficient when considering stability alone. Using the walking control system described here, it is possible to compensate instabilities arising from model inaccuracies due to the difference between real robots and the abstract model assumptions, and also to recover from disturbances caused by an uneven floor or unplanned contact with other robots [14]. When considering translational and rotational step execution accuracy, however, other factors gain importance which are neglected by common walking control algorithms, including the one presented above. These factors and their consequences are detailed in the next section.

4.2.2 Biped Proprioception

The previous section illustrated the mechanisms of the walking motion generator used throughout this work. This state of the art algorithm performs well and meets its requirements to achieve high speeds and to allow quick direction changes while keeping the robot stable in dynamic situations with unforeseeable elements such as collisions or other disturbances. However, application on real robots shows that the executed motion deviates significantly from the planned walking motion. Some of the causes for these deviations are outlined in this section. Their discussion, however, stays limited to their impact on odometry uncertainty. Background and details of their derivation, findings by simulated experiments, and extended measurement methods can be found in the corresponding references [35, 9, 14].

As mentioned earlier, proprioception affects the Bayes filter both in the motion and the sensor model. In the process update this is obvious in form of the control data u_t, while in the sensor update proprioception is usually involved in the process of interpreting or mapping measurement data with respect to the robot's state. Considering an algorithm to localize a humanoid robot on a planar map as in section 2.1.1, the latter is necessary to interpret camera images and their processing results as correspondences to features in the environment's 2-dimensional map. Central to this is the coordinate transformation between the robot's local coordinate system, usually a point on the ground between its feet, and the coordinate frame of its head and therefore its cameras. Knowing this transformation allows to recover the missing depth information of image processing results by projecting perceived objects to a position on the ground. Any error in this transformation, however, directly translates into a flawed mapping of perception results, even if the image processing itself is perfectly accurate.

The reduction of the robot's kinematic structure to its center of mass, as applied in section 4.2.1, controls the dynamic motion to avoid tipping and therefore generates joint angles to walk upright without tilting, but does not explicitly plan the robot's dynamics to avoid any unwanted orientation change. On real robots, a certain oscillation and rocking of the robot's torso is therefore unavoidable. Not keeping its torso perfectly upright is not a problem in itself, but the process of measuring this orientation change is again subject to errors, which occur in the middleware's build-in orientation estimation based on inertia measurements, as well as any augmentation by kinematic calculations due to joint angle measurement errors. Appropriate handling of the resulting errors in the sensor model will be discussed further in section 5.1.

The second area in a Bayes filter, which is influenced through proprioception uncertainty, is the motion model. As described in section 3.1, the control data u_t or odometry used in the Bayes filter's motion update may originate either from the input data, i.e. the motion command issued by the behavior, which is used to generate the robot's motion, or from a measurement of the executed motion. In the first case, the naive approach would be to integrate the requested velocity. A more appropriate solution is the use of the planned foot step path in the motion generation pipeline in figure 4.13, as these foot steps already observe kinematic or dynamic constraints and provide the progress of a reference coordinate system for each given time frame.

At this point it is beneficial to further analyze the model used for motion generation in the controller. The 3D-LIPM is introduced in section 4.2.1.3 with respect to stability control, as is the ZMP in section 4.2.1.1 as a criterion for a robot posture's stability. Examining both concepts with respect to locomotion accuracy reveals several aspects, which can be neglected or compensated by sensor feedback when considering the stability of a motion, but which cause a series of executed foot steps to deviate from the planned ones.

An obvious drawback of the ZMP is its consideration of tipping moments exclusively, i.e. only those moments around rotation axes parallel to the ground are considered, as these are the moments causing a robot to tip over and fall. Reference literature about ZMP-based biped walking [91, 92, 29] mostly ignores moments around the vertical axis, as in the worst case this only causes the robot to rotate around said axis. Although not leading to instability, these rotations cause significant errors in a robot's executed orientation and translation change. Especially if the planned gait contains asymmetrical elements, such as walking in a curve or combining forwards and sideways motions, these errors easily produce systematical deviations between the robot's planned and executed gait patterns. Considering that a Nao robot has no build-in way to sense moments, forces or movement *tangential* to its feet[6], and also misses the gyroscope axis to measure vertical rotation, there is no obvious way to measure those deviations from the planned steps.

The allowance of vertical moments and the resulting rotation is only one specific aspect of the neglect of friction forces in general. The 3D-LIPM as well as the ZMP assumptions are only valid as long as the translational forces tangential to the robot's foot do not exceed the foot's friction on the ground. Exceeding the static friction on the ground causes the foot to slip backwards instead of accelerating the robot's CoM forwards, which causes the ZMP projection to move forwards while the support polygon in form of the stance foot moves backward at the same time. Since the controller introduced in section 4.2.1.3 and described in more detail in [13, 14], as well as others based on the same principles, does not explicitly incorporate frictions constraints, open parameters such as the specification of the swing foot trajectory and the gait frequency are usually tuned implicitly in a way to minimize such sliding while maximizing high possible walking velocities at the same time. As a result, sliding of a robot's feet does not occur to an extent which causes terminal instability, but small amounts can usually be compensated either by implicit gait dynamics or (to a slightly higher degree) by explicit sensor feedback. Residual sliding obviously causes further odometry errors. In this case, symmetrical sliding mostly causes errors in the translational magnitude.

Obviously, it is not necessary to rely only on information about the planned gait. Biped odometry can also be measured to a certain degree. With the hardware capability common in humanoid robots, like the Nao's as described in section 2.1.2, this process is still closely related to using the planned foot steps, as it involves the calculation of the foot positions relative to each other at the time of switching the main stance from one foot to the other. Possible heuristics for this stance foot decision are the planned stance foot switch from the motion generation module, the lower downward coordinate in the robot's reference frame based on forward kinematic calculation, or the comparison of the overall pressure on each foot in case pressure sensors are available. Compared to the usage of the planned foot steps, besides the incorporation of measurement noise, these measured steps can be more accurate, but are not necessarily so. A frequent observation is that motor asymmetries seem to imply a systematic deviation from the planned motion, and these asymmetries are then mostly compensated by ground friction forces. In this case, the measurement can actually be less accurate than using the planned motion directly. However, slight changes in

[6]The Nao's pressure sensors only measure forces *perpendicular* to the soles of its feet.

the motion execution can lead to the reversed effect that the real deviation even exceeds the measured one.

A detailed simulation of the involved motion execution inaccuracies is done in [89] for the presented walking algorithm. It suggests that the measuring process of the overall translation between both feet is a minor source of odometry error compared to foot trajectory inaccuracies prior to the stance foot switch. Such trajectory inaccuracies, which are caused by insufficiently controlled joint motors in combination with the whole robot's body motion, frequently cause premature ground contacts. These contacts in turn cause translational and rotational sliding also of the main stance foot, as well as disturbances which might cause the robot to oscillate and to become unstable. In [9], an experimental hardware extension of the Nao's feet with optical sensors is used to verify this effect and to gain insight into its magnitude. An improved odometry measurement is possible by modeling the robot's body and movement as a stochastic multi-body system with articulation constraints, as proposed in [35]. In effect, this resembles the online calculation of a stochastic physical simulation. While the motor characteristics are not modeled as detailed as in [89], the parallel execution to the real robot's motion execution allows to use a sensor fusion of joint angles and inertial measurements to correct the simulation's stochastic state. In simulated experiments, an improvement of the odometry estimation is demonstrated in [35]. Unfortunately, this method's complexity prevents its online application on current Nao hardware.

In summary, both currently available methods, using the planned as well as the measured steps, have in common that they fail to incorporate all relevant physical processes involved in the robot's locomotion. This is caused by the high degree of abstraction in the model used to generate and control the walking motion, and the current unfeasibility to use more complicated models. Furthermore, the Nao's joint characteristics even change during runtime, e.g. due to heating, thus making a "perfect" control of this robot as well as the measurement of its odometry virtually impossible, even if extremely complex models could be used to plan and control the motions in real-time. As a consequence, the described translational as well as rotational odometry errors cannot be appropriately predicted or corrected by calibration. Instead, localization algorithms for humanoid robots have to take these odometry uncertainties into account.

4.3 Conclusion

This chapter highlights several points related to the proprioception of robots and mobile devices in general, which provide a basis for different other solutions throughout this work. The application example of intralogistic tracking is used to illustrate significantly different ways of solving the same problem, the differences between the application of different Bayes filter implementations, and also different design choices for variants of the same Bayes implementation. This way, section 4.1 serves to point out possibilities and provide guidelines for the design of solutions to other problems, i.e. how different design choices favor either robustness or efficiency, or allow easier access to extend the tracking task towards mapping. As an example, the conclusion of the discussion in section 4.1.3.1 is applied to localization of smartphones in indoor scenarios in section 6.3.

The discussion of biped walking and humanoid robot odometry in section 4.2 provides a basis for the understanding of biped walking control and the involved challenges, thus in general allowing a better grasp on the robotic benchmark of robot soccer. In the context of this work it specifically serves two different purposes: On the one hand, the understanding of the generation of stable walking motions can be exploited to infer odometry from walking systems, for which the process of motion control is inaccessible, namely human users carrying smartphones in section 6.1. On the other hand, insights to biped odometry and characteristics of the involved uncertainties are a main reason for design choices in chapters 7 and 8.

Chapter 5

Exteroception

Exteroception summarizes all means of gathering information about the outside world. This involves all outward oriented sensors, mainly acoustic and visual senses for human beings, and also ultrasonic or laser-based ones for robotic distance measurements. This chapter's purpose is to highlight several exteroceptive aspects important for the best possible usage of a given sensor. A general introduction to robotic sensors can be found in [6] and [86], as well as good overviews about different sensor types and about the modeling of sensors in the context of stochastic estimation processes.

Camera-based localization for a humanoid robot, as detailed in section 2.1, is chosen as an example application to illustrated several means to utilize a given sensory setting to its full potential. Section 5.1 evaluates different sensor model design choices with respect to their influence on the system's estimation quality and points out simple yet effective solutions. Section 5.2 explores the possibilities of active sensing, i.e. the feedback of a localization algorithm's current estimation uncertainty into the sensor's control module to optimize the perception process along with the estimation process.

Thus, this chapter highlights the benefit of investing into the analysis and design of a robot's exteroception processes. It progresses from applying the most appropriate sensor model to a given perception output to actually controlling the perception process' attention itself in an optimal way.

5.1 Practical Considerations on Sensor Model Design Choices

[1] This section focuses on Kalman filters, which have been applied in many tasks and are covered extensively in literature. Designing a Kalman filter for

[1]Parts of this section are based on an article appearing in the Proceedings of RoboCup 2012: Robot Soccer World Cup XVI. The corresponding publication, see [84], will be available at `http://www.springerlink.com`.

localization is therefore not a significant challenge. However, this filter will often not perform to its full potential. The implementation process presents several design choices, some of which are discussed frequently, while others are generally neglected or only mentioned briefly. Specialized references such as [43] as well as the most standard books [86, 78] leave the impression that the most important decision is whether to address the system's non-linearity by Taylor series expansion such as in the Extended Kalman filter or by use of the Unscented transform as in the Unscented Kalman filter (cp. section 3.3).

As explained in section 3.3.2, the UKF is often showcased as superior to the EKF in terms of estimation quality. In fact, as exemplified in section 4.1.3.2, there are applications for which applying an UKF even provides conceptual advantages compared to an EKF, and other applications, as presented in [42, 43], for which the UKF's superiority is a result of its handling of non-linearity. Findings like these have led to the impression that choosing an UKF instead of an EKF will be a major source of improvement in most systems, and that this will be the main design choice in developing an estimation system for any given application.

This section shows that other aspects besides the choice between EKF and UKF have much bigger effects. Some of the highlighted issues even allow easy alterations to existing localization algorithms and promise significant improvements. Section 5.1.1 discusses influences of representing measurements in different coordinate systems, and section 5.1.2 covers measurements which are made simultaneously by the same sensor. Those aspects may seem trivial at first and their importance not obvious, but their significant influence on the outcome will be shown below. As such, it is this section's main contribution to point out simple design choices which will lead to significant localization quality improvement with minimal effort.

5.1.1 Measurement Coordinate System Choices

The effect of measurement coordinate system choices is illustrated using the localization estimation of a humanoid robot as an example application. The state to be estimated is the robot's pose $x = (p_x, p_y, p_\theta)^T$. The robot perceives point features on the ground around it, e.g. by means of processing images recorded by one or several cameras mounted in its head. Each point feature corresponds to a landmark with known global position $l = (l_x, l_y)^T$. Those expected and actual perceptions, $\overline{z}$ and z respectively, can be expressed in different coordinate systems, each of which may be used to formulate the sensor model of the Kalman filter. This is expressed in different implementations of the measurement function $h(x_t)$ in equation 3.26.

In the following, let

$$\Omega(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{5.1}$$

be the rotation around α and (l_x, l_y) the global coordinates of a known landmark l, which is part of the robot's map. The sensor model's implicit dependence on a-priori map information is made explicit here by including the landmark's location l as a parameter in $h(x_t, l)$. The time index t is omitted in all following equations for the sake of simplicity.

5.1.1.1 Measurements in Cartesian Coordinates

As the localization problem is expressed as an orientation and a position in global Cartesian coordinates, a first intuitive choice is to express a measurement on the ground around the robot in robot-centric Cartesian coordinates as shown in figure 5.1. The sensor model to calculate the expected measure-

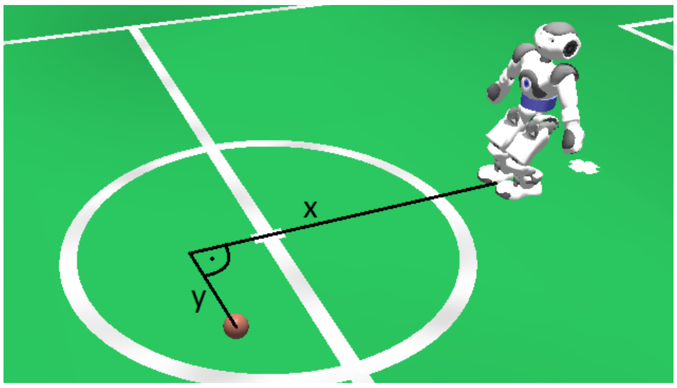


Figure 5.1: Observation given in euclidean coordinates.

ment $\overline{z} = (z_x, z_y)^T$ for the current robot pose and the correspondence to a landmark is then given in equation 5.2, and the corresponding Jacobi matrix in equation 5.3.

$$\overline{z} = \begin{pmatrix} z_x \\ z_y \end{pmatrix} = h(x,l) = \Omega(-p_\theta) \cdot \left[\begin{pmatrix} l_x \\ l_y \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \right] \tag{5.2}$$

$$\begin{aligned} H = \frac{\partial h(x,l)}{\partial x} &= \begin{pmatrix} \frac{\partial z_x}{\partial p_x} & \frac{\partial z_x}{\partial p_y} & \frac{\partial z_x}{\partial p_\theta} \\ \frac{\partial z_y}{\partial p_x} & \frac{\partial z_y}{\partial p_y} & \frac{\partial z_y}{\partial p_\theta} \end{pmatrix} \\ &= \begin{pmatrix} -\cos p_\theta & -\sin p_\theta & -(l_x - p_x)\sin p_\theta + (l_y - p_y)\cos p_\theta \\ sinp_\theta & -\cos p_\theta & -(l_x - p_x)\cos p_\theta - (l_y - p_y)\sin p_\theta \end{pmatrix} \end{aligned} \tag{5.3}$$

5.1.1.2 Measurements in Cylindrical Coordinates

Measurements can also be expressed in cylindrical coordinates, i.e. range and bearing, to indicate the distance and direction of the observed feature (cp. figure 5.2). This is often the first choice of those which are familiar with

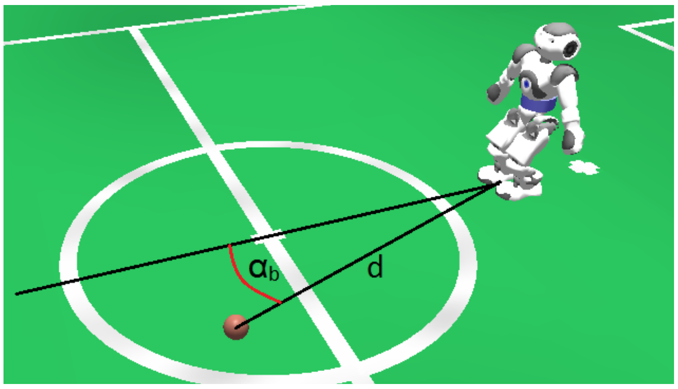


Figure 5.2: Observation given as range and bearing.

laser scanners, or of developers of robot-centric path planning algorithms. In this case, the sensor model function and Jacobi matrix are given by equations 5.4 and 5.5, respectively, with the abbreviation $d^2 = (l_x - p_x)^2 + (l_y - p_y)^2$.

$$\overline{z} = \begin{pmatrix} z_r \\ z_b \end{pmatrix} = h(x,l) = \begin{pmatrix} \sqrt{(l_x - p_x)^2 + (l_y - p_y)^2} \\ \text{atan2}(l_y - p_y, l_x - p_x) - p_\theta \end{pmatrix} \tag{5.4}$$

$$\begin{aligned} H = \frac{\partial h(x,l)}{\partial x} &= \begin{pmatrix} \frac{\partial z_r}{\partial p_x} & \frac{\partial z_r}{\partial p_y} & \frac{\partial z_r}{\partial p_\theta} \\ \frac{\partial z_b}{\partial p_x} & \frac{\partial z_b}{\partial p_y} & \frac{\partial z_b}{\partial p_\theta} \end{pmatrix} \\ &= \begin{pmatrix} (-l_x + p_x)d^{-1} & (-l_y + p_y)d^{-1} & 0 \\ (l_y - p_y)d^{-2} & (-l_x + p_x)d^{-2} & -1 \end{pmatrix} \end{aligned} \tag{5.5}$$

5.1.1.3 Measurements in Spherical Coordinates

A third coordinate system choice is given by using the vertical and horizontal angles α_1 and α_2 as indicated in figure 5.3. While the meaning of the vertical angle may not be intuitive for any direct further use, this is the coordinate system which is closest to the actual perception process in this example. With the same abbreviation of $d^2 = (l_x - p_x)^2 + (l_y - p_y)^2$ as used above and the height of the camera h_{camera}, the sensor model function and Jacobi matrix are given in equations 5.6 and 5.7.

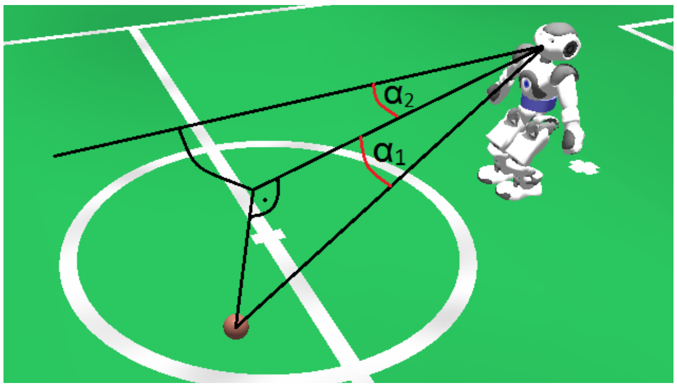


Figure 5.3: Observation given in angular coordinates.

$$
\begin{aligned}
\overline{z} &= \begin{pmatrix} z_{\alpha_1} \\ z_{\alpha_2} \end{pmatrix} = h(x, l, h_{camera}) \qquad (5.6) \\
&= \begin{pmatrix} \operatorname{atan2}(h_{camera}, \sqrt{(l_x - p_x)^2 + (l_y - p_y)^2}) \\ \operatorname{atan2}(l_y - p_y, l_x - p_x) - p_\theta \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
H &= \frac{\partial h(x,l)}{\partial x} = \begin{pmatrix} \frac{\partial z_{\alpha_1}}{\partial p_x} & \frac{\partial z_{\alpha_1}}{\partial p_y} & \frac{\partial z_{\alpha_1}}{\partial p_\theta} \\ \frac{\partial z_{\alpha_2}}{\partial p_x} & \frac{\partial z_{\alpha_2}}{\partial p_y} & \frac{\partial z_{\alpha_2}}{\partial p_\theta} \end{pmatrix} \qquad (5.7) \\
&= \begin{pmatrix} \frac{h_{camera}(l_x - p_x)}{d(h_{camera}^2 + d^2)} & \frac{h_{camera}(l_y - p_y)}{d(h_{camera}^2 + d^2)} & 0 \\ (l_y - p_y)d^{-2} & (-l_x + p_x)d^{-2} & -1 \end{pmatrix}
\end{aligned}
$$

5.1.1.4 Experimental Comparison

Two experiments are set up to compare the effects of the sensor model design choices described so far.

Simulated Perception

A simulation is set up to test the correctness of the implementation and the conformity with related work's results. Localization algorithms are used with the above mentioned different linearization and coordinate system choices and parametrized using fixed measurement covariances, which were chosen to be optimal for each approach separately. This simulation assumes a humanoid robot with noisy odometry and a perception process which measures randomly distributed landmarks with unique correspondences. This process contains errors mainly from the camera's imperfectly measured orientation due to errors in the robot head's pan and tilt joints, i.e. the errors originate from normally distributed noise in the spherical coordinate system.

As the perception process is simulated using the spherical coordinate system, it is expected that this system is the best one to be used in the localization algorithm. This is verified by the results shown in figure 5.4. Furthermore, the classical example of transforming between spherical/cylindrical and Cartesian coordinates is handled much better by the UKF than by the EKF as predicted for example by [43].

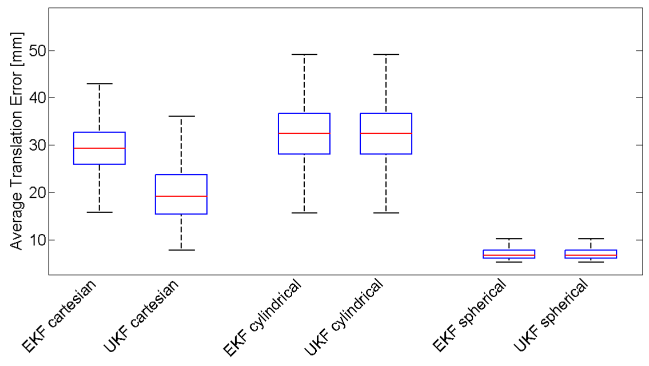


Figure 5.4: Comparison between localization quality using different linearization approaches and sensor model coordinate systems with simulated perceptions.

Real Perception Process on the Nao

To evaluate the impact on a real system, observations are recorded using a Nao. The environment is a robot soccer field as used in the Standard Platform League. Any ambiguous observations are associated with maximum likelihood correspondences based on the true robot position. The perception process also produces sporadic false positives. Those sets of observations and correspondences together with different sets of artificially generated odometry errors serve as input for all different configurations. Each generated set is processed by all approaches so that the random component in the input presents no source of bias. Note that these localization results will not diverge due to the usual problem of wrong correspondence choices once the position estimate contains a certain error, as this experiment is set up to test the sensor models, not the correctness of correspondence choices. The odometry errors contain white noise and a drift component, as this is the usual behavior of real Nao robots which are worn out or even heated up slightly asymmetrically.

An important factor for each algorithm is its parametrization. All different approaches in this experiment use the same motion update and the same process noise, which is chosen to be a certain amount above the artificially generated white noise component to compensate the drift. In common implementations, the measurement noise magnitudes are part of the robot's configuration and subject to a tuning process by the developer. Here, they are optimized separately for the approaches using a randomly picked measurement subset which is not used for the following evaluation afterwards. Thus each approach is performing with the parametrization which empirically provides the least squared localization errors.

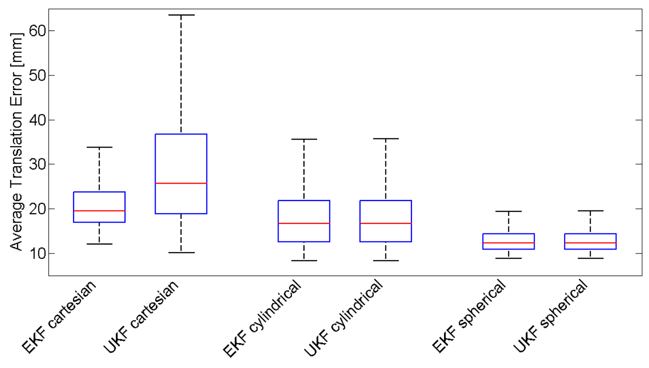


Figure 5.5: Comparison between localization quality using different linearization approaches and sensor model coordinate systems with real observations recorded on a Nao.

Figure 5.5 shows the distribution of the sums of localization errors for 1000 sets of measurements and odometry errors. It can be seen that the effect of the coordinate system choice is in general more significant than the distinction between Extended or Unscented Kalman filter. These real world results mostly verify the tendency of the assumptions in the simulated experiments, but also show discrepancies for example in the results using Cartesian coordinates. This implies that the underlying process is not fully described by assuming only normally distributed angular errors in the camera orientation. Expressing the measurement in spherical coordinates, which is intuitively the closest to the underlying process of perception, still clearly outperforms the other coordinate systems' sensor models. To use these results as a basis for development recommendations, the EKF/UKF choice is clearly second to the angular coordinate representation of the robot's measurements.

5.1.1.5 Hybrid Modifications

The empirical results above raise the question if already implemented systems, which did not use the spherical coordinate system for the sensor model design, can still make use of this information. One possibility is to adapt the measurement noise covariance matrix to better reflect the properties of the perception process, e.g. to scale the uncertainty depending on the distance of the observed feature. This has not led to any significant improvements in case of the Cartesian representation, for which more complex modifications would be necessary to reflect the spherical coordinate system's properties. The cylindrical representation, however, offers an easy improvement.

Both the cylindrical and the spherical coordinate system already share the horizontal angle; they only differ in distance against vertical angle. Applying the knowledge that errors in the distance mainly result from variations in said vertical angle, it is possible to derive an appropriate scaling factor β for the distance measurement's uncertainty.

$$z_r = \frac{h_{camera}}{\sin z_{\alpha_1}} \tag{5.8}$$

$$\frac{\partial z_r}{\partial z_{\alpha_1}} = -\frac{h_{camera}}{\sin^2 z_{\alpha_1}} \cdot \cos z_{\alpha_1} \tag{5.9}$$

$$\beta \propto \frac{h_{camera}}{\sin^2 \operatorname{atan2}(h_{camera}, z_r)} \cdot \cos \operatorname{atan2}(h_{camera}, z_r) \tag{5.10}$$

Equation 5.8 gives the relation between range observation z_r and vertical angle z_{α_1} and equation 5.9 denotes their partial derivative. Therefore, using the first rows of equations 5.6 and 5.4, the scaling factor β in equation 5.10 can be derived. Scaling the (newly tuned) expected range error with β or the corresponding entry in the measurement covariance matrix with β^2 results in the hybrid localization approach with cylindrical coordinates and distance scaled measurement covariance in figures 5.6 and 5.7. While this one presents a significant improvement over the cylindrical coordinates with constant measurement covariance and comes close to the approach in spherical coordinates, the latter one is still slightly superior[2].

5.1.2 Multiple Measurements per Time Frame

Practical Kalman implementations rarely go by the theory of one motion update and one sensor update per time step. Instead there are usually many time steps in which no observation is made, because the robot does not observe anything useful for the localization in these frames, so the sensor update is omitted. In other time steps, several observations are made at once, i.e. several different features are detected in the same time step.

[2]The median over all experiments is at 8.1 mm average translation error for the hybrid localization approach with cylindrical coordinates for the simulated experiments and at 12.9 mm for the experiments with recorded percepts, while it is at 6.6 mm and 12.2 mm for the spherical coordinates, respectively. In direct comparison of medians, averages, 25 and 75 percentiles the spherical coordinate model is superior, but not to a significant degree, as can be seen in figures 5.6 and 5.7

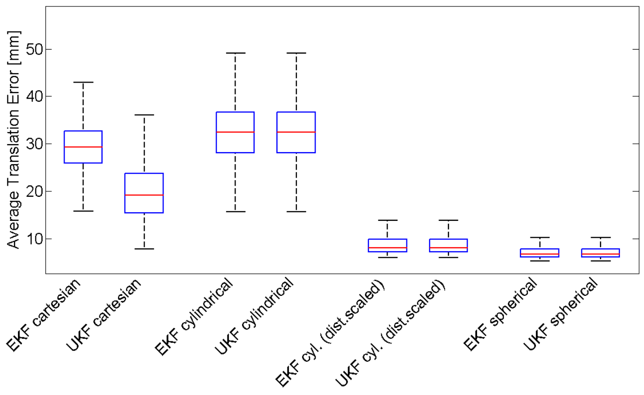


Figure 5.6: Comparison between localization quality using different linearization approaches and sensor model coordinate systems with simulated perceptions.

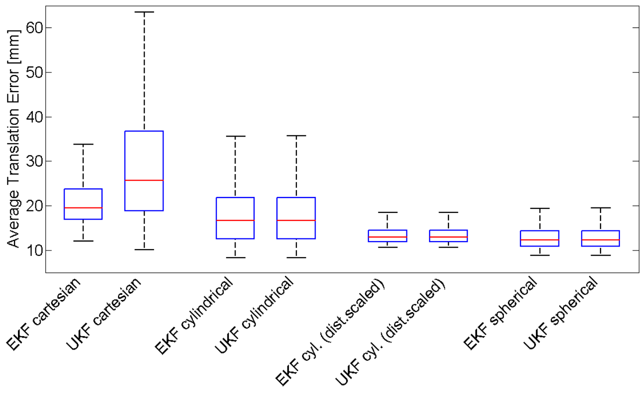


Figure 5.7: Comparison between localization quality using different linearization approaches and sensor model coordinate systems with real observations recorded on a Nao.

The common implementation is usually to execute several consecutive sensor updates. The quality of this approximation, however, depends on the perception process by which those features have been observed.

Now consider 2-dimensional feature observations as described above, each with a separate measurement covariance as in equation 5.11.

$$Q = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} \tag{5.11}$$

The stochastically correct sensor update for N_f detected features would be to execute a single $2N_f$-dimensional measurement update instead of N_f separate 2-dimensional updates. In case the different measurements are stochastically independent of each other, i.e. all off-diagonal inter-feature entries of the $2N_f \times 2N_f$ measurement covariance are zero, then a single $2N_f$-dimensional measurement update is approximated well by N_f 2-dimensional updates.

If multiple measurements originate from the same perception source and are correlated, then this simple approximation neglects potentially useful information and consequently looses in approximation quality. Taking a humanoid robot with camera based perception again as in section 5.1.1.4, multiple observations originate from the processing of a single camera image and it stands to reason that the main source of measurement error is the inaccurately estimated camera orientation due to the walking motion. Such simultaneous measurements would therefore contain nearly the same angular errors. Assuming a spherical coordinate representation for the measurements as described in section 5.1.1.3, the resulting covariance for 2 simultaneous observations is given in equation 5.12 with γ close to 1, while $\gamma = 0$ would neglect any dependence between both observations.

$$Q' = \begin{pmatrix} Q & \gamma Q \\ \gamma Q & Q \end{pmatrix} \tag{5.12}$$

Figure 5.8 shows evaluations with simulated test runs consisting exclusively of multiple observations per time step, and illustrates the differences in localization quality for iterative execution of 2-dimensional sensor updates, for $2N_f$-dimensional updates which neglect the covariance (i.e. with $\gamma = 0$), and for $2N_f$-dimensional updates with full covariances as in equation 5.12. All sensor updates in this example utilize spherical coordinate representations for the observations. As expected, the multiple 2-dimensional updates are an appropriate approximation as long as the separate measurements are independent. When observations are correlated, then significant benefits can be drawn from the information encoded in the full covariance matrix. Note that some Unscented Kalman filter implementations may become unstable for a γ too close to 1, as Q' will still be a valid covariance matrix and

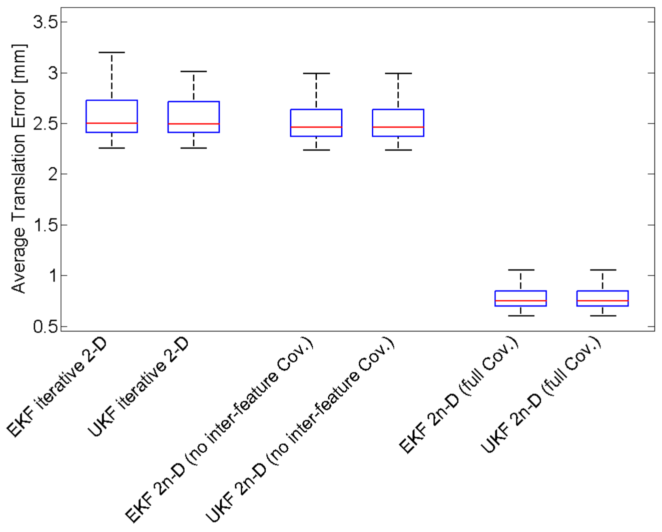


Figure 5.8: Comparison between different methods to handle multiple measurements at one time step.

therefore positive semi-definite, but very close to not being positive definite any more, which will cause the frequently used Cholesky decomposition to become numerically unstable.

Thus, together with the findings in section 5.1.1, sensor updates can be formulated to use the full information about the correlation of simultaneously observed features to exploit the perception's information content in an optimal way. In the following, context information about perceptions will be used in various ways, one of which allows to reduce even observed continuous features to sets of point correspondences, so that the results derived in this section can also be applied directly.

5.2 Active Sensing

[3] The interaction of the localization process with the control of the robot is commonly called *active localization*. Examples for control or influence of the whole navigation process are coastal navigation algorithms where the localization uncertainty is anticipated and taken into account for path planning [5]. The more complex the robot's task is, however, the more problematic to include localization quality as an additional criterion into the general planning processes for achieving the given task. A common trade-off is to allow (partial) control over the information gathering process,

[3]Parts of this section are based on an article in the Proceedings of RoboCup 2010: Robot Soccer World Cup XIV. The corresponding publication, see [11], is available at `http://www.springerlink.com`.

e.g. viewpoint and orientation of directed sensing devices, while leaving the navigation decisions unaffected. This is called *active sensing*, and in case of imaging sensors this belongs to the field of *active vision*.

Active vision in general refers to actuated camera systems with a task-oriented control of the camera movement. This might be staying on target or automatically choosing regions of interest for tracking or surveillance applications [37] or fixating on particular objects in remote collaboration scenarios [18].

Active vision for localization tasks means to move the camera with regard to the expected information gain. Therefore, it is necessary to specify such information gain, to infer which observations would improve knowledge the most, and estimate the result of different actions based on possibly multi-modal localization belief states. Previous work exists for different fields of application. The approaches depend on the range of possible actions, the types of observations and the representations of the localization belief [69, 2, 25].

This section presents the application of an approach similar to [27] to a humanoid robot with directed vision in the context of a known environment with mainly ambiguous landmarks. First, concepts are described for the measurement and estimation of localization quality to infer expected information gains. Modeling of the environment and sensors is necessary to reason about expected results of selected actions. Furthermore, the algorithm to find optimal active vision decisions is described in detail including necessary discretization and adaptations for real-time processing on platforms with restrictive resources. The presented approach differs from [27] in the underlying localization approach of a particle filter instead of a grid localization and the use of directed vision instead of an array of sonar sensors, which therefore results in different adaptations and approximations for an efficient implementation. Finally, the performance both in static and dynamic situations is evaluated.

5.2.1 Modeling Effects of Actions

The computation of an optimal active vision decision depends on the localization task and its belief representation, the actions to choose from and the estimation of their outcomes. The latter involves using knowledge about environment and sensors to first predict observations given a localization belief state, and then predict their influence on said belief state. Those predictions obviously depend on the underlying localization approach and models of environment and sensors. Figure 5.9 gives an example for the effect of a specific viewing direction decision on a particle filter's belief. The belief is represented by a spread out particle cloud, and the true pose is indicated in red. Continuing to look straight ahead produces only parallel lines and one goal post observation, which would reinforce most particles parallel to the penalty area. Adjusting the viewing direction to the right additionally includes the line crossings of the penalty area, thus collapsing the particle set to an area close to the true position. This intuitive decision process will be formalized in the following.

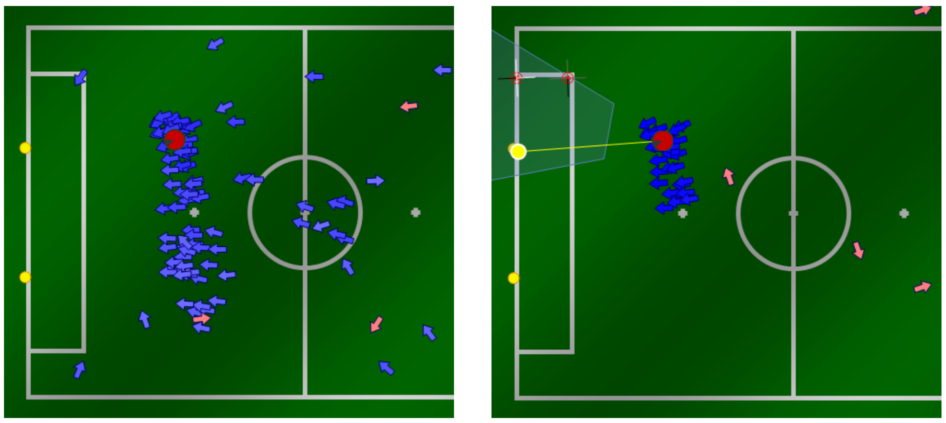

(a) Initial situation. (b) Particle distribution after head motions.

Figure 5.9: Effect of an observation on a particle filter's belief for a robot in front of a goal.

To evaluate effects of possible actions it is necessary to specify criteria for the usefulness of their result. With respect to localization, this refers to the quality of the belief state. Without knowledge about the robot's real position, it is only possible to judge the belief state's uncertainty, i.e. the covariance matrix in case of an unimodal Gaussian distribution for Kalman filters or the particle distribution. A common measure for this is the entropy.

Entropy $S(bel(x))$ is defined as the expected information content

$$S(bel(x)) = -\int p(x) \log p(x) dx. \tag{5.13}$$

with $p(x) \log p(x) = 0$ for $p(x) = 0$. This allows to compute a single scalar value as a measure for the uncertainty of a given probability distribution $p(x)$. Obviously for a distribution which is non-zero only at a single point, i.e. a Gaussian with zero variance or a Dirac delta function, the entropy $S(bel(x))$ is zero. For a uniform distribution, on the other hand, the entropy is maximal.

In case of a probability distribution given as a particle set, the analogy of this concept to thermodynamic entropy is obvious. For computing it using equation 5.13, the value of $p(x)$ can be extracted from the local particle density relative to the overall number of particles.

The common Bayes filter in section 3.2 does not include negative information, i.e. the information that a certain observation was *not* made.

Therefore the viewing direction is often omitted from the preconditions of the measurement probability, since it is only applied in the case an observation has already been made, not to compute if it *should* have been made. To compute the latter, the environment model needs to contain all necessary information to decide if a feature is visible based on current position and sensor configuration. This often includes 3D information concerning features or landmarks, whereas their simple 2D positions are normally sufficient in a map for 2D localization only. This additional information is necessary for the computation of possible observations and thus for the estimation of an optimal viewing direction. In case of directed vision, those possible observations need to be simulated for all pan and tilt angles of the camera for each possible robot position. In analogy to the process of geometric modeling, the 3D feature can be projected to the image plane to check if enough of the feature is inside the field of view for recognition.

5.2.2 Active Vision Decision Finding

Choosing an action in active vision means to choose where to point the camera. This is comparatively easy as long as the robot's (or the camera's) position is known. A naive approach would be to choose a feature from the map, e.g. the most descriptive or the closest, and to point the camera in the direction of the expected position. While this might still work as long as the belief distribution is concentrated close to the true robot pose, it becomes less effective once the distribution is less certain or even multi-modal.

The following section describes the computation of an optimal active vision decision under uncertain belief distributions.

5.2.2.1 The Optimal Action

To choose an optimal action for a given localization belief state, it is necessary to evaluate the profit of every possible candidate. For every action, the probable observations need to be computed, their influence on the belief state inferred and the result evaluated. The presented approach is based on and follows the outline of [27].

In general, possible actions might be simple motor commands to position sensors, short motion sequences or complex actions like navigation commands to move the robot to certain positions. In the following, every action u_i from the set of possible actions $U = \{u_1, u_2, \ldots, u_{N_u}\}$ will be considered atomic and consequences will only be considered for the complete execution of an action. The notation of u is identical to the control data in section 3.2 since the following holds for robot actions in general. The specific application to active vision is referred to in sections 5.2.2.2 to 5.2.2.4. The presented approach always chooses the best action based on the immediate benefit and can thus be characterized as greedy. Here, the reduction of the belief state's entropy is the robot's only goal. In the context of a humanoid robot, this is the case if the motion execution decouples head actions and other actions of torso, legs and arms, which might then be chosen separately to achieve other goals.

Section 5.2.1 illustrates the relationship between entropy and uncertainty. The optimal action u_{opt} for a given belief $bel(x_t)$ is the one minimizing the entropy $S(bel(x_{t+1})|u_{opt}, z_{t+1})$ of the belief $bel(x_{t+1})$ after execution of u_{opt} and incorporation of the observation z_{t+1} resulting from this action (see equation 5.14).

$$u_{opt} = \operatorname*{argmin}_{u \in U} S(bel(x_{t+1})|u, z_{t+1}) \tag{5.14}$$

The terminology here is that only one observation is made at any time, but this single observation might include measurements of a number of landmarks or features. This observation z_{t+1}, however, can only be predicted with certainty if the true position is known. An uncertain position given as a probability distribution only allows a decision based on the expected entropy of the next belief, as shown in equation 5.15.

$$\widehat{u}_{opt} = \operatorname*{argmin}_{u \in U} E[S(bel(x_{t+1})|u)] \tag{5.15}$$

Computing this expected entropy necessitates the simulation of all possible observations z_{t+1} for a given action. Depending on the complexity of the observation's representation and the sensing process, generating the observations can be based on recorded data as in [69] or [2]. For laser or sonar sensors this might be done using the known environment model and ray-casting operations [25] or a simple visibility simulation for cameras.

Let $p(x)$ from equation 5.13 be represented by $bel(x)$ and let $p(z_{t+1}|u)$ be the probability of observing z_{t+1} after executing u. Then the expected entropy can be calculated as follows using equation 5.13 and the belief representation (cp. equation 3.13 in section 3.2):

$$\begin{aligned}
&E[S(bel(x_{t+1})|u)] \\
&\quad = \int p(z_{t+1}|u)\, S(bel(x_{t+1})|u, z_{t+1})\, dz_{t+1} && (5.16) \\
&\quad = -\iint p(z_{t+1}|u)\, bel(x_{t+1}) \log bel(x_{t+1})\, dx_{t+1} dz_{t+1} && (5.17) \\
&\quad = -\iint p(z_{t+1}|u)\, \frac{p(z_{t+1}|x_{t+1})\, \overline{bel}(x_{t+1})}{p(z_{t+1}|u)} \\
&\qquad\qquad \log \frac{p(z_{t+1}|x_{t+1})\, \overline{bel}(x_{t+1})}{p(z_{t+1}|u)}\, dx_{t+1} dz_{t+1} && (5.18) \\
&\quad = -\iint p(z_{t+1}|x_{t+1})\, \overline{bel}(x_{t+1}) \log \frac{p(z_{t+1}|x_{t+1})\, \overline{bel}(x_{t+1})}{p(z_{t+1}|u)}\, dx_{t+1} dz_{t+1}. && (5.19)
\end{aligned}$$

By inclusion of $\overline{bel}(x_{t+1})$, possible movements of the robot are already considered. If u includes motion control, then choosing those according to equation 5.19 results in a navigation, which optimizes the robot's localization. If u includes only such commands without influence on the state x_t, then those can be controlled to optimize localization without interfering with the robot's task-oriented navigation. When u has no influence on x_{t+1}, as in active vision then $\overline{bel}(x_{t+1})$ is the same for all u and needs to be computed only once. In the case that the difference between x_t and x_{t+1} can be neglected, i.e. the robot's motion during this time is small, further simplification is possible and results in equation 5.20, which omits the calculation of the process model.

$$E[S(bel(x_{t+1})|u)] = -\iint p(z_{t+1}|x_t)\, bel(x_t) \log \frac{p(z_{t+1}|x_t)\, bel(x_t)}{p(z_{t+1}|u)}\, dx_t dz_{t+1} \tag{5.20}$$

Actions may also vary in necessary effort so that their costs need to be considered. This can be done by adding action-specific penalty terms to the minimization target. In the following, however, the unmodified expected entropy term as in equation 5.15´has been used.

5.2.2.2 Discretization

The integrals in equations 5.19 and 5.20 cannot be computed in closed form for any non-trivial applications. Discretization is necessary at certain points to allow the computation of those terms.

A first step is obvious for the application of a particle filter, in which the belief $bel(x_t)$ is approximated by a particle set $\mathcal{X}_t$. Let $\overline{\mathcal{X}}^{[i]}_{t+1} \in \overline{\mathcal{X}}_{t+1}$ be a particle from the set $\mathcal{X}_t$ after applying u, then equation 5.19 becomes

$$\begin{aligned} &E[S(\mathcal{X}_{t+1}|u)] \\ &\quad = -\int \sum_i p(z_{t+1}|\overline{\mathcal{X}}^{[i]}_{t+1})\, p(\overline{\mathcal{X}}^{[i]}_{t+1}|u) \log \frac{p(z_{t+1}|\overline{\mathcal{X}}^{[i]}_{t+1})\, p(\overline{\mathcal{X}}^{[i]}_{t+1}|u)}{p(z_{t+1}|u)}\, dz_{t+1}. \end{aligned} \tag{5.21}$$

In this case $p(\overline{\mathcal{X}}^{[i]}_{t+1}|u)$ cannot be inferred from a single particle but from the local particle density around it. A working particle localization normally tends to result in few areas with high probability after a certain time instead of a uniform distribution. A further approximation concentrates on those particle clusters, denoted as $X^{[i]}_t \in X_t$, and use average positions $\tilde{X}^{[i]}_t$ computed from these subsets instead of all particles separately, which lowers the computational cost at least by an order of magnitude and provides the needed local particle density at the same time.

Discretization of the observations cannot be motivated by the particle filter concept but depends on the sensor type and the possible observations to be made. For vision based perception those observations are commonly recognized landmarks. A complete enumeration of all possible observations would need to account for all landmarks with all possible distances and bearings which is clearly not practical. To avoid the generation of many observations whose measurement probability is near zero for the current belief, an alternative is the simulation of observations for all position hypotheses given by the particle set or the chosen position discretization. This generates a number of observations in the order of the number of landmarks times the amount of position hypotheses. Let $\mathcal{Z}_{t+1}$ be this set of possible observations and $\mathcal{Z}_{t+1}^{[j]} \in \mathcal{Z}_{t+1}$ one element thereof. This discretization step results in

$$E[S(\mathcal{X}_{t+1}|u)] \\ = -\sum_j \sum_i p(\mathcal{Z}_{t+1}^{[j]}|\overline{\mathcal{X}}_{t+1}^{[i]})\, p(\overline{\mathcal{X}}_{t+1}^{[i]}|u) \log \frac{p(\mathcal{Z}_{t+1}^{[j]}|\overline{\mathcal{X}}_{t+1}^{[i]})\, p(\overline{\mathcal{X}}_{t+1}^{[i]}|u)}{p(\mathcal{Z}_{t+1}^{[j]}|u)} \tag{5.22}$$

or

$$E[S(\mathcal{X}_{t+1}|u)] \\ = -\sum_j \sum_i p(\mathcal{Z}_{t+1}^{[j]}|\mathcal{X}_t^{[i]})\, p(\mathcal{X}_t^{[i]}) \log \frac{p(\mathcal{Z}_{t+1}^{[j]}|\mathcal{X}_t^{[i]})\, p(\mathcal{X}_t^{[i]})}{p(\mathcal{Z}_{t+1}^{[j]}|u)} \tag{5.23}$$

for the case that the robot's state change in the motion update can be neglected, e.g. if executing a head motion is fast compared to taking a step forward, so that the state change in that interval is minor.

Finally the actions to be evaluated in equation 5.15 need to be enumerated. In case of active vision these are motor commands for all possible pan and tilt angles to control the camera's field of view. Since it is not possible to process this search space in any more efficient way than to do a complete evaluation, the chosen discretization is essential for real-time computation.

5.2.2.3 Decision Computation

Based on the considerations of the previous sections, it is possible to compute the optimal action according to equation 5.15. In algorithm 2, the belief $bel(x_t)$ is given as a set of particles $\mathcal{X}_t$ and the optimal action $\widehat{u}_{opt}$ out of the set of possible actions $\mathcal{U}$ is calculated under the assumption of negligible robot motion according to equation 5.23.

Note that the expression of $|X_t^{[i]}|/|\mathcal{X}_t|$ for the term $p(\tilde{x}_t^i)$ is only valid for clusters representing equally sized volumes of state space, e.g. in case of a regular grid or similar clustering techniques, and when all particles

Algorithm 2: Active localization by entropy minimization.

```
   Input: particle set $\mathcal{X}_t$, set of possible actions $\mathcal{U}$
 1 $S_{min} = \infty$;
 2 Generate a set of clusters $X_t$ from the particle set $\mathcal{X}_t$;
 3 foreach $X_t^{[i]} \in X_t$ do // For each generated cluster:
 4     Calculate the average pose $\tilde{X}_t^{[i]}$ of all its particles $\mathcal{X}_t^{[j]} \in X_t^{[i]}$;
 5 end
 6 foreach $u \in \mathcal{U}$ do // For each possible action:
 7     foreach $X_t^{[i]} \in X_t$ do // For each cluster:
 8         Generate observation $\mathcal{Z}_{t+1}^{[k]}$ based on $\tilde{X}_t^{[i]}$ and $u$;
 9         $\mathcal{Z}_u = \mathcal{Z}_u \cup \mathcal{Z}_{t+1}^{[k]}$;
10     end
11     $S_{temp} = 0$;
12     foreach $\mathcal{Z}_{t+1}^{[k]} \in \mathcal{Z}_u$ do // For each possible observation:
13         foreach $X_t^{[i]} \in X_t$ do // For each cluster:
14             $g = p(\mathcal{Z}_{t+1}^{[k]} | \tilde{X}_t^{[i]}) \cdot |X_t^{[i]}| / |\mathcal{X}_t|$;
15             $S_{temp} = S_{temp} - g \log(g / p(\mathcal{Z}_{t+1}^{[k]} | u))$;
16         end
17     end
18     if $S_{temp} < S_{min}$ then
19         $S_{min} = S_{temp}$;
20         $u_{opt} = u$;
21     end
22 end
23 return $u_{opt}$
```

are represented exactly once. The probability $p(\mathcal{Z}_{t+1}^{[k]} | \tilde{X}_t^{[i]})$ in line 14 is the measurement probability and $p(\mathcal{Z}_{t+1}^{[k]} | u)$ from line 15 is equivalent to $1/\eta$, i.e. the inverse of the normalizing factor η in equation 3.14 in section 3.2, and can be determined by

$$p(\mathcal{Z}_{t+1}^{[k]} | u)) = \sum_i p(\mathcal{Z}_{t+1}^{[k]} | \tilde{X}_t^{[i]}) \, bel(\tilde{X}_t^{[i]}). \qquad (5.24)$$

This implies the advantage of pre-computing all probabilities and their sum before computing the entropy, which is omitted here for clarity.

5.2.2.4 Adaptations for Real-Time Processing

The use of clusters instead of separate particles, both for the generation of observations and the evaluation of their effect, already represents a first approximation. While this allows a computation that is an order of magnitude more efficient than without this approximation, several other adaptations and simplifications are possible to allow the application on robotic platforms with limited processing power such as the humanoid robot Nao.

As long as relevant observations are only caused by static elements of the environment, it is possible to pre-compute them to be stored in a lookup table for all robot states and relevant control commands. For 2D localization and pan and tilt controls for a camera this table has a dimension of 5 assuming an approximately constant height of the camera. Of course, the limited resolution in each dimension and the constant height approximation causes further loss of precision, which might influence the result. This approximation changes line 8 to a table look-up. It is labeled *Pre-calculated Observations* and evaluated in section 5.2.3.

Note that the dimensionality of 5 could possibly be reduced to 4 by summarizing the robot pose's orientation on the field and the robot's head pan into a single global pan angle. This would reduce the necessary lookup table by one dimension, but has the drawback to render considerations of self-occlusion impossible. For the Nao for example, the shoulders block a significant part of the close visible area when looking sideways. Thus, the presented approach refrains from summarizing these two dimensions for a pre-calculation of observations.

Similarly, the measurement probability might also be pre-computed and stored in a lookup table. Additionally to the 5 dimensions mentioned above, this table would need to consider the distance and bearing of the observation and in case of certain features, like line crossings on a soccer field, the feature's orientation is relevant, too. The resulting 8 dimensions would cause the table to exceed several 100 MB for useful resolutions though, so for embedded platforms with limited memory, this is no practical solution.

As can be seen in figure 5.10 for a common situation, the localization belief state often focuses on few areas of high probability. The clustering step in line 2 of algorithm 2 already causes the areas without particles to be neglected. This is valid since those areas correspond to quasi-zero probabilities. Omitting further areas with low but non-zero probability represents an approximation to equation 5.23, but considerably decreases the computational costs. This changes lines 13 to 15 to omit low-probability clusters and is labeled *Selective Computing* in the following evaluation.

5.2.3 Evaluation

The usefulness of this active vision approach to aid localization and the validity of the approximations proposed in section 5.2.2.4 are evaluated in several experiments in the RoboCup scenario described in section 2.1. Three versions of this method are compared, each with different approximations as described in the previous section:

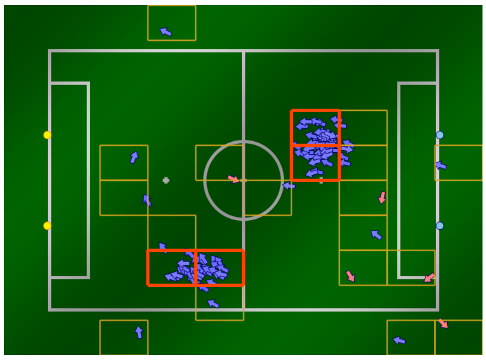


Figure 5.10: Identification of regions with high particle density.

- *Without Approximations* means the algorithm 2 without further modifications.
- For the *Pre-calculated Observations* variant, line 8 is changed to a table look-up with the according loss of precision.
- The *Selective Computing* variant, additionally to using pre-calculated observations, also omits low-probability clusters from the loop in lines 13 to 15.

The runtime necessary for algorithm 2 in these three variants is presented in table 5.1. The significant range in those measurements is a direct result of the dependency of computational complexity on localization uncertainty. If a localization belief is less certain, more positions and consequently more possible observations need to be evaluated. The timings in table 5.1 are the result of walking across the field[4] for 5 minutes.

	Maximum	Average	Minimum
Without Approximations	331.2 ms	38.5 ms	11.5 ms
Pre-calculated Observations	294.3 ms	35.6 ms	9.3 ms
Selective Computing	45.8 ms	4.2 ms	1.2 ms

Table 5.1: Runtime of active vision module with and without approximations.

The following experiments are performed in a 3D simulation for easy access to ground truth information of the robot position and the repeatability of experiments based on the exact same conditions. The environment is a

[4]The robot walks on predefined trajectories across the field, and is occasionally guided manually, since actually playing soccer is not possible with those configurations, which do not apply all approximations, i.e. the variants *Without Approximations* and *Pre-calculated Observations*.

robot soccer field as used in Standard Platform League competitions. All processes running on the real robot are also executed in simulation including motion, image processing and cognition. Active vision is implemented to decide the target viewing direction, move the head and stare in that direction for one second, then continue to choose the next direction. Since all cognition processes run at 30 Hz, the algorithm with pre-calculated observations and selective computing is suitable even for platforms with restricted resources like the Nao. However, in case of runtime peaks like the 45.8 ms maximum in table 5.1, the 30 Hz cannot be maintained and single image frames are dropped.

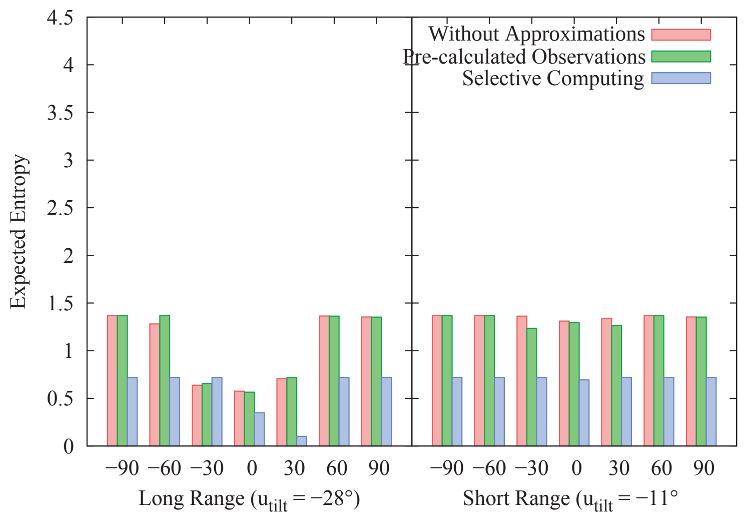
5.2.3.1 Static Situations

The discretization of possible actions used in all following experiments is 7 different pan angles and 2 tilt angles, one to point the camera at features close to the robot and one for far away observations. The choice is based on the camera's opening angles and slightly overlapping fields of view. This represents a minimal number of distinct actions. A more detailed resolution could provide better decisions, but this would imply higher computational costs.

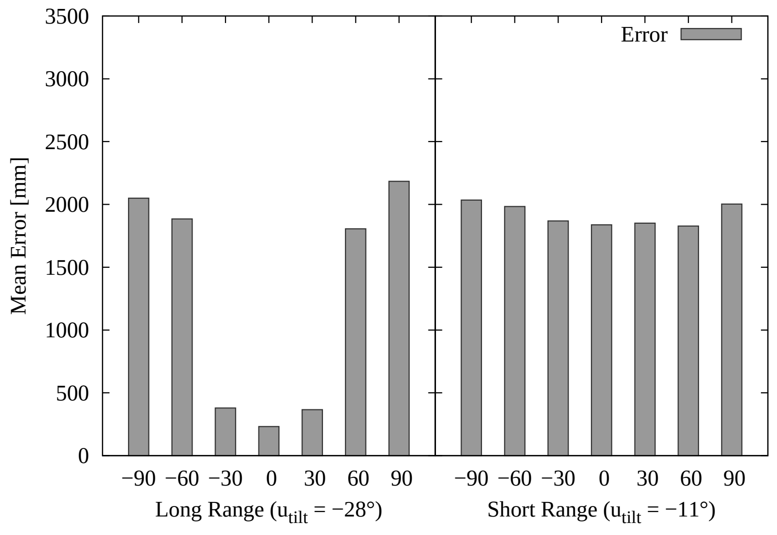
The first experimental setup shows the validity of the active vision decision process and the applied approximations. Situations are generated which allow decisions that are easy to comprehend. First a robot is placed on the field near a penalty kick mark looking at the center circle until this repetitive exclusive perception generates two symmetrical clusters of position hypotheses. This situation might occur for a robot staring at a ball lying at the kick-off position. Figure 5.11a shows the expected entropy for different action choices with and without approximations. Figure 5.11b illustrates the average position error of all particles after executing each action and processing the resulting observations. The quantitative correlation of the values is clearly visible, i.e. the lower expected entropy values in figure 5.11a can be used as an indication that lower average errors can be achieved by choosing the corresponding action, as illustrated in figure 5.11b.

A similar situation occurs frequently in soccer games: When penalized players come back into the game they are placed on the side line in the middle of the field. The side is not known to the robot in advance. In the given particle filter localization this penalty information is handled by placing 40% of the particles near each re-entry spot and distributing the rest randomly to cope with wrong referee decisions, which are not fully executed. The results for this setup are shown in figure 5.12.

A second experiment is set up to show the localization performance starting from total uncertainty in different situations focusing on a comparison between active vision guided localization and the simple approach to move

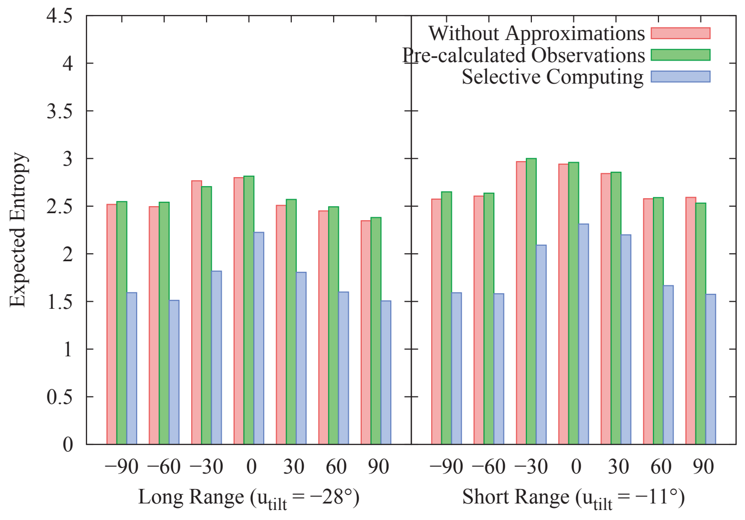


(a) Expected entropy.

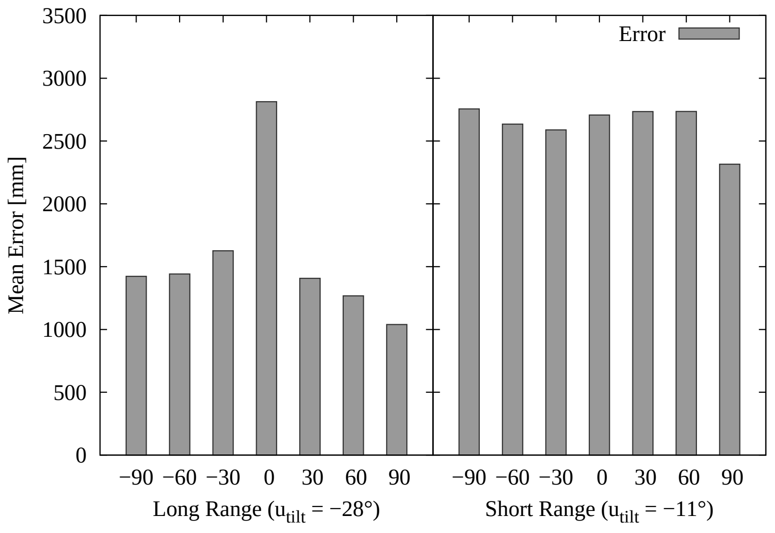


(b) Average error of particles.

Figure 5.11: The expected entropy and the real error of particles after executing an action on a penalty kick position facing the center circle.



(a) Expected entropy.



(b) Average error of particles.

Figure 5.12: The expected entropy and the real error of particles after executing an action on the removal-penalty's return position, i.e. on the side line facing the center circle.

the head from side to side to continuously scan the environment.

Figure 5.13 illustrates the course of localization errors over time for different situations for both approaches and the resulting belief state after 18 seconds. Figure 5.13e depicts an example where the active approach fails. As can be seen in all experiments the passive approach of fast scanning from side to side is more effective when uncertainty is high. This is reasonable since high uncertainty deprives the reasoning attempt of its information to make useful decisions while scanning around for several different observations resolves the uniform distribution fastest. After different distinct position hypotheses can be deduced from the belief state the active approach tends to concentrate on the features providing most information to optimally reduce the remaining uncertainty.

However, the simple passive scanning approach also yields good results in the first two situations (figures 5.13a to 5.13d), and significantly better ones in the last as analyzed above. This is explained by the setup of the first two situations, in which helpful information is found uniformly distributed all around the robot. This is not given in the third situation (figures 5.13e and 5.13f), but neither is a reasonable initial particle distribution for the active approach to base its decisions on. The following section shows the benefit of active vision in a more realistic situation.

5.2.3.2 Dynamic Situations

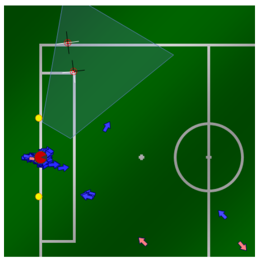
The final experiment places the robot on the sideline of the field with uncertain prior knowledge of the starting position to be on one of the side lines and the task to walk to a series of target positions (see figures 5.14 and 5.15). Prior localization knowledge is provided since random particle distributions result in random active vision decisions as shown before. In contrast to the previous experiments neither the robot itself nor its environment is static any more, as the robot walks across the field, on which other agents move around, too. The benefit of the active approach is clearly visible, especially when only few useful features are in front of the robot. In such situations, most of the scanning motion of the passive approach is wasted on featureless areas.

In summary, active vision provides the possibility to improve localization in most common situations where the localization algorithm itself already provides some result. For situations of random particle distributions, which can be detected with the same entropy criterion, a fast scanning motion should be used instead to resolve the high uncertainty by many different observations instead of a few specific ones. Most importantly, active vision is also possible and useful even for platforms with low processing power like the Nao and in environments where relevant features are distributed uniformly as is the case for a Standard Platform League soccer field.

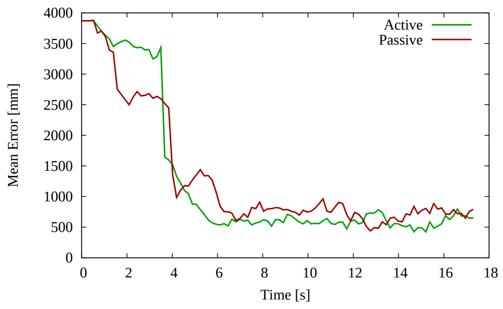
5.3 Conclusion

This chapter uses camera-based localization for humanoid robots as an application example to highlight the benefit of investing into the analysis and design of a robot's exteroception processes. This is detailed in section 5.1 by the discussion of sensor model design. This can be applied as is or in adapted form to various existing perception and estimation systems to improve the localization and tracking quality. Section 5.2 proposes to actually interfere with the perception process by actively guiding its attention to improve the underlying stochastic estimation process, and details how this can be done in an optimal and efficient way.

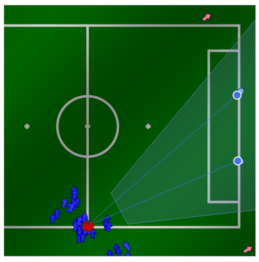
This chapter's results stand for themselves, but the general considerations about exteroception and sensor model design are also applied in chapter 6 to a different application scenario. The insight into the perception process of humanoid robots provided here also benefits the understanding of further parts of this work, specifically chapters 7 and 8.



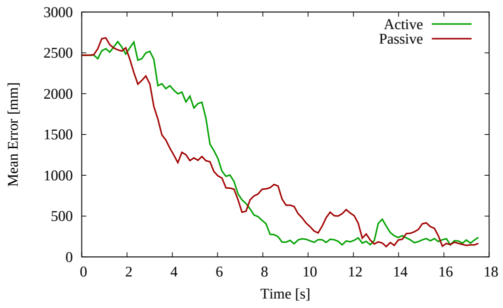
(a) Goalie using penalty area and side lines for localization.



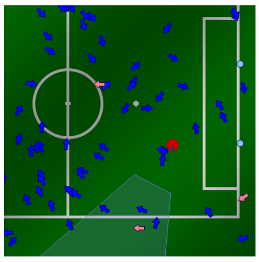
(b) Active localization needs longer to decrease uncertainty but keeps the localization error lower afterwards.



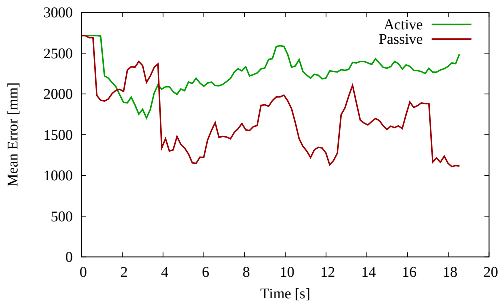
(c) Looking at the goal to resolve symmetrical hypothesis.



(d) The active version needs longer again but keeps the error low after that.



(e) This situation allows only few and ambiguous observations.



(f) A decision resulting in no useful observation does not change the particle distribution enough to change the decision in return.

Figure 5.13: Course of localization errors over time for different situations both for active and passive approach and the resulting belief state after 18 seconds.

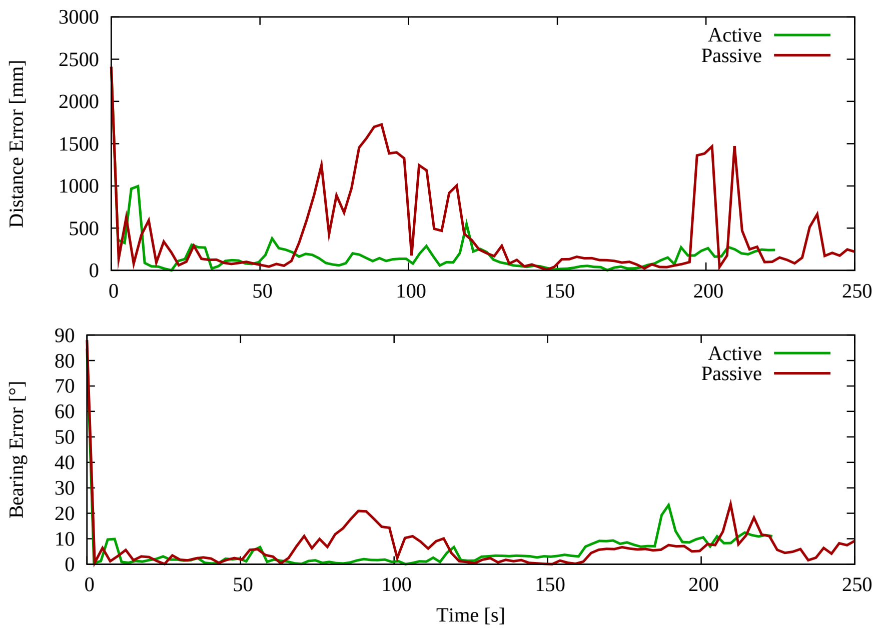
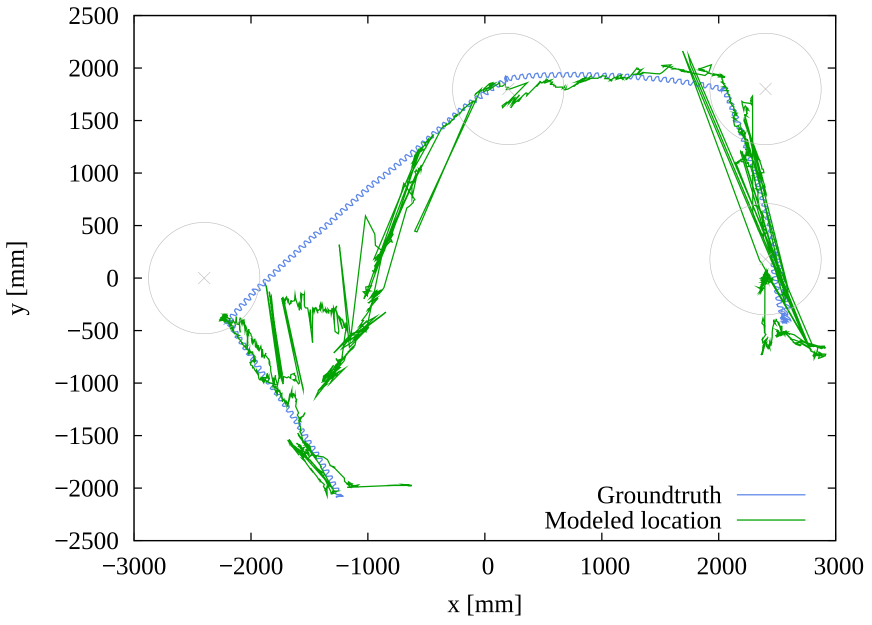
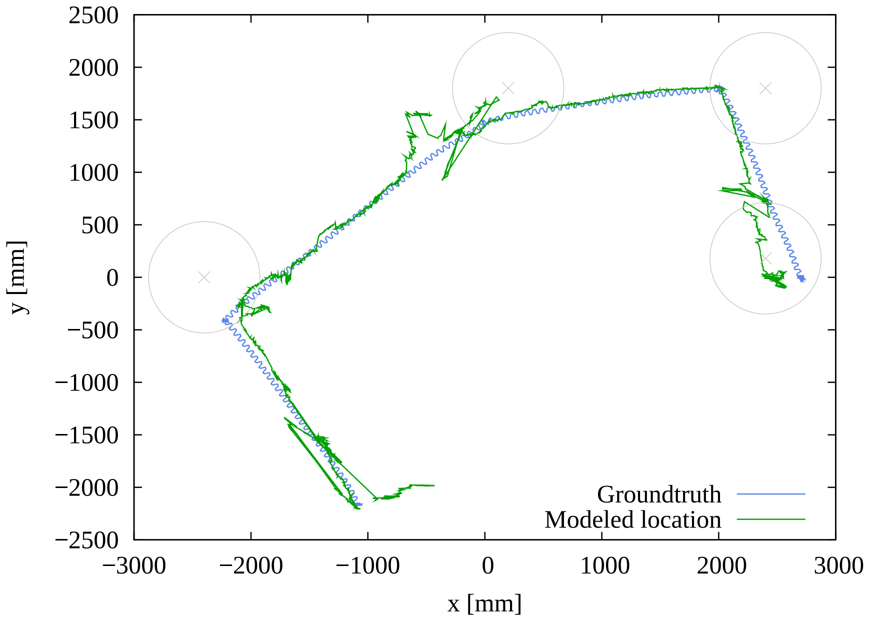


Figure 5.14: Position errors (in average 172 mm and 418 mm) and orientation errors (in average 4.97° and 6.14°) of localization while walking to target positions on the field for the active and passive approach, respectively.



(a) Passive localization.



(b) Active localization.

Figure 5.15: Localization (green) versus true position (blue) of the robot while walking to a series of target positions.

Chapter 6

Position Estimation for Ubiquitous Computing

This chapter details the development of a stochastic localization filter for the indoor smartphone positioning application scenario described in section 2.3 as a basis for indoor location-based services. In contrast to the intralogistic example in section 2.2, in which the sensory information is scarce and the environment comparably simple, ubiquitous computing usually provides a wide spectrum of sensor input from a complex environment. In such a scenario, stochastic modeling provides elaborate sensor fusion methods with the potential to achieve results superior to conventional heuristic measures as those in section 2.3.2. So far, however, such methods are uncommon in this context.

To implement a Bayes filter for localizing a smart phone, some consideration needs to be given to a suitable motion and sensor model. Designing a motion model for a smart phone might not seem intuitive, as it moves around being carried by a person, and no direct information about this person's intentional movement direction or locomotion process is available. Section 6.1 describes the computation of odometry information for smartphones based on their proprioceptive sensors and the knowledge about the walking motion generation detailed in section 4.2.

A smartphone's exteroception is mainly given in form of signal strength measurements from sources such as cell towers, Wi-Fi or Bluetooth. In reference literature as well as many applications, a number of heuristics exist relating location and received signal strength. Those however are mostly focused on providing a single "best guess" position for each new measurement, and thus are rarely stochastically sound, nor able to provide a measure for the sensor reading's likelihood for a given state. Section 6.2 covers the derivation of stochastically sound sensor models based on signal strength mapping. The same mapping mechanism is used to augment missing magnetic field information in the motion model, which is necessary for indoor scenarios.

These proprioceptive and exteroceptive components are combined into a stochastic filter system and evaluated in an office environment in section 6.3. Thus, this chapter illustrates these principles — proprioception, exteroception, and stochastic filtering — in an application, which so far has not been covered from a stochastic modeling viewpoint to this extent elsewhere.

6.1 Odometry Estimation for Mobile Handheld Devices

Designing a motion model for smartphones, or mobile handheld devices in general, presents an obvious problem: the control input for the device's motion is not known. This problem is commonly addressed by applying a high isotropic process noise to the tracking algorithm, as done in the particle filter implementation without control data in section 4.1.3.1, but in this case the result of the movement is not measurable directly.

In general, the model of a random walk, as implied by isotropic process noise, conflicts with the goal oriented movement of a smartphone's carrier. It is thus desirable to extract information about a person's movement, namely walking direction and velocity, from the smartphone's acceleration sensors. The naive approach is to apply dead reckoning based on the acceleration readings. When directly applying the measurements to inertial navigation, the walking component of the measured acceleration represents a noise signal which is far greater than the person's forward or backward acceleration itself, thus resulting in very poor navigation for carried devices. This is even true if the device is held perfectly horizontal and aligned with the walking direction, which would not be the case in a typical scenario. Any imperfectly tracked orientation change additionally corrupts a horizontal velocity estimation by overspilling from measured vertical accelerations.

The similarity of this situation to odometry estimation for humanoid robots, where inertial navigation fails due to the same reasons, is obvious, although here no information is available about the internal motion generation processes. In fact, it can be seen as its inverse problem: instead of generating body accelerations and trajectories, which lead to stable walking patterns, a normal person's walk is stable due to the same dynamics, and the belonging walking patterns need to be inferred from their measured acceleration pattern. Thus, the alternative approach presented here is to explicitly analyze the inertia measurements for the signal component originating from the walking motion and using the knowledge about human movement patterns to infer odometry information from those signals.

6.1.1 Coordinate Systems for Odometry Measurement

The sensor unit containing the accelerometer, magnetometer and gyroscope, or at least two of these components, is called inertial measurement unit (IMU). It enables the motion measurement of smartphones, usually in order to utilize this information to flip the screen layout if the smartphone's orientation changes, or simply as an input for games. While the IMU's coordinate frame is fixed to the smartphone's, it usually changes with respect to the user and to the environment. In order to properly cope with orientation and movement of the mobile device, it is necessary to consider three different coordinate frames:

- the *world coordinate frame* as the reference relative to which the mobile device needs to be localized,
- the idealized *torso coordinate frame*, which is fixed to the person's torso and aligned with the world's vertical direction (z-axis) and the forward walking direction (x-axis)
- and the *device* or *sensor coordinate frame*, in which the measurements are made.

For some smart clothing applications, the sensor coordinate frame might be fixed to the torso coordinate frame, but in general the mobile device's translation and especially rotation relative to the torso can change over time. An easy example is a device carried in a way that the person can observe and interact with the content displayed on the screen. In this case, the device's orientation will change when the screen is held in slightly different angles, rotated from landscape to portrait, or even temporarily put away for a reason unrelated to the smartphone's current application. A more challenging scenario is the odometry estimation for a smartphone carried in the trouser's pocket at the side of the leg of the walking person. In that case the device coordinate frame is actually moving relative to the torso coordinate frame with the same periodicity as the walking motion which is to be estimated. This causes additional problems when abstracting from the device coordinate frame and trying to estimate the torso coordinate frame motion relative to the world coordinate frame.

6.1.2 Understanding and Exploiting the Human Gait

Research of the human gait as well as that of motion generation for biped robots provides an understanding of the involved processes to an extent which allows the formulation of a general pattern of expected motion phases and occurring accelerations. These have already been illustrated in figure 4.12 in section 4.2.1.1 for the purpose of explaining the gait generation for humanoid robots. Figure 6.1 augments this simplified phase pattern of the human walking cycle with the expected accelerations of the torso frame relative to the world frame. While oscillations occur in all three coordinate axis, the vertical and the sideways accelerations are most obvious[1]. The vertical component is caused by a slight vertical oscillation of the person's center of mass and by the actual touchdown of the foot on the ground. Thus, its periodicity is given by the time between the two consecutive touchdowns of a foot, thus defining the step frequency. Its phase coincides with the person's CoM accelerations, albeit with the touchdown as an additional jerk. The horizontal sideways motion component is caused by changing the stance foot from left to right and back. Its periodicity is given by swinging from the left foot to the right one and back, which includes two steps, thus only half the step frequency reflected in the vertical acceleration. The forward acceleration component is less obvious and also of a smaller magnitude. Its phase and periodicity are coupled to the vertical one by the forward push of the stance foot.

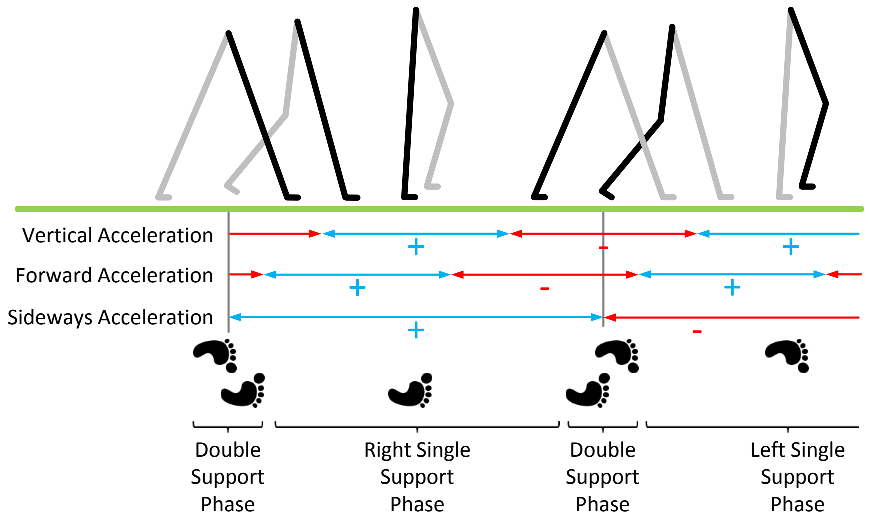


Figure 6.1: Accelerations generated by human walking.

Being able to identify those different components in an acceleration signal measured by an IMU provides most information needed for an odometry estimation, i.e. walking direction and step frequency. Also, though less directly, information about the stride length can be heuristically gained from the oscillations' amplitudes. The separation of those components, and espe-

[1]It should be noted that for controlling a humanoid robot as described in section 4.2.1, it is possible to derive control accelerations minimized along two of the axis. The presented 3D-LIPM control model explicitly avoids vertical movement in order to simplify the controller design, and it is possible to define ZMP trajectories to achieve a minimally varying forward velocity. However, even for such controls, a real physical system will still deviate from such theoretical acceleration controls and then tend to express the presented acceleration components.

cially the essential recovery of the phase shift between the vertical and the horizontal forward component, which have the same frequency, is very susceptible to orientation errors. As illustrated in figure 6.2, the first step in the proposed odometry estimation system therefore consists of a separate device orientation estimation which abstracts the acceleration signal from the device's coordinate frame. The newest generation of smartphones possesses full IMUs, including 3-axis accelerometers, magnetometers and gyroscopes, but at least the former two are available in most current devices. This allows the application of a Kalman filter to estimate the device orientation based on the gravity component present in the acceleration measurements and the expected magnetic flow vector.

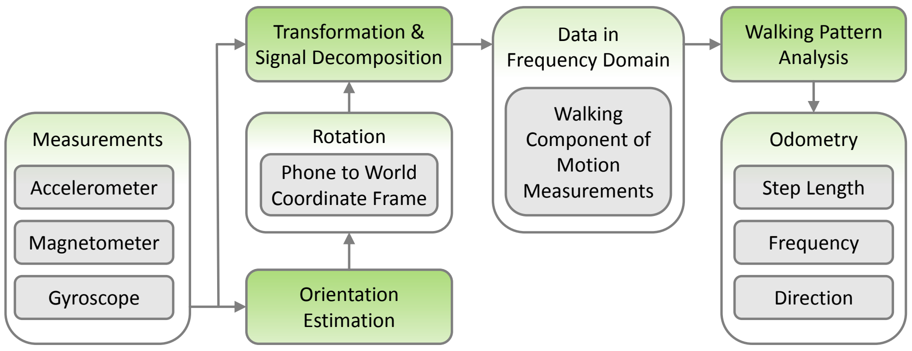


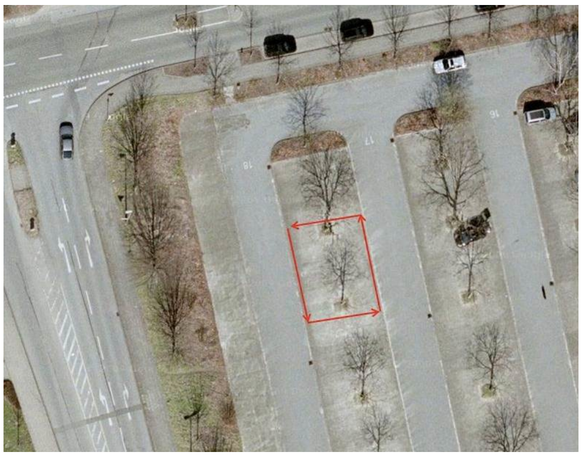
Figure 6.2: Odometry estimation from inertia measurement information.

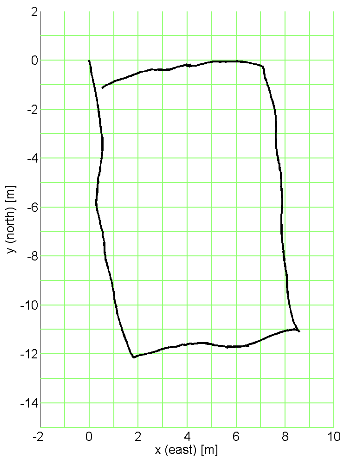
The subsequently transformed inertia measurements are decomposed using the Fourier transform. Two features in the presented implementation allow the online computation with high frame rates, while keeping the necessary processing power and therefore the mobile device's energy consumption low. The first one is the approximation of applying three 1-dimensional Fourier transformations instead of one 3-dimensional. This is sufficient in the given case, since the identification of the step frequency can be done in the vertical component, which allows to simply look up the transformed complex values corresponding to this frequency in the other two dimensions and to infer the forward direction and step length from their phase shifts and magnitudes. The sideways motion can be found the same way at half the step frequency and used together with the magnitudes of the other oscillation vectors to validate the walking motion.

The second implementation detail relevant for real-time performance is the iterative implementation of the Fourier transform instead of the more commonly used FFT algorithm, which would need to be recomputed on a buffer each time a new measurement arrives. This way, it is possible to process the data stream in linear time to the number of frequencies which need to be analyzed. This implementation is more efficient for the described purpose of online processing, even when analyzing all frequencies which the FFT would compute. Additionally, it is possible to further reduce the computational costs by skipping the frequencies outside the range of possible human gait frequencies illustrated in figure 6.1.

6.1.3 Experimental Analysis

Figure 6.3 shows an experiment to proof the concept of the odometry estimation. A person carrying a smartphone walks along a 12 m by 9 m rectangular path on an empty parking lot outside (see figure 6.3a), so no systematic measurement disturbances are expected. The particular phone used in the experiment, a Samsung Galaxy S model, does not contain a gyroscope, so orientation estimation and odometry calculation are done using the accelerometer and magnetometer. Both sensors provide readings with 16 Hz in this experiment.


(a) Test path in an outdoor scenario.

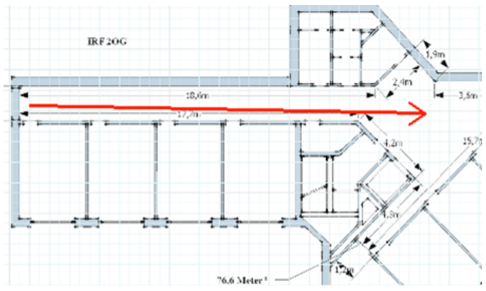


(b) Odometry estimation.

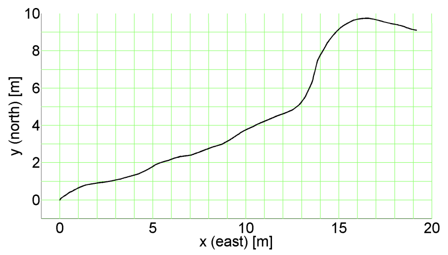
Figure 6.3: Test performed in an outdoor scenario without magnetic interferences.

While measurement frequencies of up to 100 Hz are available on this phone, the choice to reduce the sampling rate as much as possible is due to reasons of energy consumption. Even if the power needed by processing the presented algorithms lies several orders of magnitude below the power consumption even of an infrequently used display, keeping the whole system energy-conservative is typically in the user's interest.

The corresponding odometry estimation is visualized in figure 6.3b. The result contains certain errors in the walking direction estimation, on the one hand the smooth but small direction errors visible in the graph, and on the other hand two short oscillations of the estimated forward direction between the real forward and backward direction. The latter one is not visible in the graph directly, but results in a shortening of two of the sides of the rectangle. This effect is caused by the dependency of the described system on a reliable, exact orientation estimation. Nevertheless, the generated odometry still approximates the real path conceivably well, and for example many robot localization systems work successfully with far inferior motion information.



(a) Test path in an indoor scenario.



(b) Odometry estimation when assuming a constant magnetic field.

Figure 6.4: Straight test path in an indoor environment, demonstrating the influence of magnetic disturbances caused by electric devices.

A problem becomes obvious when considering the following indoor example[2] in figure 6.4. When no gyroscope is available in the mobile device, the magnetometer is the only source of heading information, since horizontal orientation changes cannot be inferred from linear acceleration measurements alone. The earth's natural magnetic field is known and the inclination and declination at a given point on earth's surface are mapped[3], so using the magnetic measurements to estimate the north direction seems reasonable. This however does not account for the numerous small-scale deviations and disturbances in the local magnetic field caused by magnetic metals or electronic devices. Figure 6.4a illustrates the straight path walked in this experiment, while figure 6.4b displays the odometry generated when assuming a magnetic field with a constant horizontal component pointing north. The obvious heading errors in the second half of the path are caused by deviations in the magnetic field leading to errors in the orientation estimation and consequently in the odometry generation.

A first indication about an explanation for these deviations is given in figure 6.5, which shows the distorted magnetic field along the straight path in figure 6.4a. A more detailed analysis is possible with the results in section 6.2, where a mapping is presented in figure 6.7 on page 122, which illustrates the horizontal component of the magnetic field for the whole floor. The derivation process used to generate this map, along with those to be used in the sensor update, will be detailed later, but its outcome, together

[2]The floor layout in figure 6.4a is not aligned with the north direction.

[3]See for example the United States Geological Survey National Geomagnetism Map at `http://geomag.usgs.gov/`.

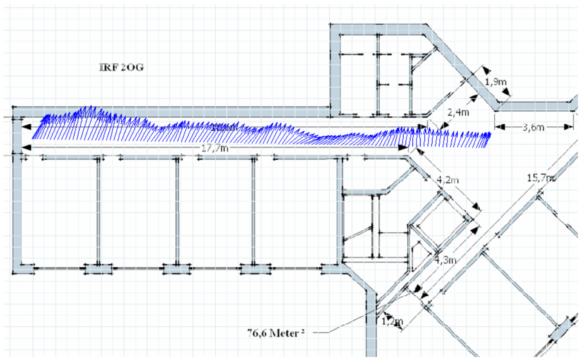


Figure 6.5: Horizontal component of the magnetic field along the straight path. Deviations from the natural earth magnetic field are caused by different electrical devices along the way.

with a closer look at the environment in question, explains the behavior of the indoor odometry estimation experiment presented here: The deviations in the first part of the path are caused by metal filing cabinets, while along the second part of the path several electronic devices are emitting electromagnetic fields of their own, among those are copy machines, printers, Wi-Fi access points, and an elevator.

A possible solution on devices featuring gyroscopes would be to base the orientation estimation exclusively on the rotation velocity and acceleration measurements, ignoring the magnetometer. This however would deprive the system of any absolute heading information and the ensuing drift would cause other problems like the divergence of the heading estimation over time. Neither can the problems be solved by filtering or calibrating the magnetic measurements in a way, since the deviation is spacial in nature (instead of temporal) and not constant.

An alternative solution is to actually map the magnetic deviation occurring in the environment. While such a mapping might be perceived as a significant additional effort, this is actually not the case if the Wi-Fi signal strength information is gathered using a device with similar capabilities as the one which those information are meant to localize, so both measurements can be collected at the same time and the maps can be generated in parallel.

6.2 Mapping and Sensor Model Design for Mobile Handheld Devices

This section covers the design of a probabilistic sensor model to incorporate a smartphone's exteroceptive measurements into a Bayes filter. At the same time, the processes derived here provide the means to correct the systematic errors in the odometry estimation process mentioned above.

6.2.1 Overview of Map Types

In section 3.2, the correction of a predicted position belief state by measurement information has been described in a general way. In most non-trivial cases, the measurement space does not directly map to the state space, so the measurement model $p(z_t|x_t)$ in equation 3.16 normally incorporates a-priori knowledge in form of a map. In section 4.1 and chapter 5, the map representations are directly derived from the respective problem analysis and not discussed in any detail. In the context of smartphone localization using signal strength reception, however, different methods have been applied so far, and a single best approach may not be obvious.

An overview of general map types clarifies the possible design choices and helps to categorize the Wi-Fi localization techniques discussed in section 2.3.2.2. Maps for localization purposes can take different forms:

- *Landmark maps* consist of a list of distinct features, the landmarks, annotated with a vector of characteristics and a location, optionally also with corresponding measurement uncertainties. They can be used if measurements correspond directly to the landmarks, even if those correspondences are ambiguous.

- *Grid maps* are used to approximate continuous features, which are too complex to be expressed as landmarks. Simple examples are rasterized maps, e.g. a 2D grid with each cell corresponding to a square area which might be occupied or free in case of a building layout.

- *Parametrized models* also approximate continuous properties. In contrast to grid maps, the approximation is not in the spacial resolution of the map, but in the (limited) complexity of the model.

Those map classes all have their applications, advantages and disadvantages, depending on the measurement information and its computation. Using Wi-Fi signal strength as an example, localization using propagation models is based on maps of access points as landmarks with known positions and characteristics such as emitted power and signal wavelength. Efforts have also been made to try to augment the simple propagation models with signal reflections and interferences in given environments, resulting in complex parametrized models which better predict the sampled signal strengths. Alternatively it is possible to take the sample measurements along a regular grid to directly get a raster map of the signal strengths, or to simply interpolate between sample measurements to gain a higher resolution grid. In the latter case, normal linear or polynomial interpolation methods do not yield satisfying results when applied to the sample measurements directly, because those methods do not consider noise in the data. Commonly, either averages of a larger number of measurements are taken as input samples for the interpolation to reduce the noise, or a dense set of measurements is smoothed, e.g. using Gaussian kernels which are large compared to the measurements' euclidean distances.

It is interesting to note that, given a continuous or rasterized map of the signal strengths, the fingerprinting localization method using the nearest neighbor in signal space, as described in section 2.3.2, can now be formulated as an optimization problem of finding x to maximize $p(x|z_t)$, and solved for example using gradient search strategies on x instead of comparing the current measurement z_t to all samples in the database [3].

The Wi-Fi localization approaches discussed in section 2.3.2.2, such as fingerprinting techniques or multi-lateration, have in common that they aim to find the best position estimate for a single measurement. Integration over time is done by smoothing these position results, if it is done at all.

In contrast, a Bayes filter requires the seemingly easier task of designing the sensor model $p(x_t|z_t)$ to evaluate the likelihood of a new measurement z_t for a *given* position estimate x_t. A fingerprinting method could be adopted for this purpose in two different ways: Trivially, a position estimate can be calculated by finding the best matching measurement in the database, and its distance to x_t can be used as a heuristic for the measurement likelihood. A more reasonable method would be to find the position in the database closest to x_t and to compare its prerecorded measurement to the recent one to derive a measure for its likelihood. Propagation-based Wi-Fi localization on the other hand directly provides an expected measurement, which could be used together with an additional variance parameter to calculate the measurement probability, e.g. using a Gaussian assumption for the measurement's noise.

Both described versions have disadvantages. The fingerprinting method only offers crude heuristics for calculating a likelihood. A simple distance-based propagation model on the other hand can only approximate the real, highly non-linear reception characteristics in a very limited way. An alternative method without these drawbacks is introduced in the next section.

6.2.2 Gaussian Processes for Mapping

To provide a stochastically sound incorporation of the Wi-Fi mapping information into a localization system as described in section 3.2, the map representation must have a high resolution (or ideally be continuous) and provide not only the expected measurement value, but also its variance at each given location. At the same time, the mapping itself must not be too difficult or time consuming, as this would not be scalable to large areas. Based on these requirements (cp. section 2.3.1), the mapping uses the same smartphone technology as intended for the final localization, with user interface and mapping procedure kept as simple as possible to be performed by untrained personal. Thus the mapping information are collected using a normal smartphone, which records continuous sensor logs together with sparse ground truth positions provided by the user by clicking on a previously loaded map of the environment. These sensor logs contain all measurements, including the inertia and magnetic sensor information as well as the relevant radio signal strength measurements. This data, when augmenting the odometry estimation with the ground truth information, can then be used to generate both the necessary Wi-Fi maps and the magnetic map of the environment in parallel. In the following, all quantities, which need to be mapped, are considered independently from each other.

Given now is a vector of sample measurements $\mathbf{z} = z_{1:T}$ of a measured quantity (either a component of the magnetic field or the received Wi-Fi strength of a certain sender) along a path $\mathbf{x} = x_{1:T}$ in the environment, which is used to generate a smooth, complete map by means of Bayesian inference on Gaussian processes [71]. For this, $\mathbf{z}$ is seen as a noisy T-dimensional measurement of the process $f(\mathbf{x})$ at the positions $x_{1:T}$. $f(\mathbf{x})$ is assumed to depend only on the location, not the temporal context, and is completely specified as a Gaussian Process by its mean and covariance function:

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), P(\mathbf{x}, \mathbf{x'})). \tag{6.1}$$

Given the mean and covariance functions, $\mu(\mathbf{x})$ and $P(\mathbf{x}, \mathbf{x'})$, respectively, many different functions f' are still possible. In this context, the sample data $(\mathbf{x}, \mathbf{z})$ acts as training points or parameters of the process, consequently defining the form of the Gaussian Process. Gaussian Processes can now be used as a form of regression to predict the function value $f(\mathbf{x}_*)$ at a set of certain test points $\mathbf{x}_*$ by simply conditioning the process on the sample set $(\mathbf{x}, \mathbf{z})$.

The main design choice for using Gaussian Processes to predict the signal strength maps is the specification of the mean and covariance functions. Here, a constant mean function is specified and the covariance function is a composite of a squared exponential and a noise function. The covariance function defines the covariance between the different x_i both in the training and test sets. The squared exponential function therefore assigns exponentially decreasing covariance values to z_i and z_j based on the euclidean distance of x_i and x_j. This leaves open the choice of several hyper-parameters specifying the magnitude of the expected mean, the expected noise and two scaling factors for the magnitude of the covariance values and the expected smoothness of the resulting process. Those hyper-parameters can either be manually set or learned automatically using the sample data set $(\mathbf{x}, \mathbf{z})$ and its likelihood for a given set of hyper-parameters as an maximization criterion. This provides a stochastically sound and largely automated way of inferring signal strength maps.

The Wi-Fi signal strength mapping for different received wireless network BSSIDs[4] in the indoor test scenario is illustrated in figure 6.6. It is important to stress that the data collection for these maps is done in a 6 minute run during which the user specified 23 ground truth positions. While maps of similar quality have been presented elsewhere, those have mostly needed a considerably larger dataset and consequently a much larger data acquisition phase.

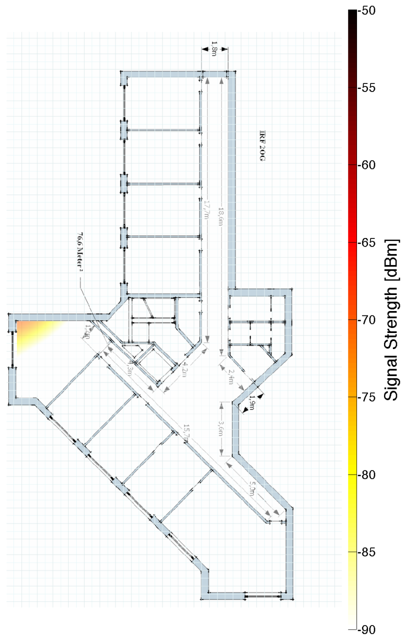
Note further, that the Gaussian Processes in figure 6.6 are created without the information of the building layout or the real positions of any of the access points. In fact, for all but two access points, these positions are not even known, nor do they need to be, which presents a significant advantage of this method. While the Wi-Fi signal in figure 6.6c could also be approximated by a propagation-based model, the models in figures 6.6b and 6.6d significantly deviate from any distance-based function.

The resulting Gaussian Process, with parameters given by the recorded sample data and automatically learned hyper-parameters, can now be used directly as the measurement model $p(z_t|x_t)$ of equation 3.16 in section 3.2 to predict the measurement at a position x_t, and also its expected uncertainty. Alternatively, the Gaussian Process can be used to generate a look-up table in form of a discretized grid map for those expected measurements. This is especially useful when applying the same method to a magnetic mapping which is based on a much larger sample set due to the higher sensor frequency of the IMU compared to Wi-Fi scanning. A map of the magnetic declinations in the test environment is visualized in figure 6.7. The difference between minimal and maximal declinations measured in this relatively small area even exceeds $90°$, further stressing the necessity of such a mapping when using the magnetometer in any way for position tracking, but especially when using a procedure as described in section 6.1.

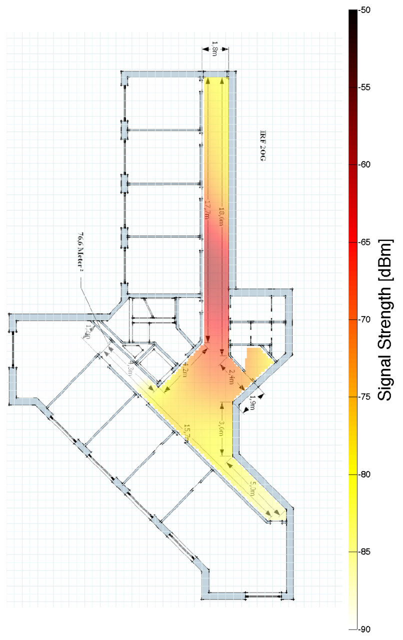
6.3 A Bayesian Sensor Fusion Approach to Mobile Phone Localization

To provide a proof of the concept described in the previous sections, a particle filter as introduced in section 3.4 has been implemented. This filter consists of a set of particles, each of which represents a distinct position hypothesis and is continuously updated by odometry and measurement information. This fuses the sensor information, i.e. the Wi-Fi measurements and the odometry estimation, with the knowledge about the environment in form of the building layout, the mappings of the magnetic field and the signal strengths of different Wi-Fi access points.

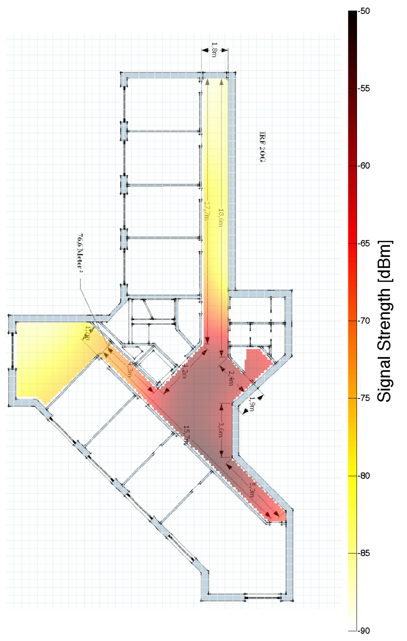
[4]A basic service set identification (BSSID) uniquely identifies not only a wireless network, which might be backed by several access points, but also the specific access point from which it originates.



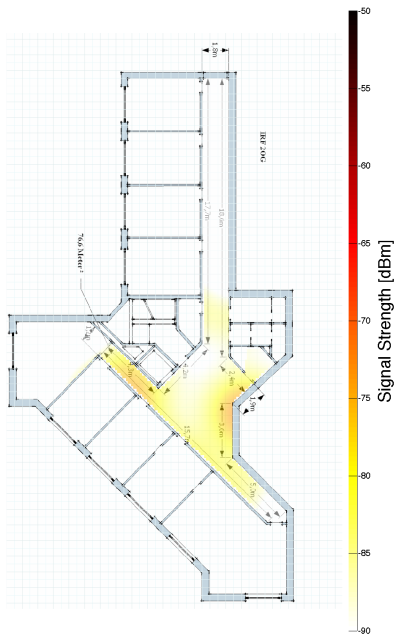
(a) Signal strength of BSSID 00:0b:86:0a:5d:02.



(b) Signal strength of BSSID 00:14:a8:14:87:f9.

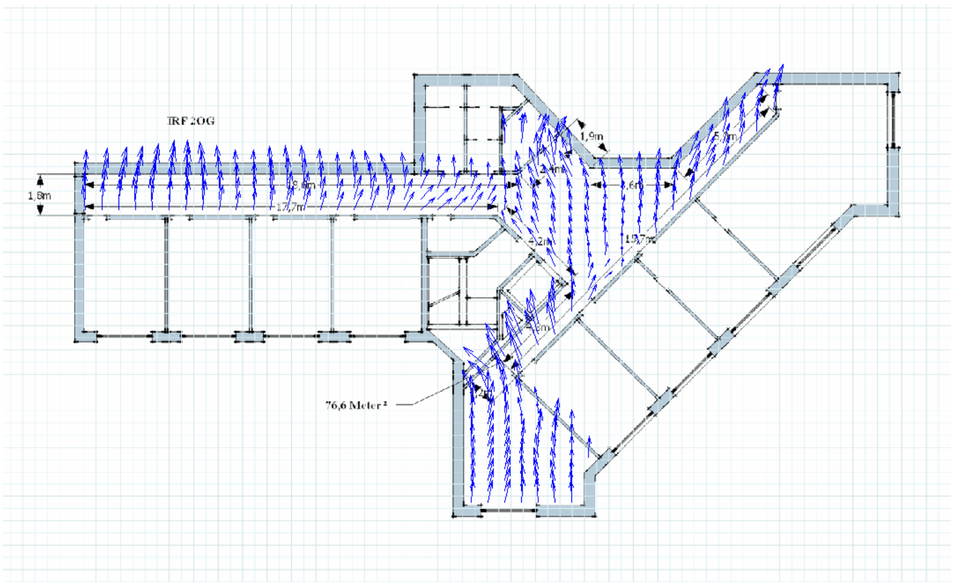


(c) Signal strength of BSSID 00:12:80:e1:99:d0.

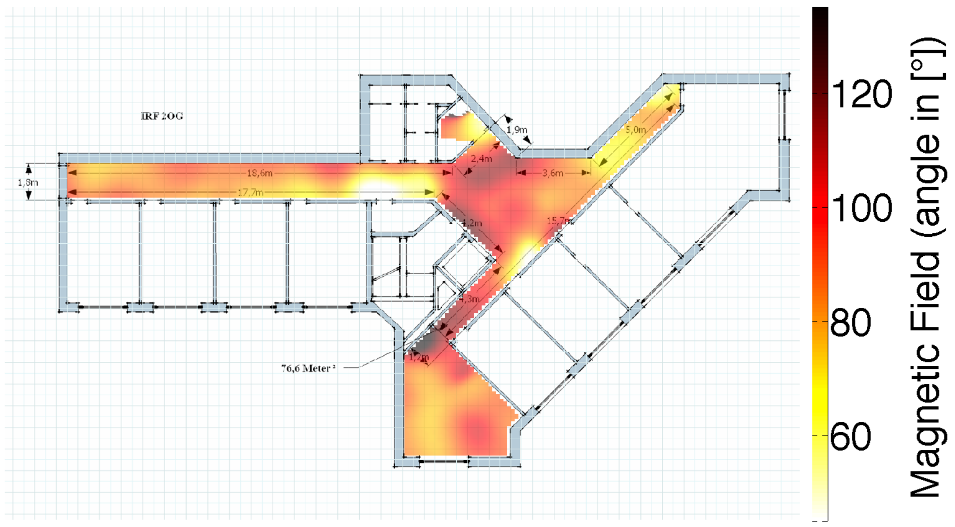


(d) Signal strength of BSSID 00:12:80:e1:a1:70.

Figure 6.6: Results of signal strength mapping for four out of more than 20 received wireless network BSSIDs.



(a) Horizontal component of the magnetic field.



(b) Angles of the horizontal component of the magnetic field.

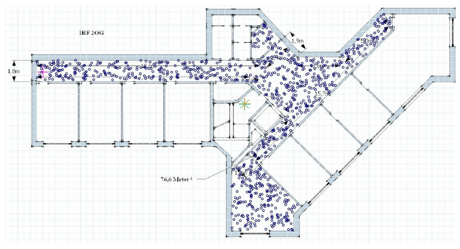
Figure 6.7: Results of the mapping of the magnetic field's declination.

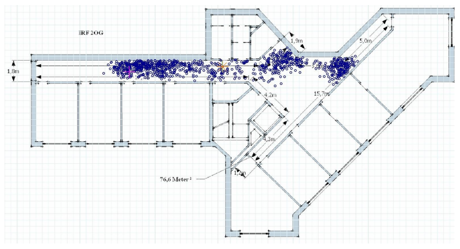
Once again the same state space choice arises as discussed in section 4.1.3.1 for the application of a particle filter to intralogistic tracking. In this case, it is possible to either model the state space with the 2-dimensional position only, or additionally with the orientation. In the 2-dimensional case, the orientation is used from the odometry estimation directly. In the other case, the current walking orientation would make up the third dimension of the state space vector and would be tracked accordingly. In this scenario, however, assumptions like mainly straight walking directions cannot be made in general. A more compelling argument for the use of the 2-dimensional state space in this specific case is the sufficiency of fewer particles, which is a critical point in terms of the energy consumption criteria and limited use of processing power as stated in section 2.3.

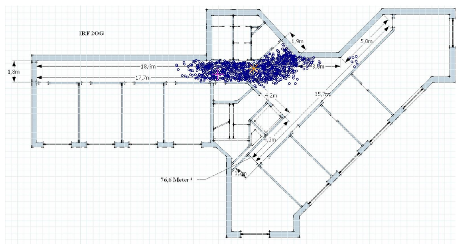
The choice here is therefore to model only the 2-dimensional position with the particles. The heading information is not included as a third dimension, but expressed in the orientation estimation as described in section 6.1.1. As mentioned in section 6.1.3, the magnetic field mapping is needed to compensate the environmental influences on the odometry estimation. Using the particle's location, it is possible to look up the expected declination to the true north direction in the magnetic field map at that position, and thus to correct the odometry estimation's direction for each particle separately. The motion update step in the particle filter allows for a specified amount of noise in the motion execution, thus effectively spreading the particles. In this implementation the noise is modeled with three separate components: an angular error in the heading, a scale error in the covered distance, and an additional white noise translational error. The latter one allows small movements in the particle distribution even when the person is not walking, which is useful for position corrections based on new Wi-Fi measurements. Wi-Fi measurements as well as the fact that it is not possible to walk through walls are used to weight the particles, and the accumulated weights are used periodically to resample the particle set.

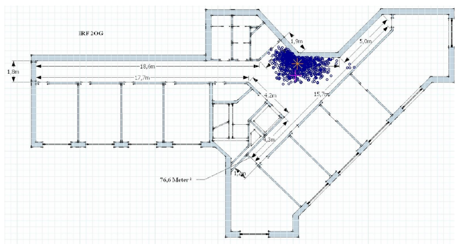
The resulting localization algorithm essentially only needs 5 parameters: 3 for the amount of motion noise, one resampling threshold, and the number of particles. In contrast to most approaches found elsewhere, the expected variance in signal strength does not need to be specified for each access point manually, but is already included in the Wi-Fi maps, as the Gaussian process infers both the mean and the variance at each position. There has also been no selection among the signals as for example done in [59], where only the strongest signals are used for localization due to being a "more stable" source of information. Instead, the presented algorithm relies on the stochastically sound mapping integration and uses all available BSSIDs without discrimination, filtering, or averaging those which come from the same access point. Note that only two access points are located inside the area used for localization and provide measurements above -70 dBm.

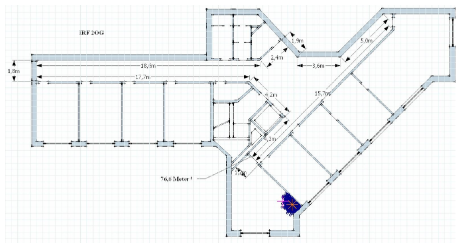
In the following, the results of a 10 minute walk around the second floor of the indoor test environment are presented, during which the user provided 41 reference points by inputting his current position on the smartphone's touch display. Of course this only approximates the real positions up to a certain precision, and the comparatively easy layout of the building further simplifies the task. Nevertheless, this experiment gives a good impression of the potential of the proposed system. In fact the odometry information alone, i.e. without the use of any Wi-Fi measurements, is sufficient to provide a good localization estimate when combined with the building layout information. This is not only true for local localization, i.e. position tracking from a known starting point, but also for global localization, where no prior information about the position is available. Figures 6.8 and 6.9 illustrate the first minute of the filter's particle distribution, without using Wi-Fi measurements and with using them, respectively. Figure 6.9 shows the filter's particle distribution in the first 45 seconds to illustrate the global localization (see figures 6.9a to 6.9d), and some exemplary later frames.


(a) Initial particle distribution.


(b) Particle distribution after 20 seconds.


(c) Particle distribution after 32 seconds.


(d) Particle distribution after 45 seconds.


(e) Particle distribution after 124 seconds.

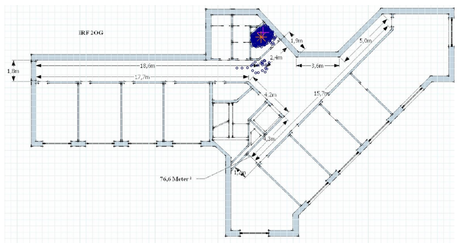
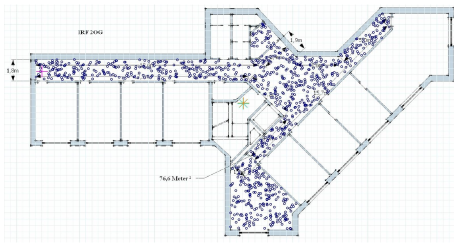
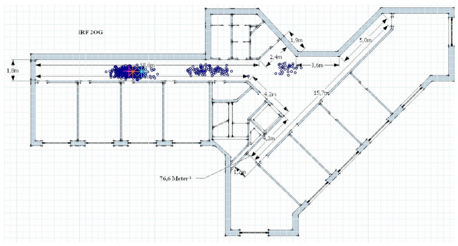
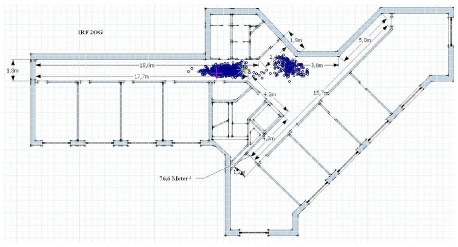
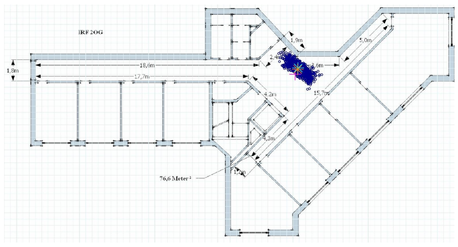

(f) Particle distribution after 533 seconds.

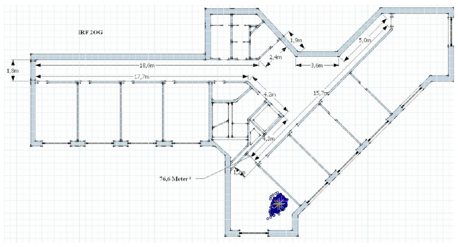
Figure 6.8: Global localization without using Wi-Fi information.


(a) Initial particle distribution.


(b) Particle distribution after 20 seconds.


(c) Particle distribution after 32 seconds.


(d) Particle distribution after 45 seconds.


(e) Particle distribution after 124 seconds.

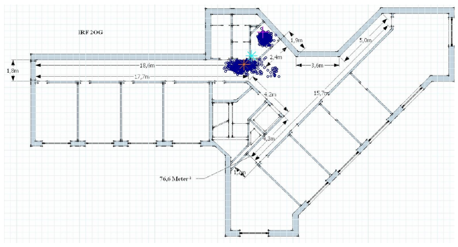

(f) Particle distribution after 533 seconds.

Figure 6.9: Global localization using odometry and Wi-Fi information.

Figure 6.10 shows the localization errors at the times of available reference points for four different setups of the localization system, with and without known starting position (i.e. position tracking and global localization), and with and without using Wi-Fi signal strength measurements. It can be seen that in this test the global localization estimation converges on the position tracking estimation after 50 seconds, from which time on the particle distributions corresponding to position tracking and global localization do not show significant differences any more. In general, the additional Wi-Fi scan information provides a significant precision improvement, as expected. In a few cases, consecutive Wi-Fi scan results, which deviate from the expected mean, actually degrade the positioning accuracy. This can be seen around the 533 second mark in figure 6.10. The corresponding multimodal particle distribution is visualized in figure 6.9f, where one cluster is in the correct position of the small room, while a second cluster is in the corridor and actually gets higher weights in the Wi-Fi sensor updates. However, it has to be noted that the position extraction from the particle set by calculating the weighted average does not take into account these possible multi-modal particle distributions, thus not providing the center of the biggest cluster, but just averaging between two clusters. So replacing this simple strategy by a more sophisticated clustering technique might improve the results without actually necessitating any changes in the localization filter itself.

The higher localization errors from seconds 250 to 300, on the other hand, are caused by errors in the odometry calculation. Due to inaccuracies in the estimation of the device orientation and the phase shift between the vertical and forward component of the step frequency, the traveled distance has not been estimated correctly, which results in significantly increased errors in all graphs. The additional usage of Wi-Fi measurements in two of the systems in figure 6.10 could only partially compensate these errors.

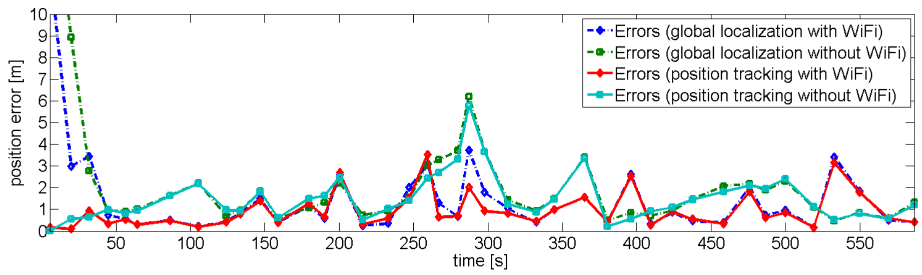


Figure 6.10: Localization errors for a 10 minute test run. Each data set is the average of 100 filter executions with the same input data.

Over all, average errors of around 1 m can be reached in this example run when using Wi-Fi measurements (0.99 m and 1.11 m for position tracking and global localization, respectively). Average errors of around 1.6 m are possible even without Wi-Fi, only using the estimated odometry and the floor plan information (1.61 m and 1.66 m for position tracking and global localization, respectively). A precision of up to 1.5 m after 40 seconds when starting from global localization has been achieved in 60% of the reference measurements without Wi-Fi information and in 74% when using Wi-Fi for localization.

These results are well within the desired range to provide location-based services for applications as described in section 2.3. This approach is set apart from other published results in its accuracy and especially in the simplicity of its preparation. It does not need any additional access points or even knowledge about the positions of existing ones. In fact, in the presented setup, such positions are only known for 2 access points, so propagation-based approaches could not even perform trilateration to deliver a single pose hypothesis based on the signal strength measurements, even if the distance would perfectly correlate with the signal strength, which is not the case here. Fingerprinting approaches on the other hand usually require a much larger data set.

The data set used in [33] amounts to 51,249 Wi-Fi scans at 510 predefined positions in a roughly 20 times larger building than the one used here, and took around 28 man-hours overall to record. The scans include signals from 33 different access points within the area. The corresponding evaluation of localization accuracy only covers test positions identical to the training positions with multiple Wi-Fi scans at each location. In contrast to this, the whole data acquisition phase for the experiments described here is done in a single 6 minute test run with 23 user supplied ground truth points, and the localization takes place in a continuous state space. An accuracy comparison, however, is difficult, as the cells defined by the positions in [33] mostly correspond to separate rooms of around 10 m^2 in average. For these, the correct response is reported in overall 97% of the tests, while the illustrations of cell accuracy and building plans suggest that the few cells located in the same rooms have a far lower accuracy of as low as 70%.

Direct comparisons are possible to the following publications: In [59], a smartphone is used to record both training and test data, and several fingerprinting methods are compared and presented. While nothing is said about the training set size, a precision below 2 m is reported only in 48% of the cases for the best applied technique, which applies temporal smoothing of the signals as well as averages those known to originate from the same access point. The work in [77] is done using a differential drive robot and a probabilistic localization approach, comparing a signal strength grid map with a discretization of 2 by 2 meters to a propagation-based model in a setup only marginally larger than the one presented here, with 3 access points with known positions within the building. The maps are generated using the robot, but no information about mapping time or the used sample size is given. The best presented propagation-based model only achieves an error below 1.5 m for 61% of the time. The results using the signal strength maps show an error below 1.5 m for 82% of the time, which is slightly better than the results presented here as 74%. In these experiments, however, odometry information are available due to the differential drive locomotion in a far easier way and with superior accuracy. Nevertheless, this just stresses the usefulness of applying probabilistic robotics methods to the presented application scenario for the localization of smartphones.

6.4 Conclusion

In this chapter, a Bayesian sensor fusion approach to mobile phone localization is introduced, that aims to pave the path for high-resolution positioning in future location-aware services such as detailed in section 2.3. No additional hardware is assumed beyond standard, market-available smartphones. This scenario presents a challenge in various respects: A smartphone's exteroception, when not using its camera, does not provide any directional measurements from which geometric relations to its environment can be derived. This together with the usually rather complex environment itself stresses the need for a robust stochastic sensor model to allow position corrections. The most significant problem, which probably explains the lack of probabilistic localization techniques applied to smartphones in current literature, is the lack of direct access to the involved locomotion process. The locomotion's intention as well as execution reside within the person carrying the phone, and can thus only be accessed indirectly through the smartphone's proprioception.

Solutions to both these challenges are presented here, as well as their incorporation into a stochastic filter. Using this probabilistic localization technique, it can be shown how odometry estimation using the smartphone's embedded sensors fused with Wi-Fi scan results can yield compelling position data, if backed by map information for magnetic field emission and Wi-Fi radio data. Overall, the sensor fusion approach outperforms common heuristic Wi-Fi localization techniques on smartphones, and shows a superior performance with reaching an average error of 1 m even using simple and time-saving procedures for gathering the necessary mapping data.

Chapter 7

Multi Hypothesis Kalman Systems

[1] In the robotics domain, a variety of algorithms has been proposed to solve localization tasks, ranging from efficient and robust multiangulation methods [4] to constraint-based techniques [28]. The majority of commonly applied algorithms relies on probabilistic methods and Bayesian algorithms such as introduced in chapter 3. Key aspects for practical systems are efficiency, robustness and appropriate handling of uncertainty in the data in form of noise in odometry and sensor readings as well as false positives.

As expressed by Gutmann and Fox, it seems to be common consensus that "Markov localization is more robust than Kalman filtering while the latter can be more accurate than the former" [32, page 454], where "Markov localization" in this context refers to non-parametric Bayes filters such as particle filters or grid-based Bayes filters. Of course, this is only expressed as a general trend, and hybrid solutions, special variants and problem-specific adaptations always have the potential to outperform pure implementations of the general methods. However, when belief distributions are expected to be multi-modal, particle filters are often the method of choice [86], even if Kalman filters based on multiple models have been well established in other fields of research [1], also referred to as Gaussian sum or Gaussian mixture filters.

As described in section 3.3, the Kalman filter was originally derived for estimating linear Gaussian systems. Non-linearity aspects are addressed by variants such as the Extended or Unscented Kalman filter. Those linearize the process and measurement functions in order to maintain the convenient Gaussian representation throughout the filter update steps.

However, there are systems in which the non-linearity is too severe, thus

[1]Parts of this section are based on an article in the Proceedings of RoboCup 2011: Robot Soccer World Cup XV. The corresponding publication, see [41], is available at `http://www.springerlink.com`.

causing Gaussian approximations to perform unsatisfactory or even to diverge completely. This is obvious especially for multi-modal systems, as a single Gaussian distribution can not represent a distribution with multiple separated maximums. Such non-linear non-Gaussian systems can be approximated by weighted sums of Gaussians [1]. Those filters are referred to as Gaussian sum or Gaussian mixture filters. Every belief distribution can be approximated closely by an appropriate number of Gaussians theoretically. In practice, higher numbers of Gaussian terms lead to an increase in computational complexity which is impossible to manage. Due to this, multiple-model Kalman filter implementations loose some of their generality, computational efficiency and theoretical elegance. This might explain the under-representation of these algorithms in the robotics domain. Such a Gaussian mixture filter aiming at high performance and applicability on limited platforms also faces some of the problems of operating particle filters with an extremely low number of particles. This chapter's main contribution is to point out and apply techniques, which have originally been introduced in particle filtering contexts, to multiple-model Kalman filtering, to establish the stochastic reasoning in the context of common Gaussian mixture filtering, and to present an evaluation of the proposed method in a number of experiments with real robots.

The next section gives an overview about Gaussian mixture filters in general, followed in section 7.2 by the problems arising from necessary approximations, ambiguous measurements, false positives and possible kidnapped-robot situations. Section 7.3 proposes an alternative approximation to the common implementation of model splitting and merging and establishes its stochastic soundness. This has been implemented for a RoboCup Standard Platform League scenario (cp. section 2.1), i.e. for humanoid robots with highly uncertain odometry in a dynamic soccer scenario where most landmarks are ambiguous field features, occlusion frequently occurs and false positives are likely to be generated from observations of the audience. Situations with various degrees of similarity to robot kidnapping happen due to frequent struggles and shoving among the robots and interventions by the referees. This work is related to [70] in terms of the application scenario and the general idea of using a multiple-model Kalman filter to address the correspondence problem for ambiguous landmarks and false positive observations. However, the proposed solution differs in the overall approach and therefore also in various implementation details. Efficiency, localization quality and robustness are evaluated in section 7.4.

7.1 Gaussian Mixture Filtering

This section gives an overview of the applicability of Gaussian mixture filtering to state estimation in domains of multi-modal probability distributions.

Note that in most applications only a subset of these aspects is actually implemented [70, 68, 21, 50, 88, 47]. In order not to commit to any specific form of non-linearity approximation to the Kalman filter, the following discussion makes use of the general Bayes filter convention under the Markov assumption (cp. section 3.2), specifically the initial belief distribution $bel(x_0)$, the motion update in equation 3.11 and the sensor update in equation 3.16.

In case of linear Gaussian systems, the estimation problem is solved optimally by the standard Kalman algorithm. Real problems, however, rarely fulfill those criteria. Non-linearity aspects concern the process and sensor model. Those need to be linearized in order to apply the familiar Kalman filter equations, which then do not yield the optimal solution but only an approximation. This can be done either using Jacobians in the Extended Kalman filter or the unscented transform in the Unscented Kalman filter.

Non-Gaussian aspects can influence the Bayes filter concept at several further points, even if the sensor update seems to be the most focused on in work related to robot localization. Approximating non-Gaussian systems with Gaussian mixtures offers a solution to those issues. The belief is now represented by

$$bel(x_t) = \sum_{i=1}^{N_g} \alpha_i \frac{1}{(2\pi)^{n/2}|P_i|^{1/2}} e^{\left(-1/2(x_t-\mu_i)^T P_i^{-1}(x_t-\mu_i)\right)} \tag{7.1}$$

where μ_i and P_i are the means and covariances of the individual Gaussians (cp. equation 3.17) and α_i are the weights which sum up to 1 over all N_g models. N_g might change during operation of the filter, as will be explained in the following sections. The explicit dependency on time in the indexes of those parameters is dropped for simplicity.

The changes introduced by Gaussian mixtures will be addressed in the following sections. In practical applications, those are rarely implemented all at once. Most implementations only use the possibilities of the Gaussian mixture representation in the one or two aspects most crucial to the filtering process and enforce a strict limit to the number of separate Gaussians to maintain acceptable processing time.

7.1.1 Initial Belief and Re-localization

The first aspect, which would only be insufficiently described by a single Gaussian, is the prior belief $bel(x_0)$. A very simple and often used assumption is that of a uniform initial belief distribution over the entire state space. Since this cannot be expressed with any accuracy with classic Gaussian filters, those are neither applicable for global localization nor for re-localization, e.g. in kidnapped robot scenarios.

Any given initial belief can be approximated with a mixture of Gaussians with uniformly distributed means μ_i, e.g. on a grid, weights α_i according to the value of the belief at each mean's position, and symmetric covariances P_i scaled to minimize the difference between the given target function and this mixture representation. This has originally been proposed in [1]. Sometimes the initial belief is alternatively generated using the first sensor measurements, e.g. as done in [21] and [47]. While this seems intuitive, it relies much more on the first measurement than on all the following ones and might fail in case this first one is a false positive, when no hypothesis is maintained in regions not conforming with this measurement as done in [47]. Several other strategies have been proposed for the specification of the initial belief, some of which are related to the re-parametrization problem covered later in section 7.1.4.

Re-localization from kidnapped robot scenarios is closely related, since allowing for the possibility of robot kidnapping means assigning a small probability to the case that the robot is repositioned without any knowledge, which is similar to the initial global position finding. Special care has to be taken that the weights of Gaussian terms introduced to allow re-localization are small enough not to disturb the normal localization tracking process, but that those small weighted terms do not always die out without the chance to be validated by enough measurements to become important.

The injection of additional Gaussian terms for the purpose of re-localization bears some similarity to the "spawn and birth" models in Gaussian Mixture Probability Hypothesis Density (GM-PHD) Filters [7], where the "birth" of additional Gaussian terms accounts for the appearance of previously unknown or un-tracked targets. Mathematically, the intensity for spontaneous birth in this multi-target tracking scenario is identical to allowing a small repositioning possibility in localization applications. Instead of representing a new hypothesis for the whole state estimation, however, those new terms in the GM-PHD Filter represent only possible new features in the state estimate.

7.1.2 Process Update

As mentioned earlier, non-linear process models $p(x_t|u_t, x_{t-1})$ are commonly addressed by linearizing around the current mean in Kalman filter applications. This approximation is appropriate as long as the non-linearity around the current state is not too severe and the state is corrected frequently enough by measurements.

Walking robots often produce highly uncertain odometry where large errors are to be expected especially in the rotation component (cp. section 4.2.2). Longer periods without observation leave an initial Gaussian belief spread out in a way that it cannot be approximated properly by a single Gaussian. In this case it seems reasonable to design the prediction step in equation 3.11 to make full use of the Gaussian mixture representation. This is done by modeling the process noise itself as a Gaussian sum. Assuming multiple terms in the process noise's mixture representation, the overall number of Gaussians in the filter increases by this factor, i.e. the number of terms in the Gaussian, with each process update. Each term's weight is the product of the weights of the original belief hypothesis and of the process noise term. To maintain efficient computability, this increase needs to be countered by pruning the resulting belief representation as described in more detail in section 7.1.4.

7.1.3 Sensor Update and Correspondence Problem

Two different ways to apply Gaussian mixtures to the sensor update step can be distinguished. Analogous to the process update, non-linear sensor models producing non-Gaussian beliefs over time can be approximated using Gaussian sums. This makes sense in situations where several consecutive measurements produce non-linearity spreading the belief in a similar way. However, when measurements are often complementary, e.g. bearing-only measurements associated with landmarks in different directions, the sensor model's non-linearity aspect does not build up but is kept at bay by those measurements soon enough, so that the state can be approximated satisfactory by a single Gaussian.

A different case arises from the observation of ambiguous landmarks and the resulting correspondence problem, which is of great importance in many localization applications. For multi-modal systems, the common linearization techniques already make a Kalman filter operate "more as a maximum likelihood estimator than as a minimum variance estimator and the mean follows (hopefully) one of the peaks of the density function", as stated by Alspach and Sorenson in [1, page 440]. Thus it seems intuitive to handle this case in single-Gaussian Kalman filters by choosing the correspondence with maximum likelihood and proceeding by linearizing a sensor model for a unique landmark update.

The natural way to apply Gaussian mixtures in this context is to construct a sensor model with one Gaussian term for each possible correspondence. In [68], Gaussian mixture sensor models are used to avoid an exclusive correspondence choice for the likelihood calculation in a particle filter localization. In the Kalman approach, the sensor update using this model results in applying all possible correspondences to all hypotheses maintained by the current belief prediction $\overline{bel}(x_t)$ as done in [70]. The terms in $bel(x_t)$ therefore increase by the factor of the number of terms in the sensor model.

Furthermore, the weights α_i in the Gaussian sum are updated in two steps: first according to the splitting of models, then according to each separate update. In case of model splitting as for ambiguous landmark observations, the weight of a new term is the product of the weights of the original belief hypothesis and the correspondence choice. This first adjustment can be skipped in case of uniform correspondence possibilities as done in [70], since the weights are normalized later.

The weight update in each separate sensor update (with fixed correspondence) functions to adjust each weight according to "how well a model explains the observations". [1] proposes to recursively adjust the weight α_i by multiplication with the probability of the measured k-dimensional innovation $(z_t - \widehat{z}_i)$ by

$$\alpha_i = \nu \alpha_i' \left(\frac{1}{\sqrt{(2\pi)^k \left|P\right|}} e^{-\frac{1}{2}(z_t - \widehat{z}_i)^T P^{-1} (z_t - \widehat{z}_i)} \right) \tag{7.2}$$

where $\widehat{z}_i$ is the expected observation for the fixed correspondence according to the ith model, P is the sum of the measurement and the prediction covariance, and ν is a normalization factor. Equation 7.2 is slightly adapted by [70], adding a term ϵ_0 expressing a static probability of the observation being an outlier, i.e. a false positive in the measurement process such as echoes in sonar data or incorrectly classified objects in a vision system. This is done to improve the lack of robustness to outliers. A further discussion of the implications of this will be given in section 7.2.

7.1.4 Pruning the Belief Representation

To maintain efficient computability, the number of terms in the Gaussian mixture representation of the belief needs to be limited. Both the process and sensor update potentially lead to a multiplicative growth in the number of terms as described in sections 7.1.2 and 7.1.3, respectively. This multiplicative increase per time step results in an exponential growth of Gaussian terms over time, consequently preventing any efficient computation. Therefore, this is countered by pruning the resulting belief representation.

Several different methods have been proposed for this. The general formulation for the task is to find a suitable Gaussian mixture representation with a specified maximum number of terms approximating a given probability distribution. This is similar to finding the initial parametrization as described in section 7.1.1, only in this case the target distribution is already given as a Gaussian mixture, but with a higher number of terms. In [88] this is done by computing a re-parametrization by an iterative expectation maximization approach with a heuristic initialization using an efficient version of k-means clustering. In [51] a distance-based method is proposed for density estimation as a weight optimization problem for Gaussian mixtures with a regularization term to achieve smoothness in the resulting densities. The method achieves a good approximation quality with sparse representations compared to several other approaches using expectation maximization, support vector machines, and Gaussian process regression.

In most practical applications however, such sophisticated re-parametrization strategies are often too time-consuming. Instead of applying iterative optimization or regression procedures, simple heuristics are used to reduce the number of terms in the Gaussian mixture. This is done by combining multiple terms into one and by neglecting terms with small enough weighting factors whenever possible [1, 70].

7.2 Efficiency-related Trade-offs and Model Limitations

The extension to use sums of Gaussian for Kalman filtering offers a theoretically sound and intuitive solution to most of the problems classical Kalman filters face in realistic application scenarios. However, the compromises which have to be made to ensure an efficient computation, as well as some of the common assumptions described in section 7.1, potentially nullify certain aspects which make the Gaussian mixture representation desirable in the first place. Two potential problems related to pruning and to the handling of false positives are described in the following section. This serves as a motivation for the alternative approach described in section 7.3, which differs from other common implementations [1, 70].

7.2.1 Influence of Pruning on Quality of Estimation

As shown in section 7.1, multi-modal filter updates potentially produce exponential growth of the number of terms in the belief representation. This implies the necessity of pruning those terms to maintain a limit of the computational complexity. The resulting decrease in estimation quality is often neglected or taken as unavoidable without further discussion. Alspach and Sorenson state in [1, page 442] that it is possible to perform model merging "without seriously affecting the approximation". This statement is only valid as long as the entire state space is still covered, which is obviously the case for distributions along a grid as in [1]. For higher dimensional state spaces and strict real-time limitations on mobile platforms, such a coverage can often not be maintained. Consequently, pruning is applied to an extent far beyond the reduction of clusters along a grid to separate Gaussians. Instead, also small terms are removed which do not coincide with other more significant terms. This is either done by incorporating them into the closest other term, changing its mean and variance slightly so that the new Gaussian's tail end accounts for the removed term, or the small term is neglected completely [70].

Such pruning policies do not only affect the quality of the approximation by the obvious loss of information from removing those terms, but also by further effects resulting from this. One of the advantages of the Gaussian sum representation lies in the fact that the approximation introduced by linearizing the process or sensor models is better for a belief represented by

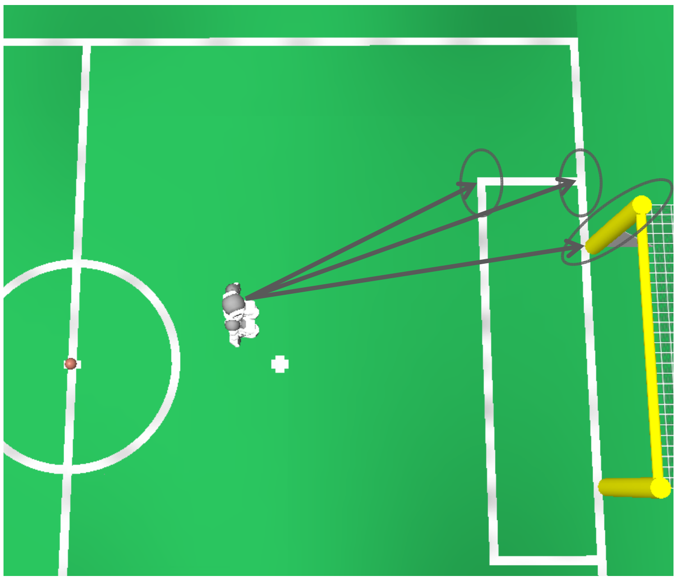
Gaussians with small covariances compared to large ones, which obviously extend further away from the point around which the model is linearized. Thus the quality of approximation corresponds directly to how small the covariance around the separate Gaussians is. Since pruning reduces the number of Gaussians resulting in bigger covariances, this conflicts with the advantage of the Gaussian sum against single big Gaussians even in situations of uni-modal belief distributions.

An even more significant problem shall be illustrated using a simplified example. Consider a robot which is well localized with a current belief state consisting of a single Gaussian. Its position is then altered at a time t_0, e.g. by a collision with another agent, changing the robot's orientation significantly without perceiving it. In figure 7.1, the observation the robot makes *after* its orientation change might well correspond to a different feature, which had a similar relative position *prior* to the pose change. So in the following time steps the robot repeatedly makes ambiguous observations, which correspond to one of two possible landmarks in its map. In this scenario (see figure 7.2a), let each blue link correspond to the correspondence choice c_1 and each red link to choice c_2. c_1 corresponds to the real observed landmark, but c_2 is the choice which fits the (wrong) initial belief state $bel(x_{t_0})$ better. Figure 7.2 visualizes the model splitting resulting from each multi-modal sensor update without any pruning.

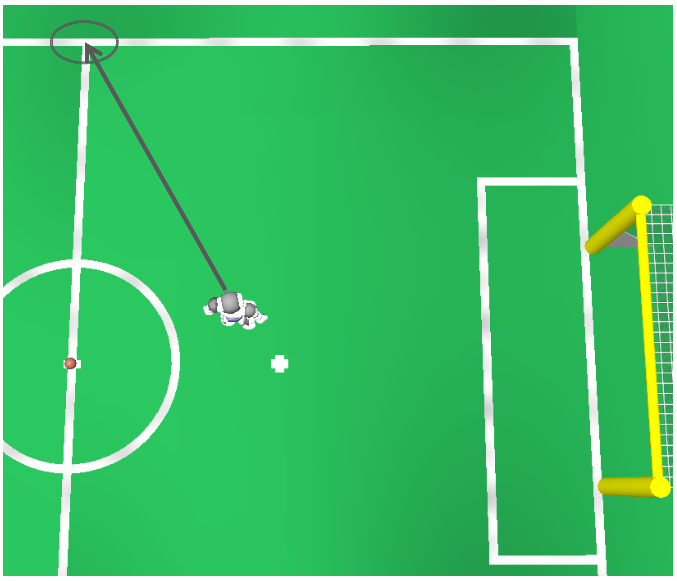
In this situation, updates based on the correspondence choice c_2 result in small differences between expected and real measurement, while those based on choice c_1 initially produce large ones, which only decrease when frequent updates using c_1 shift the corresponding mean closer to the true position. In this scenario, continuous multi-modal updates obviously do not resolve the localization ambiguity, but lead to a belief state $bel(x_{t_i})$ including a term $\alpha_{2,t_i} N_{2,t_i}$, which is the result of a series of "correct" choices and which, after the corresponding mean is close to the true position, also has a high weight. In a situation like this, few additional complementary observations of different landmarks might resolve the ambiguity and leave the true position as the most probable estimate in the belief. Thus the Gaussian mixture filtering allows for more than just mere maximum likelihood estimation, if extensive pruning does not interfere with this characteristic.

Assuming equal a-priori probabilities for both data associations and assuming simple uni-modal process updates, the weight factors α_i of the different terms in the belief of the following time steps t_i is changed exclusively by the sensor update in equation 7.2. Since initially the correspondence choices c_1 do not fit the belief as well as the alternative choices do, the weight α_{2,t_3} in figure 7.2 is much smaller than α_{1,t_3} at time step t_3. This makes paths in a tree of correspondence choices, which do not instantly lead to high likelihood estimates, ideal candidates for pruning techniques as described in [70].

In other words, should a robot come into a situation, in which an "incorrect" correspondence choice initially provides a higher likelihood, then

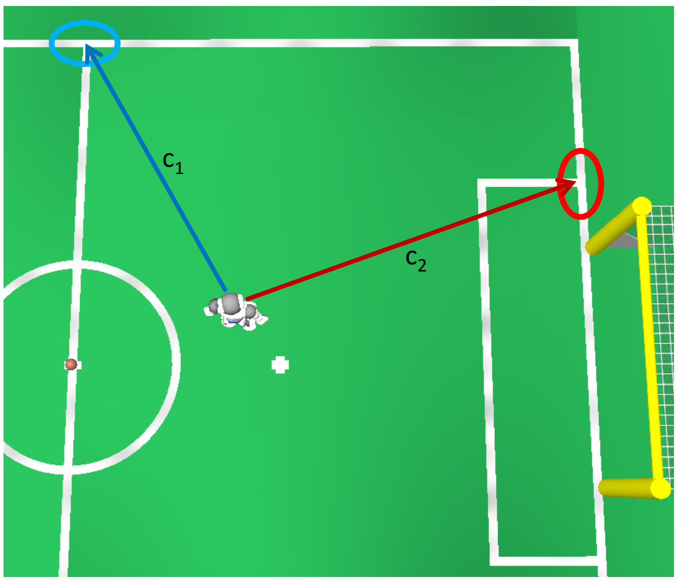


(a) Observations before orientation change.

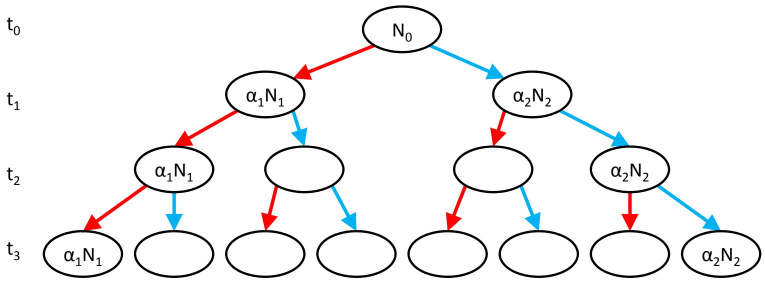


(b) Observations after orientation change.

Figure 7.1: Correspondence situation.



(a) Correspondence choices for the single ambiguous observation.



(b) Model splitting without pruning.

Figure 7.2: Model splitting resulting from a series of ambiguous landmark observations.

an aggressive pruning strategy prevents the exploration of other less likely paths of correspondence choices. Even if one of these paths leads to an overall better and more likely later estimate, none of them is followed long if it does not lead to a competitive likelihood sufficiently fast.

So this kind of aggressive pruning does not only decrease the estimation quality, but actually removes one of the most significant advantages of Gaussian mixture filtering: the possibility to maintain different hypotheses of which some may temporarily be unlikely, but still allow the observation of the influence of new observations on regions of the state space away from the maximum likelihood estimate. Note that this is also essential for relocalization in kidnapped-robot scenarios.

Since the described behavior is a recognized deficiency of such aggressive pruning, there also exist strategies to prevent this without allowing too excessive growth in the belief representation. Among those are heuristics such as time-to-life counters for certain new terms which prevent early deleting or merging with other more confident models. Another possibility is to randomly generate new terms by applying the full innovation for a belief term's mean and a correspondence choice without taking the scale of the covariances into account, thus allowing a "jump" into a region of the state space which might provide higher likelihood for a combination of future measurements and correspondences. All those strategies, however, are heuristic in nature without a solid stochastic foundation, and are therefore rarely mentioned in publications.

7.2.2 Integrations of False Positive Observations into the Sensor Update

The robustness of localization algorithms expresses the filter's ability to maintain a good estimate under various amounts of noise. One aspect of this is the inaccuracy of the process and measurement model, often expressed by an additive random component which is assumed to be of a Gaussian distribution. Kalman filters are particularly effective in dealing with such noise. False positive observations or outliers are another aspect. Those account for the fact that in real-world applications measurements are significantly more often far from the expected mean than the main distribution would suggest if it would be pure Gaussian.

In many implementations of Kalman localization algorithms, such outliers are not handled explicitly, but instead the measurement noise parameter is increased to cope with those large errors. A common compensation is to enlarge the tail end of an otherwise Gaussian distribution. [70] assumes a probability of ϵ_0 that an observation is an outlier, and adapts equation 7.2

as follows:

$$\alpha_i = \nu \alpha'_i \left((1-\epsilon_0) \frac{1}{\sqrt{(2\pi)^k \, |P|}} e^{-\frac{1}{2}(z_t - \widehat{z}_i)^T P^{-1} (z_t - \widehat{z}_i)} + \epsilon_0 \right) \tag{7.3}$$

This change prevents the weighting of Gaussians to drop too much by an update with a single outlier. While this approach seems intuitive and is often applied to the weighting functions in particle filters, it does not affect the actual Kalman update with the outlier observation itself. Instead the implicit assumption is that outliers appear randomly and therefore do not systematically influence the system estimation.

An alternative handling of such false positives is proposed here for two reasons. First, false positive observations are not necessarily random in nature. Depending on environment and sensor setup, it sometimes is likely that some characteristic in the environment produces outliers in the robot's cognition process. Once those outliers originate from a certain source, which is not included in the robot's internal map of its environment, those observations cannot be handled as random noise any longer, since they are systematic. Repeated updates with those "false" observations would introduce a bias into the estimation process. The second reason is that the sensor update step described in section 7.1.3 already provides the means for an alternative solution.

Instead of incorporating the possibility for false positive observations into each sensor update for each correspondence, a more natural alternative is to handle false positives as just another correspondence alternative. The underlying assumption is that each landmark observation, not just the inherently ambiguous ones, either corresponds to one of the known locations of such landmarks on the map or to another source not included in the map. In SLAM contexts, these additional observation origins might be mapped and used for further localization purposes, resulting in a multiple-hypothesis EKF-SLAM approach. In localization tasks, however, such observations corresponding to unmapped landmarks can simply be discarded, i.e. no update is performed at all to the term generated by this correspondence choice.

This approach provides the possibility of a position tracking unbiased by false positives, thus more robust especially in situations where false positive observations do not occur randomly[2]. Obviously, this also has drawbacks: Whereas previously all terms in the Gaussian mixture have been subject to the same number of updates, this is not the case now, leaving the choice either to use a fixed factor ϵ_0 of equation 7.3 for the weight update or to fundamentally change the working of the weight update. At the same

[2]The term robustness describes a filtering method's ability to cope with erroneous input data. This includes ordinary Gaussian measurement noise, but also larger errors such as corrupted data or false positive measurements. The latter ones are focused here because of their frequent occurrence in robotic perception.

time, the increase in the number of correspondence choices also potentially increases the growth rate of terms in the Gaussian mixture.

7.3 Selective Multiple-Hypotheses Filtering

Sections 7.1 and 7.2 provide an overview about Gaussian mixture filtering and important aspects and approximations, and also established the view that due to linearization and aggressive pruning, Kalman filters might tend to behave as maximum likelihood estimators. In this chapter the maximum likelihood character is accepted and combined with a different trade-off between pruning and state space coverage to propose an alternative approach to multiple-hypotheses Kalman filtering which utilizes some techniques originating from particle filtering.

7.3.1 Similarities to Particle Filtering

Particle filtering, or Monte Carlo localization, is an alternative implementation of the Bayes filter equations 3.11 and 3.16. Instead of expressing the belief in parametrized form by one or several Gaussians, $bel(x_t)$ is given as a set of samples or particles whose distribution in state space is proportional to the belief's probability distribution. As mentioned in the beginning of this chapter, particle filters are often the favored choice for applications in which state estimations are expected to be multi-modal and measurements subject to significant noise.

Many similarities between multiple-hypotheses Kalman filtering and particle filtering exist, since each particle in itself represents a hypothesis for the robot's position. Both algorithms have originally been designed under the assumption of enough hypotheses to appropriately cover the state space. But the computational complexity of both increases for a growing number of hypotheses (linear in the particle filter; even quadratic for model merging in Gaussian mixture filters). In implementations aiming at real-time performance on limited computational resources, the number of hypotheses is often reduced to a point where those assumptions are violated. For particle filters with extremely low numbers of particles, several techniques have been proposed to compensate for the low state space coverage (cp. section 3.4). Applicability of some of those techniques to multiple-hypothesis Kalman filtering is motivated by pointing out the following similarities.

The most obvious parallel in the actual implementation is the weighting of particles and Gaussian terms, which is a straight forward application of the sensor model $p(z_t|x_t)$ and therefore identical in the general implementation of equation 7.2. In the basic particle filter the new particle set for each time step is sampled from the old one proportional to the particle weights, thus requiring a resetting of the weights. In implementations aiming at real-time performance on limited computational resources, techniques have been introduced which resample only when necessary, thus updating the weight exactly as in equation 7.2. In this context, adaptations similar to equation 7.3 have been proposed to accommodate outliers. An even more drastic restriction of the weight update has been proposed in [65] as *temporal smoothing* where the weight update is truncated to prevent too high fluctuations of particle weights. The same problem has been pointed out by Quinlan and Middleton in [70] for multi-hypotheses Kalman filters.

Particle filters with extremely low numbers of particles cannot maintain hypotheses in areas of (temporarily) low probability, therefore an increase of probability due to recent measurements cannot be modeled. Introducing new particles randomly in the resampling step allows the possibility of high weighted particles away from previous high probability regions. A more effective solution in the form of *sensor resetting* has been proposed in [57]. Sensor resetting describes the introduction of new particles based on recent measurements and the sensor model. Instead of adding many new particles uniformly distributed and discarding most of them in the resampling step after a few measurement updates, particles are inserted directly in regions of potential high probability based on the recent measurements. In [65] for example, additional particles are generated with high priority from multi-angulation of unique landmark observations calculated as in [4], and with lower priority as multiple hypotheses from non-unique observations. This technique is directly applicable in multiple-model Kalman filtering, too, as motivated in the following section.

7.3.2 Stochastic Context of Maximum-Likelihood Choices for Multiple-Model Filtering

As argued in section 7.2.1, operating Gaussian mixture filters with a strictly low limit on the number of terms and the consequential aggressive pruning deprives such filters of much of their multiple-hypothesis tracking potential. The parallel to particles filters with low numbers of particles has been pointed out in the previous section. The established policy in this case consists of two measures. The first one is to limit the influence of single inconclusive measurements. The second is to accept the impossibility to track all important hypotheses in the exponentially growing number of paths, while at the same time providing the means to recover from the neglect to model those which might rise in importance again in the future, e.g. by sensor resetting.

This works contribution is to transfer these concepts from particle filtering to Gaussian mixture Kalman filters. This mainly means a modification of the sensor update step. First, it is obvious that the weight update needs to be adjusted. But more importantly, an additional new sensor update in parallel to the old one will perform the same function as the sensor resetting part of the particle filter's resampling step, i.e. introducing new Gaussian terms not based on the previous state estimate, but on the recent sensor measurements' sensor model only. Note that this is not equivalent to model splitting since the new terms are not correlated to the old ones.

The resulting filter frequently injects new low-weighted models into regions with high probability based on the last observations, which might be expected to rise in weight in case theses hypotheses also fit future observations. This however relieves the necessity to track multiple paths of correspondence choices for updates of existing models as described in section 7.1.3 and illustrated in figure 7.2. In [70] many multi-modal sensor updates basically result in uni-modal ones due to the aggressive pruning. While this already corresponds to implicit maximum-likelihood updates, it is now possible to explicitly do exactly this. As a consequence, both the process update and the old sensor update can be applied with the uni-modal maximum-likelihood choice, which can be done in far less computation time. This explicitly neglects alternative paths in the decision tree of correspondence choices, but relies on the sensor resetting functionality to pick up those paths which lead to conclusive estimates as illustrated in figure 7.3 relating to the example in section 7.2.1.

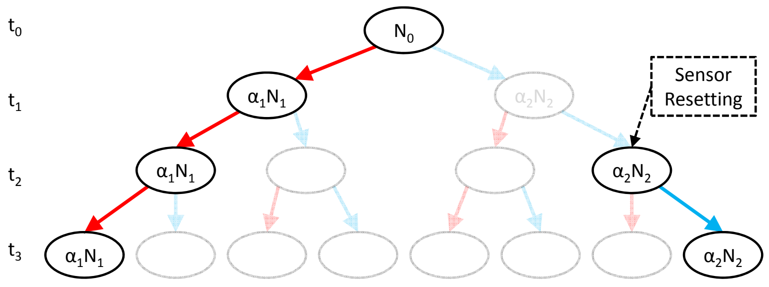


Figure 7.3: Tracking only maximum-likelihood correspondence choices for ambiguous landmarks, relying on sensor resetting to generate terms close to those corresponding to the neglected paths leading to conclusive estimates.

This application of the sensor resetting concept also solves the problem that common Kalman implementations have concerning the kidnapped robot problem. Both, a sudden change of robot orientation with consequential wrong data association as described in the example in section 7.2.1, and a real teleportation event will be handled accordingly.

7.3.3 Modified Sensor Update

Section 7.2.2 motivated the advantage of handling the possibility of false positive observations in the correspondence choice instead of incorporating this possibility into every separate likelihood calculation. In the context of maximum-likelihood correspondence choice sensor updates, this means the possibility to directly discard measurements for a certain localization hypothesis, if the likelihood of this observations being a false positive is higher than it belonging to any landmark occurrence in the map.

This allows the possibility that at least one model estimates the correct current state without any bias even by frequent false positive observations. However, one problem occurs which might be an explanation why this approach has not been used so far: Enabling false positive correspondence choices automatically leads to terms with different numbers of sensor updates each time a measurement is associated with a landmark by some terms and with a false positive by others. Consequently, updating the weights with equation 7.2 as described in section 7.1.3 is not directly possible any more. One option would be to assign a fixed likelihood to false positives and use that for those updates, but this would leave a balancing problem between the weighting of hypotheses which explain only a subset of observation, but really accurately, and those which explain more observations, but not as conclusively.

Following the temporal smoothing idea for particle filters [65] described in section 7.3.1, the weight update for hypotheses can be adjusted so as not to be influenced that much by different degrees of outlier measurements. Instead it is possible to weight different models based exclusively on how many observations can be conclusively explained by them.

7.4 Evaluation

To show the applicability and practical relevance, the proposed multiple-hypotheses Kalman filter is applied to the RoboCup scenario (cp. section 2.1). The walking algorithms presented in section 4.2.1, and more detailed in [13, 14], can be tuned to reach a speed of up to 44 cm/s, which has been achieved during the RoboCup 2010, but are commonly used in a configuration with a maximum speed of 30 cm/s due to reasons of motor temperature management. While those relatively high speeds are advantageous in many game situations, chasing the ball and focusing on it for several seconds at a time allows the build-up of significant odometry errors without the possibility for continuous corrective measurements. High uncertainty in the odometry information can be considered a common characteristic of humanoid robots and biped walking in general and needs to be considered when designing and evaluating appropriate localization filters.

Most measurements are ambiguous: Observing a single goal post leaves at least two correspondence choices, while a field line crossing can be associated with 6 true positions on the field in case of a T-crossing or with 8 positions in case of an L-crossing. Even more correspondences are possible when allowing incomplete or uncertain classification of crossings (which is done here), e.g. in case of occlusion or for the observation of two perpendicular lines whose intersection is outside of the image. Also the center circle, which is a unique observation, implies two possible positions due to its symmetry. As mentioned in section 2.1.1, one important feature of the SPL environment is that no barrier exists between the soccer field and its surroundings, which frequently includes colorfully clothed audience. So a localization algorithm needs to be robust not only to noisy measurements due to staggering robots but also to frequent false positives. This is the setup in which the proposed multiple-hypotheses Kalman filter is evaluated.

The software running on the robot includes the manufacturer's middleware NaoQi as well as the robot control software consisting of a motion thread running at 100 Hz and a cognition thread running at 15 Hz in this experiment. All modules needed for playing a regular SPL soccer match are activated. To provide ground truth data for the evaluation of the localization quality, a camera system is mounted above the field and detects markers attached to the robot. An additional module in the robot's software uses this ground truth for comparison with the localization filter output.

The presented approach is evaluated in experiments on real robots in a typical SPL scenario. Additionally, it is compared against a particle filter solution utilizing sensor resetting, temporal smoothing, and lazy resampling, and which has been in use in RoboCup competitions up to the development of the presented multiple-hypotheses Kalman filter. For all experiments, both localization algorithms run in parallel on the robot, thus working with the exact same input from image processing and ensuring the comparability of the results.

A first experiment evaluates the re-localization ability after several teleportation events. In figure 7.4, each red mark on the time axis indicates

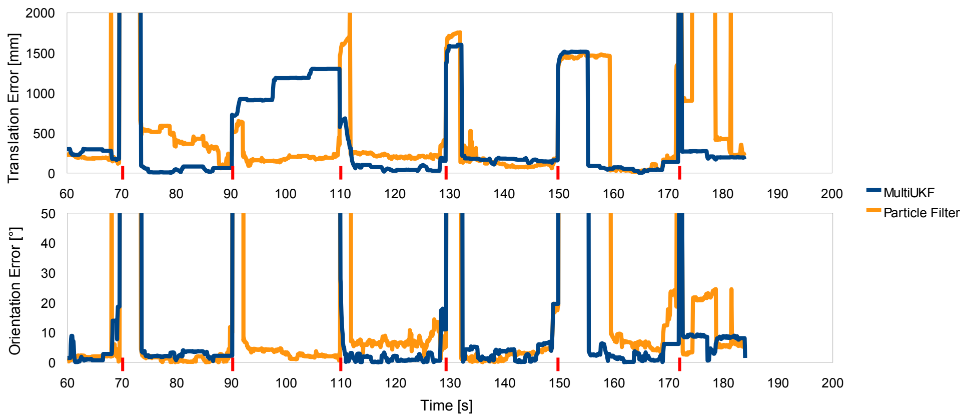


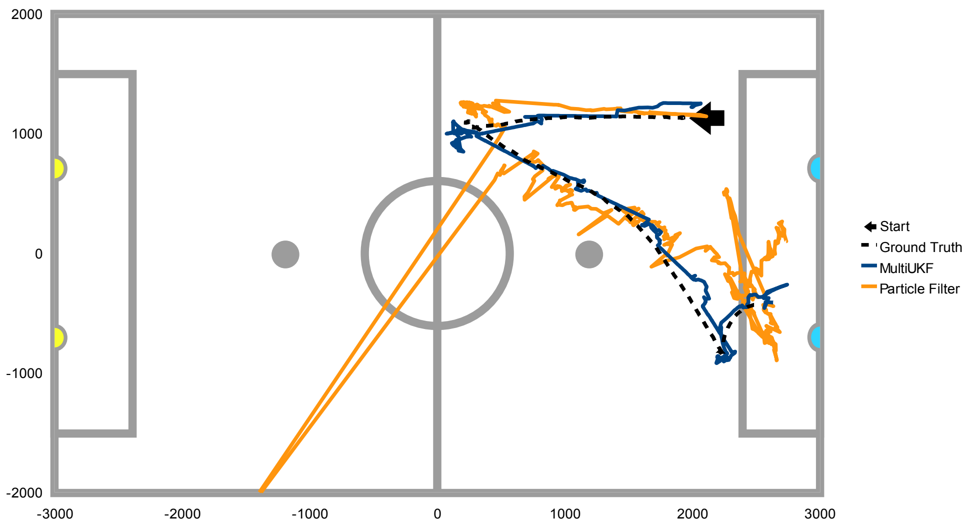
Figure 7.4: Ability for re-localization from kidnapped-robot situations.

that the robot has been picked up to be placed at a random new position on the field. The illustrated performance represents typical behavior for both filters, which expectedly show similar behavior since both get their re-localization ability from the same principles. At timestamp 70, both filters re-localize at the exact same time, because their sensor-resetting submodules utilize the same observations. Between timestamps 90 and 110 the multiple-hypotheses Kalman filter generates a hypothesis close to the correct position, but does not rate it high enough in those 20 seconds to output it as the likeliest location. Similarly the particle filter jumps between different clusters after timestamp 170. Based on this experiment, none of the methods is superior to the other in this aspect, but it shows that the proposed filter is able to perform at the same level as a particle filter implementation, which has been tuned especially to handle such kidnapped-robot situations. Note that this is classically regarded as a particle filter's specialty and similar performance has not been reported for related Kalman variants.

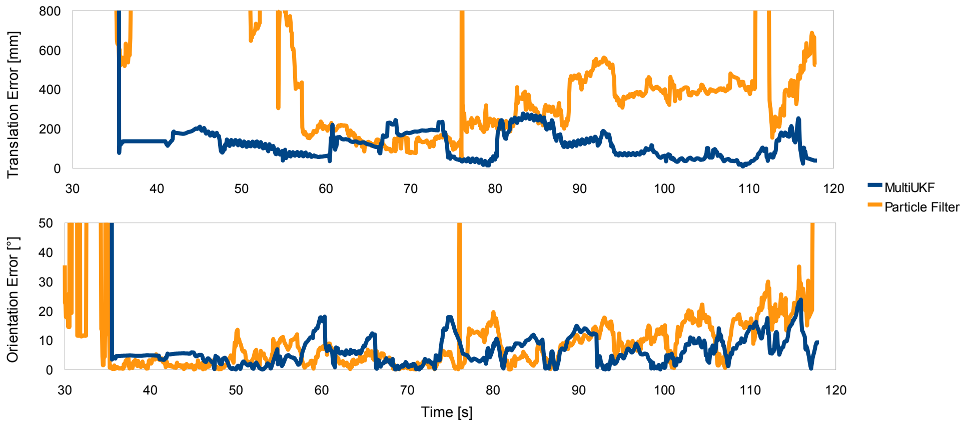
Additional experiments have been set up to quantitatively evaluate the localization quality. In all experiments the robot is placed on the field without prior knowledge of its position and the movement is started after a fixed time which has normally been enough to establish a good first localization hypothesis. Figures 7.5 and 7.6 show different ground truth paths on the field and the localizations results of the multiple-hypotheses unscented Kalman filter and the particle filter. The Kalman filter's characteristic of smooth and accurate position tracking is clearly visible and the filter's output in form of its strongest hypothesis is superior to the particle filter's most probable cluster.

Figure 7.7 shows one run where many persons walked around the field, thus occluding field features and provoking false positives at the same time, e.g. blue jeans sometimes are falsely recognized as blue goal posts. While both localization approaches show diminished results, the multi-hypotheses UKF clearly produces estimations closer to the real robot path.

A comparison of the different algorithms' runtime is given in figure 7.8. It has to be noted that the presented measurement is not unbiased, since larger runtimes tend to be influenced more by random threading issues and thus the larger module update time of the particle filter might include motion thread update cycles. Nevertheless, the multiple-hypotheses Kalman filter is clearly much more efficient. While this does not allow to compare this implementation of a Gaussian mixture Kalman filter to the one proposed by Quinlan and Middleton [70], it can be argued to be more efficient, too, since a similar comparison to a state of the art particle filter in [70] showed only slightly better runtime and averaged around a third of the image processing time, which would still be in the range of multiple milliseconds. The average runtime of the approach presented here is 0.4 ms. Figure 7.9 illustrates the runtime measurement of the cognition thread in the robot control framework. As can be seen, the presented approach obviously eliminates the localization problem as a computational bottleneck. Most of the time is spend on image processing, the remaining time difference goes to infrastructure and other tasks such as ball tracking or behavior decisions. The periodic tendencies in the measurements (which can also be seen in figure 7.8) are a

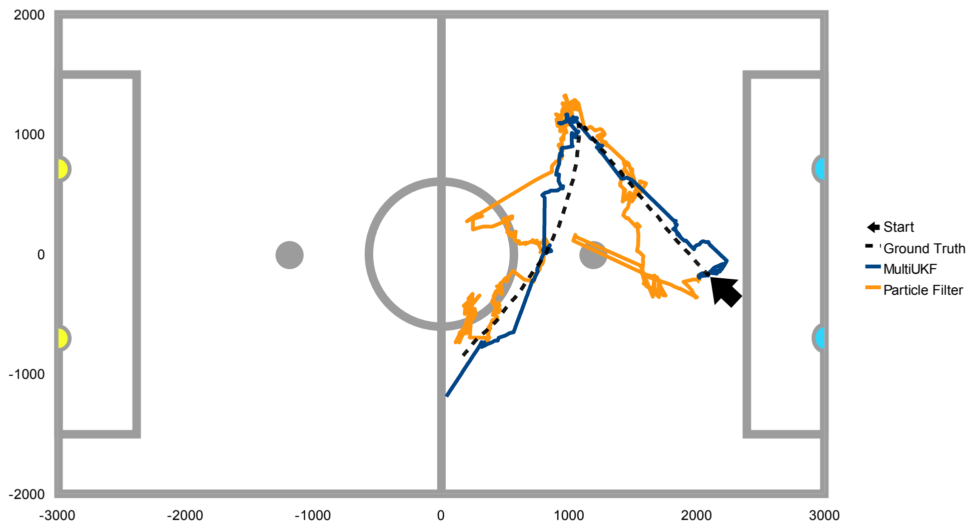


(a) Estimated positions and ground truth on the field.

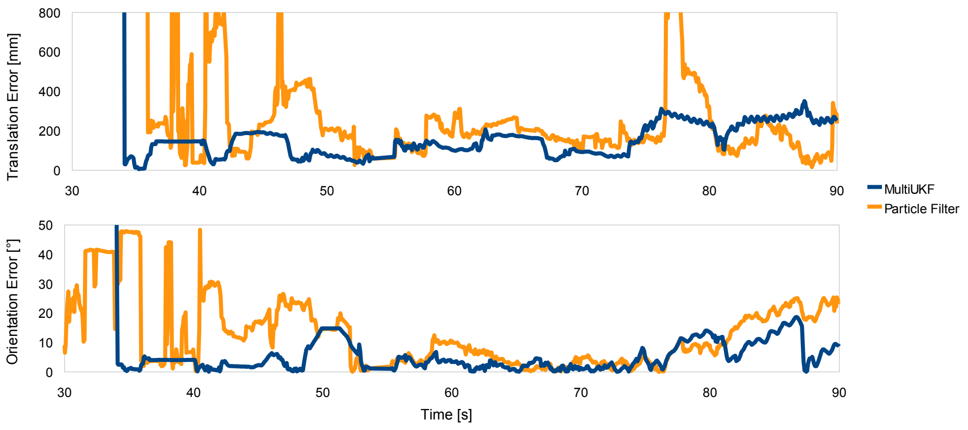


(b) Errors in pose estimation.

Figure 7.5: Localization of Multiple-Hypotheses UKF compared to previous particle filter solution (which was used in previous competitions). Both are running in parallel on the Nao using the same perception as input. Ground truth is provided by a camera mounted above the field.

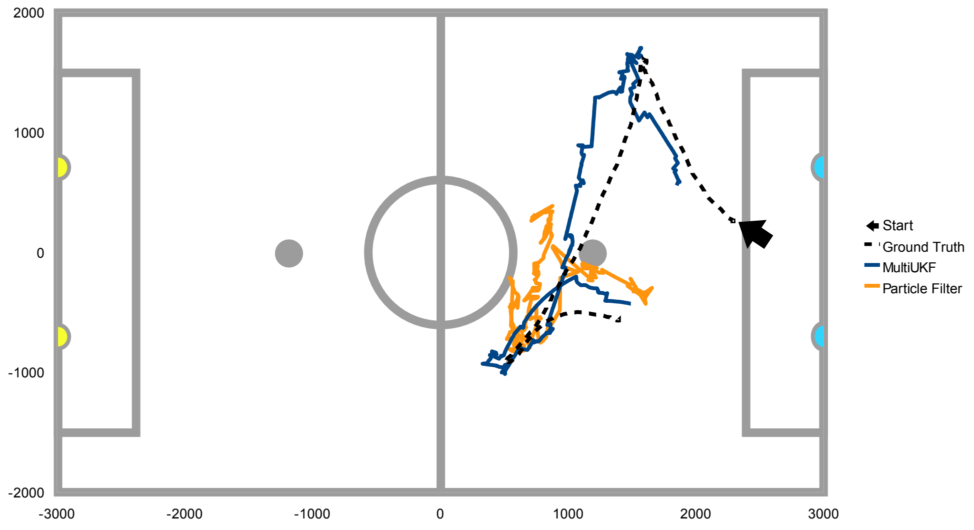


(a) Estimated positions and ground truth on the field.

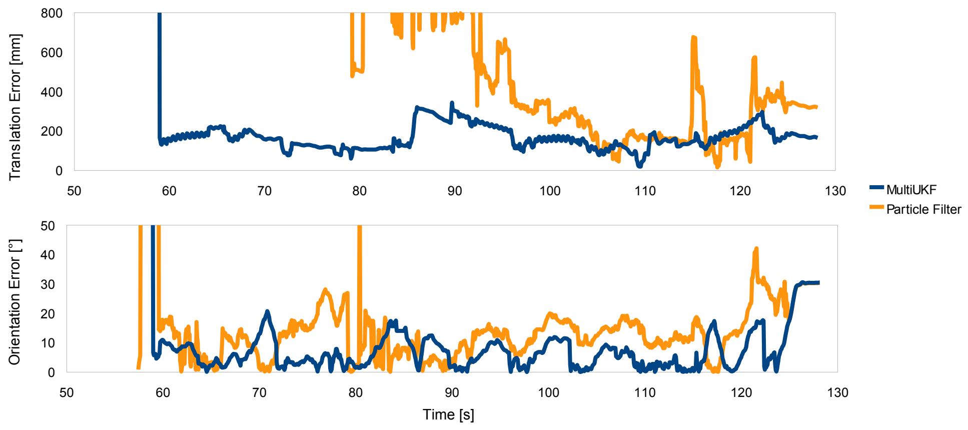


(b) Errors in pose estimation.

Figure 7.6: Comparison of localization performance.



(a) Estimated positions and ground truth on the field.



(b) Errors in pose estimation.

Figure 7.7: Comparison of localization performance in a worst case situation with much occlusion and many false positive observations.

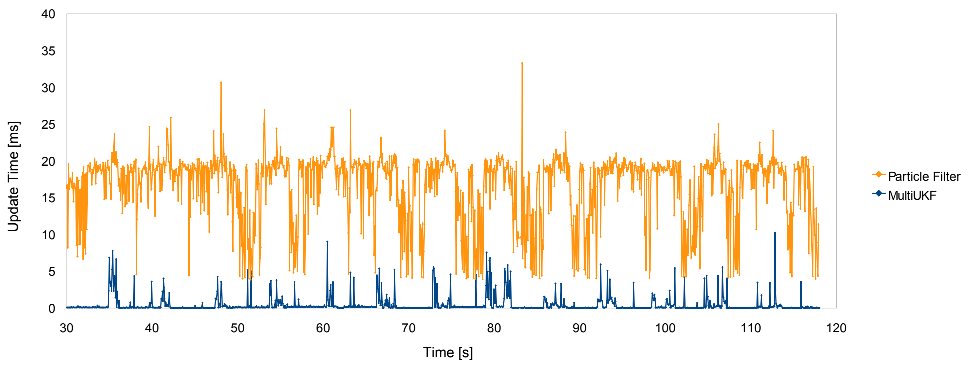


Figure 7.8: Runtime comparison between the multiple-hypotheses Kalman filter and the particle filter.

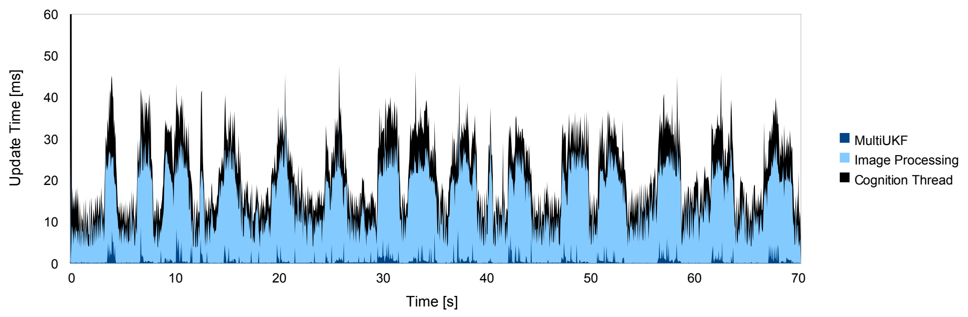


Figure 7.9: Overall cognition thread runtime in a configuration without the particle filter.

result of the robot's head motion, which also searches for the ball in front of the robot's feet where only little localization information can be extracted.

7.5 Conclusion

This chapter presents an approach for Gaussian mixture filtering which utilizes techniques from particle filtering to incorporate valuable aspects of both filter strategies. The resulting multiple-hypotheses unscented Kalman filter is superior to common Kalman filters in its ability of fast re-localization in kidnapped robot scenarios and its representation of multi-modal belief distributions, and it outperforms state of the art particle filters in localization accuracy and computational efficiency. A direct comparison to the approach proposed in [70] or similar classical Gaussian mixture filters could not be done, yet, as its implementation has not been available. Besides performing extremely well in various experiments, the presented localization approach has been applied in RoboCup tournaments and its reliability, robustness and runtime-efficiency have played an important role in achieving a second place in the world championship 2011.

Chapter 8

Distributed Agents in Dynamic Environments

[1]So far, this work covered localization problems of single mobile devices in mostly static environments. In chapter 6, the environment is not previously known per se, but after an initial mapping phase, no new information about the environment is accumulated and used during the localization phase. For the purpose of localization, those maps are therefore also static a-priori knowledge.

In this chapter, the estimation task is extended beyond static environments and single robots. The robot soccer scenario described in section 2.1 is encompassed as a benchmark and challenge for stochastic state estimation and robotics in general. Thus, multiple robots in a team are not only used for coordinated behavior planning, but also their perception and estimation of the dynamic environment and their position therein is done cooperatively. Certain considerations are made in section 8.1 towards the tasks involved in cooperatively modeling a dynamic environment. Sections 8.2 and 8.3 then present different solutions for the stated problem, based on two different existing filter solutions, a particle (cp. section 3.4) and a Kalman filter (cp. section 3.3 and chapter 7), and their respective SLAM extensions (cp. section 3.5).

8.1 Considerations Towards Dynamic Environments

Knowing their own position relative to the environment and certain points and objects of interest is essential for mobile robots to autonomously achieve

[1]Parts of this chapter are based on two articles, which are published in the Proceedings of RoboCup 2012: Robot Soccer World Cup XVI and in the Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems 2010, respectively. The corresponding publications, see [85] and [17], are available at `http://www.springerlink.com` and `http://ieeexplore.ieee.org`.

their tasks. The task of localization in a perfectly known structured environment alone has been discussed in previous chapters. The other extreme is simultaneous localization and mapping (see section 3.5), where no prior information is available and a robot has to build a map while at the same time navigating and localizing itself using this map. Real-world robotics applications can be placed between simple localization and SLAM: Some information about a robot's environment is nearly always available, be it floor plans for indoor or aerial images for outdoor environments, even if those may be on another planet. This prior information however is rarely sufficient enough to neglect further information to be collected during the robot's operation. Floor plans miss static features like most furniture, quasi-static features like chairs, and of course dynamic features like people or other moving agents. The same holds for any outdoor prior information.

This chapter's purpose is to give an overview about localization in this context, tracking and aspects of distributed modeling among a team of robots.

8.1.1 Localization and World Modeling

Separating the state for localization and tracking decreases the estimation problem's dimensionality and therefore the complexity of algorithmic solutions like the Kalman or particle filter. This is the major reason that modeling dynamic elements of a robot's environment is normally approached by assuming a known location or working in a robot-centric local coordinate system. Additionally, tracking of dynamic objects by stationary observers is widely explored, which suggests approaching this problem for a moving observer in similar ways.

Modeling the full state of both the localization and the surrounding world, however, has advantages in certain situations, some of which are covered in more detail in section 8.1.5. In case of an unknown number of ambiguous objects, this problem bears some resemblance to SLAM. Most SLAM algorithms, however, deal with completely unknown, but static environments only. Previously the fields of localization, i.e. estimating a position based on a known map, and SLAM, i.e. no prior information but mapping of static environments only, seem to have been strictly separated. Only recently approaches were published covering SLAM with a-priori information. In [52] the prior information based on aerial images is used for localization but not fully integrated into the SLAM aspect of this work. Only single unimodal position hypotheses are considered as constraints for the graph SLAM algorithm, despite the multi-modal nature of the particle filter used to generate these. Thus [52] is closely related to [62] where GPS is used in the same way instead of localization by prior information.

In most SLAM algorithms, moving objects are not explicitly mapped but filtered out beforehand and considered noise [34]. Alternatively, moving objects are tracked in separate models without influence on the SLAM outcome as in [76].

8.1.2 Heterogeneous Information Sources

The proposed system should use information about previously known and previously unknown, static and dynamic features, and incorporate all those into a coherent estimate of both the robot's own positions and the potentially dynamic states of the other objects. This implies various different, and more importantly, heterogeneous information sources.

Distinctions can be made according to the characteristics of each feature: whether it can be used for localization directly or needs to be mapped, too, either as a static but previously unknown feature, or a dynamic one including motion updates. A feature's associated uncertainty can vary both with respect to the reliability and precision of its observation as well as the inherent predictability of its motion model. Simple in-animated objects on the one hand may just follow physical equations of motion. Other autonomous agents on the other hand may change their intention and action unpredictably, while being harder to measure reliably due to their more complex shape, varying silhouette and changing backgrounds.

The important consequence of these considerations is that each such distinction offers the possibility to apply approximations without losing too much precision in the estimation result. A thorough analysis on the implication for those heterogeneous information sources and possible approximations can be found in sections 8.2 and 8.3, where the applicable ones are described in each respective implementation's context. One relevant approximation shared in both solutions is the aggregation of measurements to build local short-term models of each observation type, which is described in the following section.

8.1.3 Percept Aggregation in Local Temporary Models

In case of highly uncertain observations of dynamic objects or agents in the environment, it is useful to aggregate these measurements to build local short-term models of each observation type, thereby decreasing the uncertainty associated with the observed target and allowing to filter false positives. Once such a short-term model is sufficiently recognized, it can be forwarded to the central estimation system as a meta-measurement and deleted from the temporary local model. The deletion of such models is important to preserve the independence assumption between consecutive measurements which is part of the Bayes filter concept. Insufficiently validated local hypotheses can then be pruned away without having a negative influence on the system's estimate. The short life span of those local models, e.g. below one second, prevents odometry errors to accumulate, but often allows the integration for example of a series of image processing results to obtain superior measurement quality.

In the following, the local modeling is exemplified for dynamic features, i.e. the ball and other robots in a robot soccer scenario. Once they are recognized in the image processing module, measurements of robots or the ball are handled like other point features by only providing their ground contact point, so their measurements consist of two angles $z = (\alpha_1, \alpha_2)^T$ as described in section 5.1.1.3. Each object's model $\mu = (m_x, m_y, m_{v_x}, m_{v_y})^T$ consists of a 2-dimensional spacial component and a corresponding velocity component, which are denoted $\mu' = (m_x, m_y)^T$ and $v = (m_{v_x}, m_{v_y})^T$ when used separately. For the spacial component of the state μ' and the height of the robot's camera h_{camera}, equations 8.1 and 8.2 can be used to calculate the sensor model in form of the observation matrix H as in equation 8.3[2]. Note that the velocity is not observable by processing a single camera image.

$$h_{\alpha_1}(\mu) = \operatorname{atan2}(h_{camera}, |\mu'|) \tag{8.1}$$

$$h_{\alpha_2}(\mu) = \operatorname{atan2}(m_y, m_x) \tag{8.2}$$

$$H = \begin{pmatrix} -\frac{h_{camera}\ m_x}{|\mu'|^3 + h_{camera}^2\ |\mu'|} & -\frac{h_{camera}\ m_y}{|\mu'|^3 + h_{camera}^2\ |\mu'|} & 0 & 0 \\ -\frac{m_y}{m_x^2 + m_y^2} & \frac{m_x}{m_x^2 + m_y^2} & 0 & 0 \end{pmatrix} \tag{8.3}$$

For objects which are simply governed by the physical laws of motion, instead of being motorized or controlled, the motion model in the control update for a time difference Δ_t consists of a continuous motion slowed down by a friction factor ϕ as the force generated by the friction divided by the mass of the object. ϕ is negative to indicate the decelerating force. Since the state is modeled in local coordinates, the robot's own motion, given by the translational and rotational odometry $(\Delta_x, \Delta_y, \Delta_\theta)$, also transforms the local estimate. This results in the following time update for the velocity vector:

$$v_t = V\ v_{t-1} \tag{8.4}$$

with

$$V = \begin{cases} \left(1 + \frac{\phi\ \Delta_t}{|v_{t-1}|}\right) \Omega(-\Delta_\theta) & \text{for } |v_{t-1}| \geq |\phi \Delta_t| \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & \text{else}, \end{cases} \tag{8.5}$$

where $\Omega(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$ is the rotation around α (cp. equation 5.1 in chapter 5). The full time update therefore predicts the state μ_{t-1} according to equation 8.6.

[2] A detailed derivation can be found in appendix C.

$$\overline{\mu}_t = f(\mu_{t-1}) = \begin{pmatrix} \Omega(-\Delta_\theta) & \Delta_t \Omega(-\Delta_\theta) \\ 0 & V \end{pmatrix} \mu_{t-1} - \begin{pmatrix} \Delta_x \\ \Delta_y \\ 0 \\ 0 \end{pmatrix} \tag{8.6}$$

With the derivation given in appendix C, this results in the Jacobi matrix F for the process update as the partial derivatives of m_x,m_y,m_{v_x}, and m_{v_y} at $(m_{x,t-1}, m_{y,t-1}, m_{v_x,t-1}, m_{v_y,t-1})^T$:

$$F = \begin{pmatrix} \Omega(-\Delta_\theta) & \Delta_t \Omega(-\Delta_\theta) \\ 0 & F' \end{pmatrix} \tag{8.7}$$

$$F' = \begin{cases} \begin{pmatrix} \frac{\partial f_{m_{v_x}}}{\partial m_{v_x}} & \frac{\partial f_{m_{v_x}}}{\partial m_{v_y}} \\ \frac{\partial f_{m_{v_y}}}{\partial m_{v_x}} & \frac{\partial f_{m_{v_y}}}{\partial m_{v_y}} \end{pmatrix} & \text{if } |v_{t-1}| \geq |k\Delta_t| \\ 0 & \text{else} \end{cases} \tag{8.8}$$

with

$$\frac{\partial f_{m_{v_x}}}{\partial m_{v_x}} = \left(1 + \frac{\phi\ \Delta_t}{|v|}\right) \cos(-\Delta_\theta) - \frac{\phi\ \Delta_t\ m_{v_x} \left(\cos(-\Delta_\theta) m_{v_x} - \sin(-\Delta_\theta) m_{v_y}\right)}{|v|^3} \tag{8.9}$$

$$\frac{\partial f_{m_{v_x}}}{\partial m_{v_y}} = -\left(1 + \frac{\phi\ \Delta_t}{|v|}\right) \sin(-\Delta_\theta) - \frac{\phi\ \Delta_t\ m_{v_y} \left(\cos(-\Delta_\theta) m_{v_x} - \sin(-\Delta_\theta) m_{v_y}\right)}{|v|^3} \tag{8.10}$$

$$\frac{\partial f_{m_{v_y}}}{\partial m_{v_x}} = \left(1 + \frac{\phi\ \Delta_t}{|v|}\right) \sin(-\Delta_\theta) - \frac{\phi\ \Delta_t\ m_{v_x} \left(\sin(-\Delta_\theta) m_{v_x} + \cos(-\Delta_\theta) m_{v_y}\right)}{|v|^3} \tag{8.11}$$

$$\frac{\partial f_{m_{v_y}}}{\partial m_{v_y}} = \left(1 + \frac{\phi\ \Delta_t}{|v|}\right) \cos(-\Delta_\theta) - \frac{\phi\ \Delta_t\ m_{v_y} \left(\sin(-\Delta_\theta) m_{v_x} + \cos(-\Delta_\theta) m_{v_y}\right)}{|v|^3}. \tag{8.12}$$

Thus, local models of dynamic objects in the robot's environment can be modeled using separate Kalman filters. In case of the unpredictability of the motion of autonomous robots, it is possible to neglect the estimation of their velocity and apply high process noise instead.

8.1.4 Distributed Modeling

The sharing of information among a team of autonomous agents is especially desirable in cases where single robots have a very limited field of view and when occlusion frequently occurs. The distribution of information can be done with two conceptually different approaches. One approach can be classified as bottom-up and distributes measurements between robots, which are subsequently handled by common sensor fusion techniques. This is for example done in [83] and [20]. The top-down approach as applied in [39] and [96] consists of merging the individual robots' world maps. The prerequisite for such map merging is that all poses of participating robots need to be known, either in a consistent global coordinate frame or relative to each other. A common implementation in exploration scenarios is with uniquely identifiable robots which initiate map merging when observing each other, or when all robots are confidently localized in the global reference frame.

This latter approach would exclude poorly localized robots from map merging. However, those might also profit from the shared information, even specifically to resolve their poor localization in case of symmetries. If the measurements are distributed among robots, basically only each sending robot needs to be localized successfully in a global reference frame. An additional advantage is the computational and architectural simplicity of observation distribution compared to map merging, especially if observations are already aggregated in temporary local models as described in section 8.1.2, which in case of reliable localization can be distributed at the same time when used for the local integration into the global world model. Note that this approach does not guarantee a globally consistent model among all robots, since insufficiently localized robots do not send out information and therefore integrate more (but potentially also more unreliable) knowledge. The difference between the models of well localized robots, however, can be minimized by globally scheduling the exchange and integration phases of the observation distribution, which will be described in section 8.2.2.

8.1.5 Estimating a Robot's Position and Dynamic Environment in a RoboCup Scenario

Although the algorithms and concepts described throughout this chapter are formulated as general as possible, and are applicable to a wide range of standard robotic applications, their motivation and evaluation are based on the RoboCup scenario. This section therefore states several requirements and objectives derived from this scenario and its hardware constraints. Those are referenced later as a guideline for the implementations of the approaches for simultaneous localization in a structured robot soccer environment and mapping of the unknown dynamic elements therein.

In this robot soccer scenario (cp. section 2.1.1), incomplete prior knowledge about the field is given by a map consisting of mostly ambiguous features. Global localization and re-localization (kidnapped-robot problem) is considered necessary. This and probable ambiguity of observed features suggests the necessity of a multi-modal belief state representation. Dynamic objects in the scene are mostly ambiguous, too, and assumed to be other autonomous agents, some but not all of which the robot may be in contact with. Due to noise in the perception processes and the possibility of false positive perceptions, the estimation process needs to be robust, with uncertainty explicitly included in the model. Runtime performance is essential, since the algorithm aims at real-time execution on limited embedded platforms.

As mentioned in section 8.1.1, the separate modeling of the robot's localization and the dynamic part of its environment is often chosen when performance is critical. However, this chapter evaluates the possibilities of unified modeling under those harsh performance constraints, as there are several benefits over separate modeling, which are not to be neglected. Modeling of dynamic objects might potentially be useful for localization. Unknown movement of those objects prevents accurate localization, but shared information among robots might still resolve a multi-modal or symmetrically ambiguous localization state. Additionally, robot-centric modeling needs the incorporation of the robot's control information or odometry measurements during the filter's time update. This information is often uncertain, imprecise motion execution or even unplanned contact with other elements of the environment. As described in section 4.2, this is a particular problem with humanoid robots. Thus, even when those separately modeled objects are stationary, they are prone to drift due to integration of the odometry errors, while this same error is already compensated in the localization's pose estimate.

8.2 Real-time Dynamic FastSLAM

[3] This section presents an implementation of the approach of localization and simultaneous tracking in a unified estimation state as motivated in section 8.1.1. In order to aim for an implementation efficient enough to run on embedded platforms with limited processing power such as the Nao (cp. section 2.1.2), the FastSLAM algorithm described in section 3.5.2 is chosen as a basis for a cooperative world modeling system. In the following, the basic concept as well as several adaptations for the optimal utilization of the considerations in section 8.1.2 are presented and evaluated.

8.2.1 FastSLAM Adaptation to Prior Knowledge and Heterogeneous Information Sources

FastSLAM is a popular SLAM algorithm for handling multi-modal distributions while keeping low complexity (cp. section 3.5.2). In FastSLAM, a particle filter estimates the robot's pose, but with each particle carrying its own map and thus representing not only a localization state but a hypothesis for the map and the robot's whole path therein. The central idea of FastSLAM is that by providing path hypotheses, the map features are independent of each other and can thus be estimated separately. The result is a particle filter with each particle possessing several separate Gaussians modeling the features which are not covered by the particle distribution alone. When observing a landmark, FastSLAM first updates the Gaussians of each particle analogous to the Kalman sensor update and then calculates the particles' weights for resampling from the innovation probability. Classic FastSLAM approaches the standard SLAM problem, i.e. simultaneous localization and mapping of static features without any prior information. Every feature that can be detected and used for localization is commonly mapped during the robot's operation and represented by its Gaussian.

In case of available prior information it follows naturally to divide the set of observable features into those used only for localization and those that also need to be mapped. In real-world scenarios the latter may also contain dynamic objects which are obviously not included in any a-priori map. Those of course need a full Kalman Filter including motion updates for estimation. Additionally, there might also be static map features as mentioned in section 8.1.5. Note that the same concept also applies to situations without prior information, if well established parts of a map are regarded as "fixed" and used instead.

So far perceptions are distinguished by whether or not they correspond to prior known features. Additionally, it is possible to identify different classes

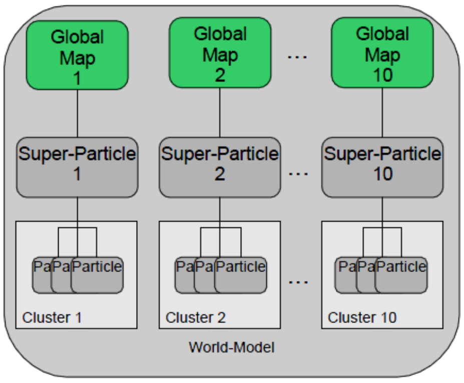
[3]Parts of this section are based on an article in the Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems 2010. The corresponding publication, see [17], is available at `http://ieeexplore.ieee.org`.

of objects to be modeled with respect to their uncertainty. Observations of dynamic elements of a robot's scene like people or other autonomous agents are commonly much less precise than those of static features. This is due to a number of reasons: The shape to be observed might move relative to varying backgrounds or might even change due to its own movement as is the case for people or walking robots. The resulting change is both in outline and appearance due to lighting and shadows. When tracking humans, the appearances to be recognized might also be generalized to cope for different clothing. For fast moving elements their velocity results in motion blur. Most images will not display the complete outline against the background. Instead, most perceptions will only be of parts of those agents, so the observed human's (or humanoid robot's) distance and position cannot be estimated with accuracy. Additionally to these reasons for uncertainty in their perception, other autonomous agents are also more difficult to model simply because they are, after all, autonomous and their behavior therefore unpredictable to some extent. Contrary to the mapping of static features, the position uncertainty grows when dynamic features are not seen and their constant observation just confines this uncertainty instead of collapsing it towards zero.

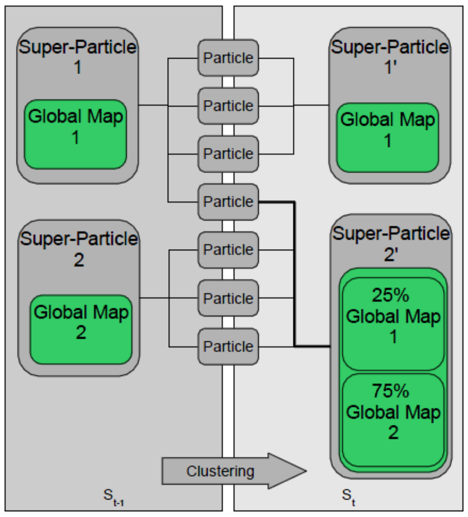
Tracking objects with an inherent limited precision due to their high process or observation uncertainty, while at the same time obtaining relatively precise models of other features, does not match the homogeneous modeling of all features. The result is a filter with many particles carrying the same models for those dynamic elements whose uncertainty often exceeds the localization uncertainty expressed by the distances between those particles. Multiple calculation of the same models is computationally inefficient. The novel approach proposed here is to identify such particles, to encapsulate them into super-particles, and consequently to estimate only one corresponding model for each dynamic element as illustrated in figure 8.1a. A super-particle's weight derives from the sum of its particles' weights. This approach can be seen as the particle filter's equivalent of using different resolutions for localization and mapping of certain elements.

Grouping of particles into super-particles is done using clustering methods analog to the identification of the region with highest density for extraction of a single pose from the particle filter for behavior decisions. In this case a deterministic grid clustering approach with overlapping grid cells is used[4], as shown in figure 8.2. Cells with the highest probability density are selected to generate super-particles as weighted averages. This is done using a greedy algorithm, which chooses the biggest cell, then marks all particles in this cell as assigned before finding the next biggest cell. This way, a particle can not be assigned to two clusters at once, but is always associated with the bigger cluster. This procedure is better suited than k-means clustering

[4]Overlapping cells are chosen to minimize the effects of discretization.



(a) Super particle.



(b) Map propagation during super particle generation.

Figure 8.1: Using super-particles to introduce a lower resolution scale.

for example, because the maximum distance of particles in the same cluster is limited, which is an essential characteristic as described above.

A new set of super-particles $\mathcal{S}_t$ is generated at each time step by considering only the current particle distribution. It is therefore possible for particles to transfer between close super-particles, so that transferring corresponding world maps from one time step to the next is not straight forward. Allocation of a super-particle's world map at time t to those of the previous time step $t-1$ is done by identifying corresponding particles. Exchanging of particles among close super-particles results in merging of the maps. This is illustrated in figure 8.1b, in which the new super-particle $2'$ receives particles from different super-particles in the previous time step, which are merged accordingly. Note that for simplification, figure 8.1b assumes identical weights

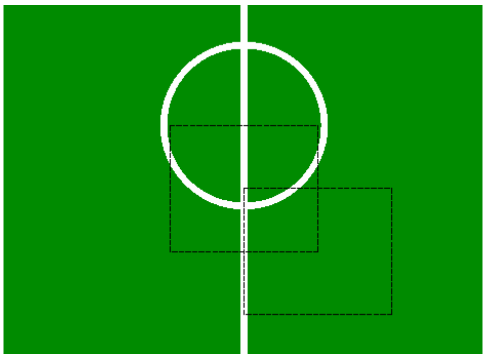


Figure 8.2: Two overlapping grid cells in the deterministic grid clustering approach.

for all particles.

This introduction of super-particles reduces the mapping cost by the factor between the number of particles and the number of super-particles. Note that the world map of the super-particle representing the highest probability density can directly be used as the filter's resulting maximum-likelihood estimation. Therefore the eventual merging of maps between super-particles is computationally far less demanding than the generation of one consistent map from all particles in the cluster with highest probability density in case each particle carries its own world map as in classic FastSLAM. This last step is avoided in most algorithms by directly using the single particle with the best current rating instead.

It remains to discuss the number of super-particles needed for a sufficient approximation of the current belief distribution. This depends, of course, on the task and application in general and on the specific belief state at a certain time, i.e. a highly regular and symmetric environment results in more clusters of high probability density, which survive longer than in environments where symmetric position hypothesis can be resolved easily. All such clusters need to be covered by super-particles. On the other hand, such environments also need a high number of particles for localization so that the gain by mapping through super-particles remains significant.

For uniform particle distributions, however, the number of super-particles needed to cover all those particles comes close to the number of particles. In this case, it stands to reason that those particles not covered by a certain number of super-particles correspond to regions of a belief distribution too low to have significant impact on the global map model.

Figure 8.3 shows the percentage of particles neglected by fixed numbers of super-particles for different situations in the scenario that is used for evaluation in section 8.2.3. Each situation starts with a uniform initial distribution and displays the distribution development over time. Figure 8.3a illustrates the normal case, i.e. when the robot is able to localize itself and the uniform distribution resolves to a few local maxima in the position probability density. In this case the number of super-particles needed to achieve good coverage is very low. Figure 8.3b on the other hand shows the worst case: the robot is placed in a way that no observation is conclusive enough to result in any significant clustering of the particle set and localization of the robot is not possible until it moves (or rotates) into a position where additional observations can resolve the uncertainty.

A second adaptation to heterogeneous information sources is possible via the update frequency. Since a single perception of a class of unreliable observations has far less impact than those of classes of highly reliable observations, it is possible to buffer those unreliable perceptions into one temporary model in local robot coordinates and update the super-particles' global maps with a lower frequency using this local model described in section 8.1.3. The sensor update step is identical to updating with single observations. The main argument besides lower computational cost is that multiple unreliable sensor readings generate an accumulated perception more reliable than the original ones, while single outlier observations can be filtered out. This accumulation, however, has to be only temporary with a limited timespan to prevent the continuous integration of odometry error into the model as discussed in section 8.1.1 and 8.1.5.

8.2.2 Multi-Robot Sensor Fusion

Due to the reasons mentioned in section 8.1.4, the bottom-up approach of distributing observations is used to implement the multi-robot sensor fusion. The percept buffering introduced in the section 8.1.3 and motivated above presents another advantage with respect to distributed modeling, since not only the update frequency of distributed percepts is lowered, but the decrease in bandwidth is also substantial. Temporary local maps can be distributed together with a robot's localization information in form of its super-particles' poses and covariances. Alternatively only the winning super-particle's pose and covariance can be sent, or sending might be omitted completely for localization states below a given validity threshold.

As can be seen in figure 8.4a, the super-particles' maps are updated with remote global observations analogous to the local ones. Additionally the received localization information allows not only to update but also to mark modeled dynamic objects with a robot's ID for future reference.

The distribution can be synchronized by calculating the clock offsets between all communicating robots from receptions times and packet time stamps, and picking the "oldest" clock as the reference time frame (see figure 8.4b). This is done to ensure a maximum level of consistency, as all robots incorporate approximately the same data into their world models, and thus decide based on the same knowledge. Complete consistency of maps on different robots cannot be guaranteed even if all robots send and receive simultaneously, however, due to possible different data association decisions resulting from propagation of pose uncertainty into communicated percepts. Synchronization is achieved by continuous estimation of the offsets between the robots' time frames and alignment of the distribution phases relative to a reference time frame, e.g. that one with the earliest starting time (cp. figure 8.4b).

8.2.3 Evaluation

This section evaluates the usefulness and performance of the proposed system in several experiments each focusing on different aspects. The experiments are conducted in a simulated environment of a RoboCup Standard Platform League scenario (see figure 8.5). Except for the approach presented above, the software running in each simulated robot is identical to that which was running on the real robots during the RoboCup world championship 2009 in Graz, Austria. The cognition process including image processing runs at 30 Hz and the motion at 50 Hz. The sensor data used for localization and modeling of the robots environment, in this case the ball and the other robots, is gathered by image processing, whereas the recognition of static features like field lines and their intersections or goals is more reliable than those of other robots because of the reasons mentioned in section 8.2.1. The simulated robot's odometry is adapted to show uncertainty similar to that measured in the movement of a real robot (cp. section 4.2.2).

The general experimental setup of the $7.4\,m \times 5.4\,m$ game area with 4 Nao robots is shown in section 8.5. The robots' movements are indicated by arrows of corresponding colors. The presented approach runs with 100 particles and a maximum of 10 super-particles. This low number of particles, which is sufficient for accurate localization while still ensuring computational efficiency, is achieved by several common techniques for particle filters such as sensor-resetting, temporal smoothing, and lazy resampling [65]. This scenario is based on common game situations: all robots are moving and no assumption about their motion models is made since behavior decisions and consequently walking directions can be subject to frequent change. So the localization mainly relies on static features and no significant difference in localization accuracy can be observed in this setup compared to a localization reference system with the same number of particles. The maximum number of super-particles is determined empirically to cover most of the particles even in worst case situations.

As the cognition process runs with 30 Hz the theoretical runtime available on the robot's limited CPU is 33 ms, but the motion control process running in parallel at 50 Hz further restricts the available time. The presented algorithm has an average runtime of approximately 20 ms on the target platform, which violates the timing constraints in the current configuration[5] but still puts the algorithm in the range where an application even on such a limited platform becomes possible. This, the possibility to run several solutions in parallel for evaluation without timing issues, and the repeatability of the experiments are the main reasons for executing the experiments in simulation.

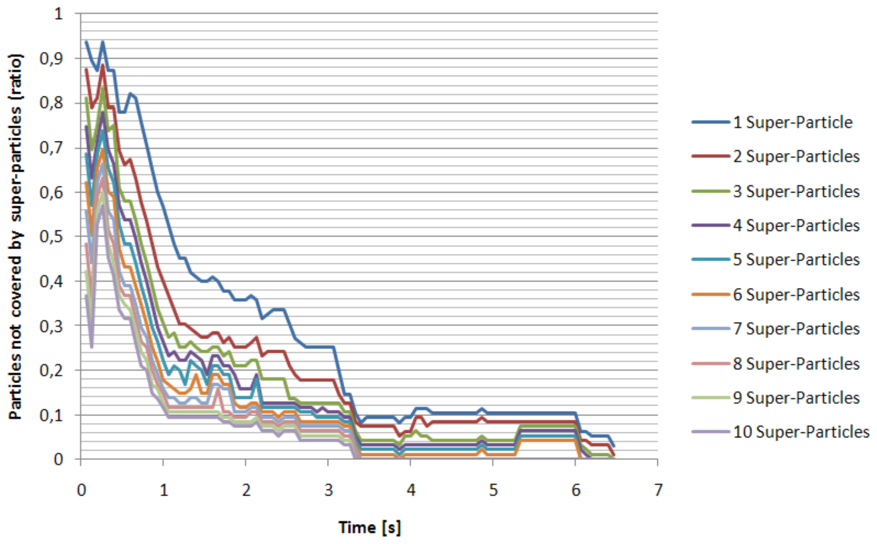
The first aspect to be evaluated is the potential gain by an unified modeling of a robot's localization and its dynamic environment compared to the common approach of tracking the dynamic elements separately in a local robot-centric model. The resulting modeling errors can be seen in figure 8.6. Both algorithms run in parallel on the same robot using the same perception and there is no team communication during this experiment. As can be seen in figure 8.6a, there is no advantage for robots that are observed frequently. This is expected since frequent observation prevents potential drift introduced by integrating odometry errors over longer time periods. For other robots however this drift has significant impact and can be compensated only to some degree by infrequent observations as can be seen in figure 8.6b. The gain of unified modeling is clearly visible.

The information gain of the multi-robot sensor fusion, compared to using local perception only, is evaluated using the same setup repeated with and without communication between the robots in the same team. In average the number of robots not included in the reference robot's map decreases significantly from 1.2 to 0.3 while the number of redundant or false positive mappings increases only slightly from 0.16 to 0.25. The increase in mapping accuracy for multiple robots observing another robot is shown in figure 8.7.

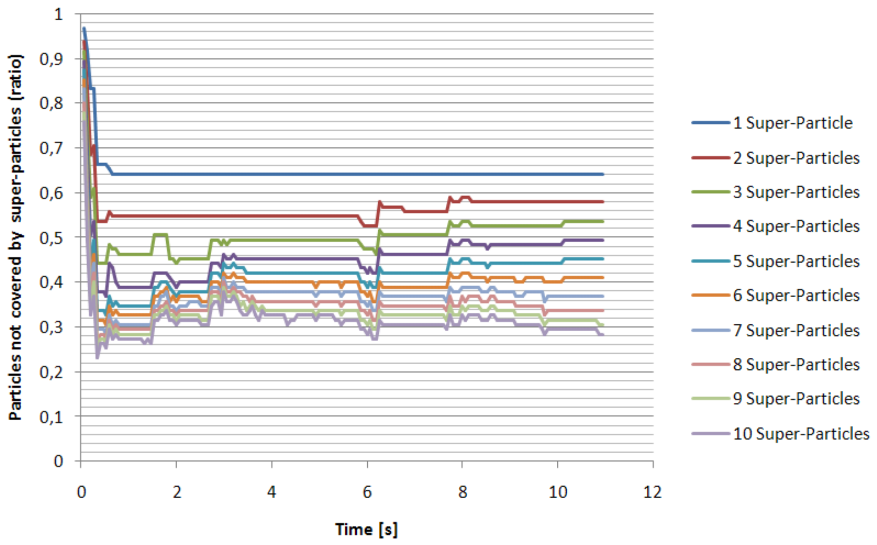
A final experiment is set up to prove the preservation of the SLAM functionality of the presented algorithm. For this the unchanged algorithm is tested in an environment where all the static features that can be seen in figure 8.5 are removed, so the image processing only produces percepts for robots and occasionally some field line false positives when parts of robots, e.g. extracted arms, are misinterpreted. The reference localization system only relies on the robot's uncertain odometry and is distracted further by those false positives. As can be seen in figure 8.8, the resulting pose quickly diverges. Oscillation of the position error is caused by switching of the filter's output between different equally uncertain particle clusters. The presented algorithm's better accuracy is clearly visible. Divergence is still not avoided completely, which is a consequence of mapped features neither assumed to be static nor kept in the map after longer periods without observing them. This is appropriate for highly dynamic situations, but of course prevents closing of large loops. Therefore the robot's perception can only negotiate

[5]While the motion frame rate was capped to 50 Hz in the NaoV3+ used in 2009, the upgrade to the NaoV3.2 in early 2010 increased the possible motion frame rate to 100 Hz, which allowed a much smoother motion control. As a consequence, even less processing time was available for the cognition process, and the dynamic FastSLAM approach could not be employed on real Naos in subsequent years.

a trade-off between the drift introduced by inaccurate odometry data and the uncertainty in the process model of dynamic objects.

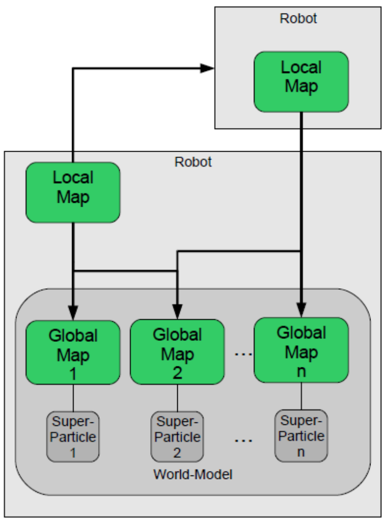


(a) Good case, i.e. localization converges to few (or even one) local probability maxima.

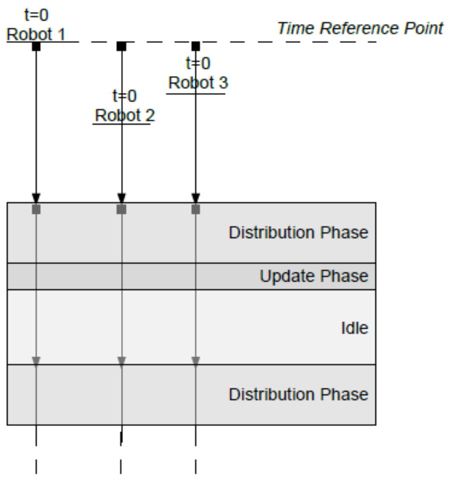


(b) Worst case, no perception conclusive enough for generating a useful belief state.

Figure 8.3: Coverage of particles by the use of a fixed number of super particles in different cases.



(a) Updating with local and distributed partial maps.



(b) Synchronous distribution and updating of world maps.

Figure 8.4: Distribution of temporary local models.

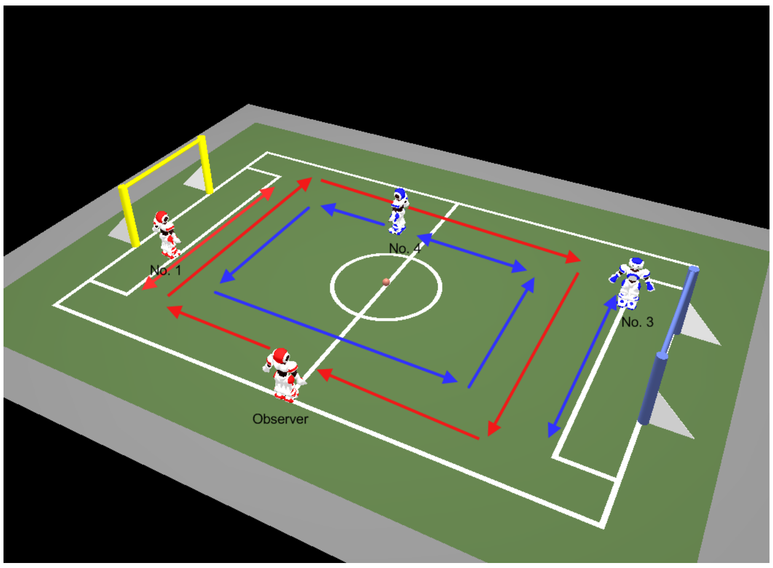
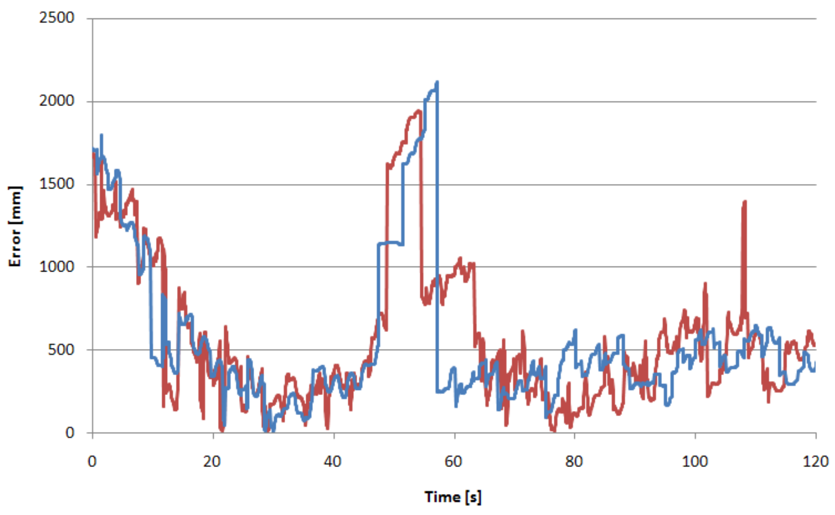
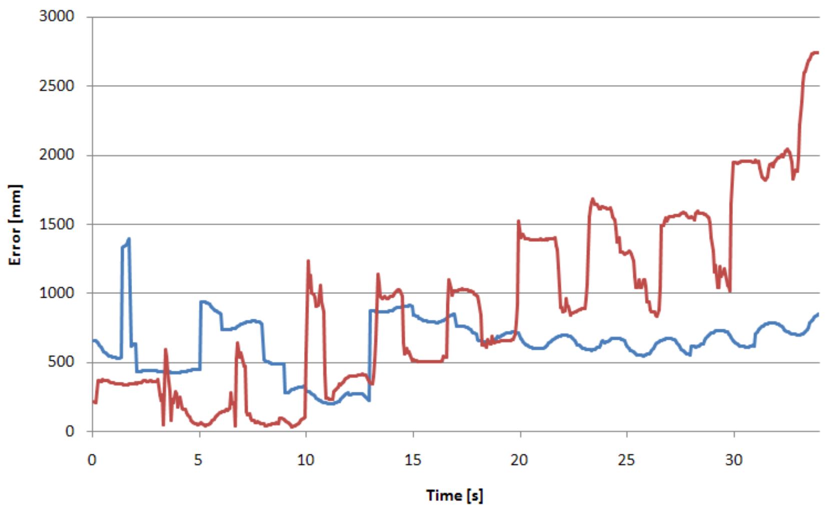


Figure 8.5: Experimental setup of a RoboCup Standard Platform League scenario.



(a) No gain for frequently observed robots.



(b) Tracking of infrequently observed robots significantly improved.

Figure 8.6: Advantage by unified (blue) compared to separate modeling (red).

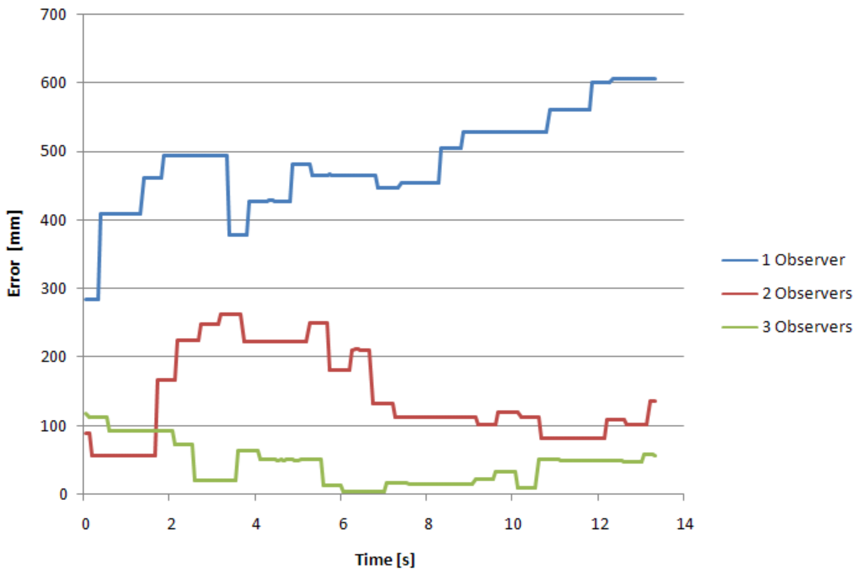


Figure 8.7: Accuracy gain by multi-robot sensor fusion.

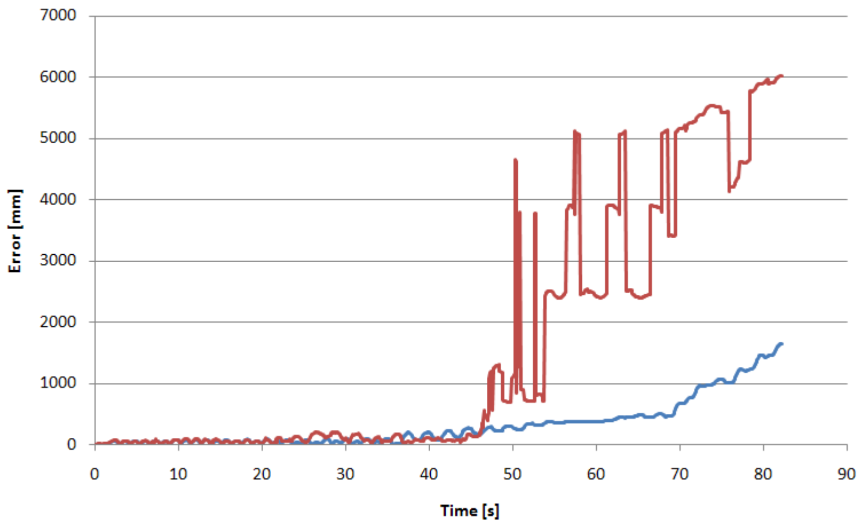


Figure 8.8: Preservation of the filter's SLAM functionality of the proposed algorithm (blue). Reference localization is shown in red.

8.3 Real-time Dynamic EKF-SLAM

[6] Two arguments hold against the solution presented in section 8.2: the insufficient performance of the implementation to be executable on a Nao's Geode CPU in real-time, and the parallel implementation of a superior localization solution, namely the multi-hypotheses Kalman system described in chapter 7. Due to these reasons, a different system is presented in this section as an extension of the multi-hypotheses Kalman filter to enable a simultaneous tracking and localization in a unified world state estimation.

8.3.1 Implementation

The objective now is the realization of the demands specified in section 8.1, namely to model the robot's surrounding environment in one unified model using the information of a whole team of robots as input, in the context of a multi-hypotheses Kalman filter system. It is hardly possible to implement this as a real-time system on an embedded platform without applying measures to decrease the computational complexity. The presented approach consists of three stages, which will be covered in the following sections. The first stage handles the static map information to realize most (but not all) of the localization problem, and is based on the algorithm presented in chapter 7, which can perform as a very efficient stand-alone localization. Parallel to this runs a stage performing local percept aggregation according to the temporary short-term models described in section 8.1.2. Finally, section 8.3.1.3 presents the integration of local and distributed perceptions into a consistent global world model.

8.3.1.1 Multi-model Kalman Localization

In contrary to the system described in section 8.2, which is based on a particle filter localization, the approach presented here bases on a multi-hypotheses Unscented Kalman Filter (UKF) localization. This utilizes an approach to Gaussian mixture filtering which combines the accuracy of the Kalman filter and the robustness of particle filters without sacrificing computational efficiency.

To use this in the context of a unified world model, it is necessary to keep track of the history of each hypothesis' origin for fusion and spawning of new hypotheses, and the change of the likelihoods among the set of hypotheses. If a former most likely hypothesis is surpassed in likelihood by another one, this corresponds to a re-localization event, e.g. with a kidnapped robot or after temporary localization loss caused by extreme odometry errors or collisions.

[6] Parts of this section are based on an article appearing in the Proceedings of RoboCup 2012: Robot Soccer World Cup XVI. The corresponding publication, see [85], is available at `http://www.springerlink.com`.

Otherwise, each estimate's change can be considered as a pre-filtered input for the global estimation system. This input bears the characteristics, on the one hand, of partially corrected odometry data, and on the other hand the characteristic of buffered and pre-processed sensor data. In addition to this, integration of further information, including the communicated observations of other robots, can affect the pose estimates, so those changes need to be fed back into the localization module. The following sections will address the integration into the global model and its stochastic soundness.

8.3.1.2 Local Percept Aggregation

When building upon the UKF localization described above, the full state cannot be factorized as in FastSLAM, but needs to be expressed as a joint probability function, as in the EKF-SLAM solution (cp. section 3.5.1). The increase of estimation complexity by the high-dimensional state is countered by aggregation of some of the image processing results into temporary percept-buffers with the aforementioned advantages, as motivated in section 8.1.2. This is applied to full extent to the dynamic features, i.e. the ball and other robots, as described in section 8.1.3.

The separate localization module described in section 8.3.1.1 is in itself also a buffer integrating information from static, known world features into a localization belief model, and is used analogically to those percept-buffers, but the state is not deleted periodically after forwarding the belief to the SLAM part of the algorithm. This localization reflects part of the SLAM state, and changes to this part of the SLAM state are fed back into the localization module's state. Thus the virtual localization measurements used to update the SLAM state are basically the innovation introduced by new static feature observations. Therefore, those measurements are still conditionally independent from previous measurements given the current belief state, so the Markov assumption is not violated.

8.3.1.3 Local and Distributed Knowledge Integration

The state of the full model of the robot's environment consists of its own pose $p_0 = (p_{0,x}, p_{0,y}, p_{0,\theta})^T$, the poses of all cooperating robots ($p_i = (p_{i,x}, p_{i,y}, p_{i,\theta})^T$ with $i \in \{1, ..., n\}$), and the states of the dynamic objects. While only a small subset of cooperating robots or other elements are observed at the same time and modeled according to section 8.3.1.2 in each time interval, they remain part of the full model also during time intervals where these are not observed. It is possible to dynamically shrink or expand the state vector if new unknown robots are observed. Alternatively a separate mechanism could keep track of active and inactive slots in the state vector by using time-to-live counters. This latter approach has been chosen here to prevent frequent rescaling of both the state vector and its covariance matrix.

The integration of the locally accumulated and the distributed information into the model is done in the process and sensor update. The own pose and those of cooperating robots can be updated with the pose changes propagated from the individual localization modules relative to the pose used for the last update. The ball is updated using a motion model similar to the one in equation 8.6, but without the odometry-related rotations necessary for tracking in the local coordinate system. Other autonomous agents can either be updated according to the latest velocity estimations, or just using an identity and appropriately high process noise following the reasoning proposed in section 8.1.2.

The sensor update consists of two different kinds of observations. If a robot, either the local robot itself or any of the communicating robots in the team, has made observations of static world elements which have been used to update the separate localization estimate in the first stage (cp. section 8.3.1.1), then this absolute pose estimate is used as a direct measurement of the corresponding pose in the state vector, i.e. the measurement Jacobian is an identity in the corresponding submatrix.

The other case is the observation of a dynamic feature by one of the robots in the team. If the observed dynamic feature is a robot (without further identified characteristics such as team markers etc.), this dynamic object may either be any of other robots in the team, or one of a number of non-cooperating other robots in the environment. In this case, the maximum likelihood correspondence will be chosen to be updated, or a new model will be inserted or activated if the other choices are too unlikely. The corresponding expected observation is in a robot-relative euclidean coordinate system, since this is the format of the local models distributed as aggregated percepts. It is expressed as a function of the observed object's model $(m_x, m_y, m_{v_x}, m_{v_y})$ and its observer's pose p_i, with $i = 0$ for local observations and $i > 0$ for communicated ones, which are otherwise not distinguished any further.

The observation model is given by equations 8.13 and 8.14

$$\begin{aligned} h_{m_x,m_y}(p_i) &= \Omega(-p_{i,\theta})\left[(m_x, m_y)^T - (p_{i,x}, p_{i,y})^T\right] && (8.13) \\ h_{m_{v_x},m_{v_y}}(p_i) &= \Omega(-p_{i,\theta})\left(m_{v_x}, m_{v_y}\right)^T && (8.14) \end{aligned}$$

from which the corresponding entries in the measurement Jacobian can be calculated as in equation 8.15, with c_θ and s_θ short for $\cos p_{i,\theta}$ and $\sin p_{i,\theta}$, respectively.

$$\begin{pmatrix} \frac{\partial h_{m_x}}{\partial m_x} & \frac{\partial h_{m_x}}{\partial m_y} & \frac{\partial h_{m_x}}{\partial m_{v_x}} & \frac{\partial h_{m_x}}{\partial m_{v_y}} & \frac{\partial h_{m_x}}{\partial p_{i,x}} & \frac{\partial h_{m_x}}{\partial p_{i,y}} & \frac{\partial h_{m_x}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_y}}{\partial m_x} & \frac{\partial h_{m_y}}{\partial m_y} & \frac{\partial h_{m_y}}{\partial m_{v_x}} & \frac{\partial h_{m_y}}{\partial m_{v_y}} & \frac{\partial h_{m_y}}{\partial p_{i,x}} & \frac{\partial h_{m_y}}{\partial p_{i,y}} & \frac{\partial h_{m_y}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_{v_x}}}{\partial m_x} & \frac{\partial h_{m_{v_x}}}{\partial m_y} & \frac{\partial h_{m_{v_x}}}{\partial m_{v_x}} & \frac{\partial h_{m_{v_x}}}{\partial m_{v_y}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,x}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,y}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_{v_y}}}{\partial m_x} & \frac{\partial h_{m_{v_y}}}{\partial m_y} & \frac{\partial h_{m_{v_y}}}{\partial m_{v_x}} & \frac{\partial h_{m_{v_y}}}{\partial m_{v_y}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,x}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,y}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,\theta}} \end{pmatrix} \tag{8.15}$$

$$= \begin{pmatrix} c_\theta & s_\theta & 0 & 0 & -c_\theta & -s_\theta & -(m_x - p_{i,x}) \cdot s_\theta + (m_y - p_{i,y}) \cdot c_\theta \\ -s_\theta & c_\theta & 0 & 0 & s_\theta & -c_\theta & -(m_x - p_{i,x}) \cdot c_\theta - (m_y - p_{i,y}) \cdot s_\theta \\ 0 & 0 & c_\theta & s_\theta & 0 & 0 & -m_{v_x} \cdot s_\theta + m_{v_y} \cdot c_\theta \\ 0 & 0 & -s_\theta & c_\theta & 0 & 0 & -m_{v_x} \cdot c_\theta - m_{v_y} \cdot s_\theta \end{pmatrix}$$

Re-localization events can be handled by resetting the corresponding state variables and removing the covariances, i.e. setting all entries in the covariance matrix in the rows and columns to zero. If such a previous mis-localization by a team member resulted in modeled false positives, those will stay as isolated features in the state for some time and will be deleted or inactivated after a certain time without observation. This serves as a self-repair routine to remove clutter from the environmental model, and to prevent the growth of the state by the accumulation of models of such elements. The same is done if two models of unknown features are decided to correspond to the same origin after a series of observations, so that the information needs to be fused into the first model and the seconds needs to be deactivated. Alternatively it would be possible to keep multiple environment models for each localization hypothesis, as done in [17].

8.3.2 Evaluation

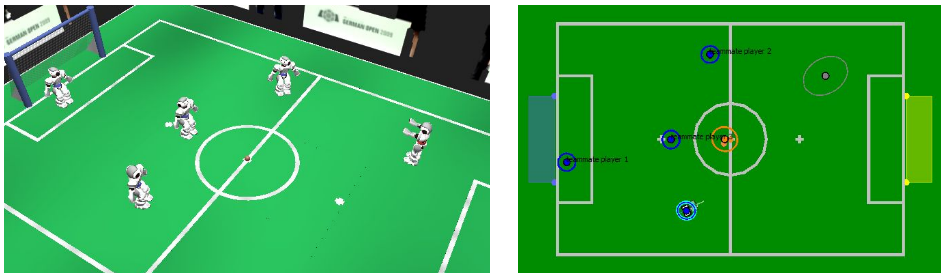
The mechanisms for modeling the robot's own position and its environment in a unified state are conceptually similar in this solution to the one presented in section 8.2. The first important difference of this system is its applicability on a real Nao. This is partly due to the already superior efficiency of the underlying localization as shown in chapter 7. The runtime necessary to additionally perform the unified world modeling on the current most likely hypothesis and the communicated information is on average below 1 ms, with peaks in the range of 3 ms to 5 ms. This solution is therefore able to run in real-time on real Nao robots and can be evaluated in realistic soccer environments.

In the following, the evaluation does not concentrate on object modeling advantages compared robot-centric tracking or the benefits of cooperative modeling, which have already been verified in section 8.2.3 using similar mechanisms. Instead, this evaluation goes beyond these by analyzing the system's potential to even improve a robot's localization using the unified modeling state and communicated information.

To evaluate the presented approach, a simulated situation first illustrates the possibilities and the qualitative effect in section 8.3.2.1, followed by a quantitative analysis in soccer games using experiments with real robots in section 8.3.2.2. Both setups use an SPL scenario as specified by the 2011 rules.

8.3.2.1 Qualitative Demonstration in a Static Simulated Setup

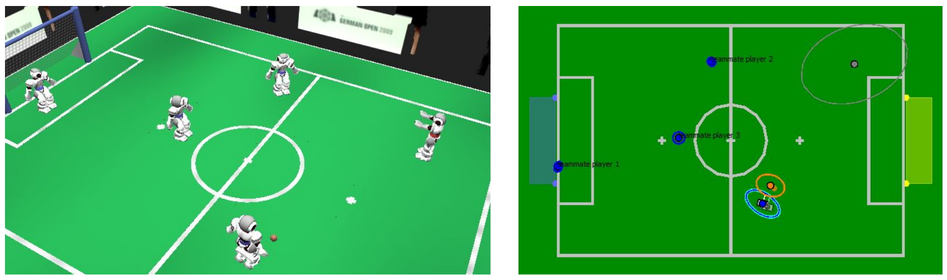
Figure 8.9 illustrates a simple scenario in a simulated environment. The robots in a team share their information for distributed cooperative modeling. Figure 8.9b shows the resulting model with 2D covariance ellipses extracted from the full state. In the following, one robot looks down and does not see any static field features any more, and both it and the ball are teleported to another location on the field (cp. figure 8.10). The use of distributed percepts and the modeling of the own pose together with the ones of other robots and the ball position and velocity allows the robot not only to correct its position, but also to find its orientation again.



(a) Setup of the robots on the field. (b) World model generated from local and distributed information.

Figure 8.9: Scenario with a team of robots looking around and sharing perception information to cooperatively model their environment.

This simple experiment shows the potential usefulness of such a combined modeling of a robot's dynamic environment and its pose in it. RoboCup SPL games contain periods where robots are chasing the ball, approaching it for precise positioning to shoot at the goal, or even dribbling it. During those periods, odometry errors are integrated into the robot's localization if not countered by frequently looking up at static field features to correct the robot's pose estimation. If looking at the ball also allows the correction of those odometry errors, especially the orientation, this is expected to be a clear advantage.



(a) Scenario after teleportation of ball and downwards-looking robot.

(b) World model generated from local and distributed information.

Figure 8.10: Following the situation in figure 8.9, one robot looks down and only sees the ball but no landmarks, then this robot and the ball are teleported. The shared information however still allows for a correction of both position and orientation of the robot.

8.3.2.2 Quantitative Performance Evaluation in a Dynamic Real-World Experiment

The artificial situation created in the previous section just serves as an example of how localization benefits may be gained. To allow a quantitative evaluation of the approach's performance, the perceptions of a robot have been recorded during normal game situations with real robots on a regular SPL field. Those perceptions include the proprioception, i.e. odometry, orientation and joint angle information, exteroception, i.e. perceptions of objects by means of image processing, as well as the distributed local models of other cooperating robots running the same code, and ground truth information provided by a camera system mounted above the field. The latter is used only for the evaluation.

This set of input information is then processed by three different module configurations:

- *Reference configuration*: The localization as described in chapter 7, together with a simple module for cooperative tracking of dynamic objects without any feedback into the localization, serves as the reference solution.

- *SLAM configuration 1*: The second configuration is the one described in section 8.3.1, which models the own state and all dynamic elements of the environment in a unified state.

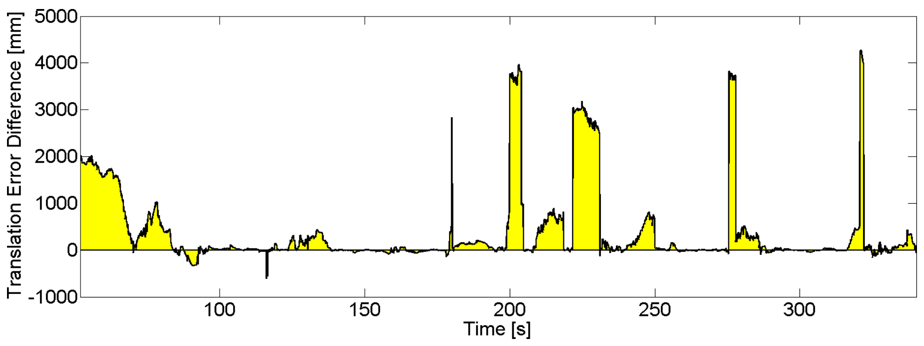
- *SLAM configuration 2*: The third is identical in principle to the second one, but neglects visual obstacle perceptions. Thus, only observations of static field features and of the ball as a dynamic object are utilized to model the environment. This configuration is motivated by implications of the following experiments.

All these solutions are based on the same competitive solution for the localization problem with all features described in chapter 7. It is therefore not this experiment's purpose to show that the localization works, but to evaluate the additional benefit gained by unified modeling of the full state. To do so, the differences in the localization error between the reference and the SLAM configurations are evaluated. In the following, a representative extract is chosen to illustrate the positive and negative effects of the modeling methods and information sources. Note that communicated information among the team of robots has no influence on the reference configuration's localization outcome. To separate the benefit of this additional information source and that of the modeling method, the analysis is done with and without providing communicated information for each configuration.

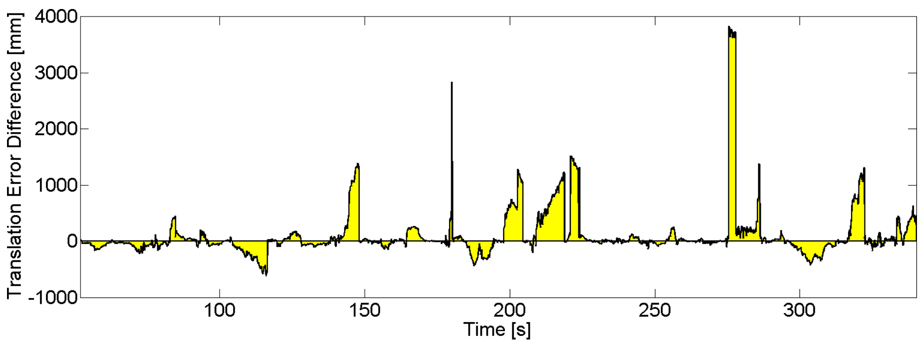
Figure 8.11 visualizes the differences of the translational localization errors between the reference configuration and the SLAM configuration 1. Both plots show the SLAM configuration's errors minus the reference configuration errors, thus negative values mean lower errors and a superior localization quality of the SLAM configuration. As can be seen in figure 8.11a for the utilization of only local information, the SLAM configuration 1 actually shows a severely degraded localization accuracy compared to the plain localization system of the reference solution. This even leads to occasional switches to incorrect pose hypotheses on the wrong side of the field, indicated by sudden large errors at 180 s, 200 s, 220 s, 275 s and 320 s. Overall, a localization error below 25 cm can only be achieved in 49% of the time compared to the 72% of the reference configuration. As expected, the SLAM configuration 1 benefits from shared information and the frequent miss-localizations do not occur any more, but even with this additional information source it still performs significantly worse compared to the stand-alone reference localization system (cp. figure 8.11b). A detailed analysis reveals that the robot perceptions of the used image processing system are too unreliable to be of any benefit for the localization accuracy. This is due to frequent false positives and a huge positional uncertainty which far exceeds that of other measurement types generated by the same image processing routines.

A consequence of this finding is the motivation of SLAM configuration 2, which disregards visual obstacle perceptions and thus does not include opponent robots into the full state. Thus, the state contains only the robot's own pose, the poses of all team mates, and the ball's position and velocity. For the setup with only local information, figure 8.12a shows that the severe miss-localizations of SLAM configuration 1 in figure 8.11a do not occur any more. In fact, the SLAM configuration 2 has a lower error compared to the reference configuration 62% of the time, but the difference itself is insignificantly low.

The real advantage of the SLAM configuration 2 compared to the ref-



(a) Localization error difference without any communicated information.

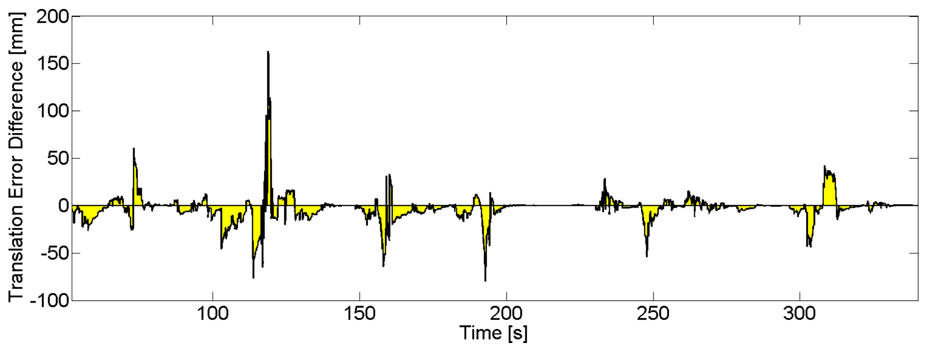


(b) Localization error difference with the utilization of communicated information.

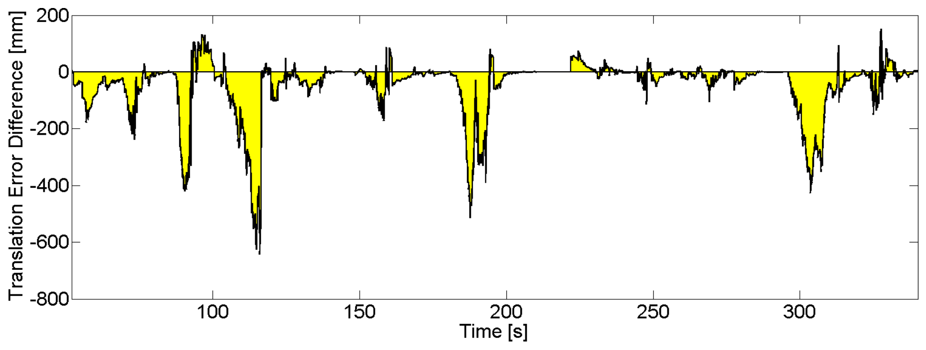
Figure 8.11: Difference of translational localization errors of reference and SLAM configuration 1, i.e. including the use of visual obstacle perceptions. Negative values mean larger errors of the unassisted localization compared to modeling the full state.

erence system can be seen in figure 8.12b, where communicated team information is included as an input for the unified modeling. Here, the proposed system provides beneficial information for the robot's own localization 75% of the time. A direct comparison of the localization quality of SLAM configuration 2 and the reference configuration shows that the robot pose translation errors for the full system model are below 25 cm in 83% of the time, and only 72% of the time for the unassisted underlying localization module, and the average errors over the whole experiment are 166 mm compared to 213 mm. However, note that with the SLAM configuration 2 of the system, in the teleportation experiment such as presented in section 8.3.2.1, the robot's orientation can obviously not be recovered as easily as described above.

Similar to the simulated experiments in section 8.2.3, the benefit of modeling the full state compared to separate tracking is evaluated for estimating the state of the dynamic objects. To evaluate this, a separate ball tracking module runs in parallel to the SLAM configuration 2, and the results



(a) Localization error difference without any communicated information.



(b) Localization error difference with the utilization of communicated information.

Figure 8.12: Difference of translational localization errors of reference and SLAM configuration 2, i.e. without the use of visual obstacle perceptions. Negative values mean larger errors of the unassisted localization compared to modeling the full state.

of the two different modeling methods are compared without communicated information and using the same robot pose information. The results confirm the tendencies already detailed in section 8.2.3 for the FastSLAM approach, namely that the difference is minimal as long as dynamic objects are observed frequently, but that an improved robustness is observable for periods of infrequent observation. Consequently, the modeling results display equal accuracy (within a margin of 5 cm) for 66% of the time, during which the ball is either observed nearly every frame or has been out of sight long enough to be marked as “lost” by both solutions. However, 71% of the remaining time the full state modeling is supperior to the separate tracking module. Over the whole experiment, the position error is within a 20 cm bound for 52% of the time for the unified modeling, but only 40% of the time for the separate modeling. The use of communicated information increases the time with errors below 20 cm to 73%.

Overall, the experiments with real robots presented here show the benefit of modeling the full state of the robot's environment. Unfortunately, observations of other robots are currently not precise enough that using models of opponent robots for a robot's own localization improves the estimation. However, using distributed information among a team of robots in a model of the full state, including the own localization, the team mate poses and the ball, proves advantageous. It improves the relative pose estimations compared to separate modeling and also provides a globally consistent model useful for the robot's behavior decision making.

8.4 Conclusion

This chapter presents two different implementations of an approach for simultaneous localization and modeling of dynamic elements of a robot's environment using a-priori information and efficient handling of classes of sensor information with huge differences in uncertainty. In section 8.2, FastSLAM is adapted for the use of prior map information and extended with the concept of super-particles to reduce the number of world maps for dynamic elements. These features reduce the algorithm's runtime enough to be in the range where an application becomes possible even on restricted embedded platforms such as the humanoid robot Nao by Aldebaran Robotics. In several experiments the advantage of a united state estimation using the presented approach is shown against the common solution of tracking dynamic objects only in a local robot centric model. The aspects of distributed modeling using this system are presented and the preservation of the SLAM functionality of the proposed FastSLAM adaptation is proven in a scenario where no a-priori known features are present but only dynamic elements.

Thus, the presented method is an extension to the popular FastSLAM algorithm overcoming the strict linear relation between the number of particles necessary for localization and the complexity for mapping the robot's environment. Applicability is also given in pure SLAM contexts without prior information. In case of a first reasonably good map estimate, additional features can be tracked using super-particles while maintaining the good localization performance possible by a high number of particles. Parts of a map estimate, which develop a reliable global consistency, could also be flagged as known and used similarly as prior information not in need of further estimation. The algorithm's performance on such a limited platform implies its applicability to much larger problems when using state of the art hardware.

In section 8.3, the competitive stand-alone localization module from chapter 7 is extended to perform as a full state model. In addition to the advantages shown before, the additional gain in localization performance is evaluated both in a simulated situation as well as in several real world experiments with multiple robots and ground truth provided by an external camera system. While the robot perception in the current vision system is not good enough to benefit from using temporary opponent models as additional features for localization, usage of the ball as a dynamic feature significantly improves the localization quality.

An additional advantage of estimating the full state in a cooperative modeling approach is the existence of a single model which contains all information in a globally consistent way. This renders the switching between local tracking of the ball and a global team ball model obsolete, for example, and therefore simplifies behavior specification.

Chapter 9

Conclusion and Outlook

In this work, stochastic filtering is applied on mobile devices to model their location and eventually also relevant parts of their dynamic environment. Three application scenarios are introduced as a motivation, but also as sources of experiments for the evaluation of the proposed methods: humanoid robotics with robot soccer as a benchmark task, mobile measurement units on intralogistic conveyor belt systems, and location-based services on smartphones. After an overview of concepts and state of the art algorithms from the field of probabilistic robotics, this work's contribution is presented in a series of applications, findings and conclusions. Those are organized along the following topics: proprioception, which summarizes all inward perception, the sensing of one's own body and its dynamics; exteroception, which covers all outward perception; localization by utilizing concepts from the former two topics as the means to estimate one's own location in various different environments and scenarios; and world modeling, i.e. the estimation not only of one mobile system's location, but the localization of a whole team of robots and other dynamic properties of a complex dynamic environment simultaneously in a unified modeling approach.

By presenting various results and insights, this work provides general concepts and guidelines for the modeling of proprioceptive and exteroceptive processes. These aid in the conception, design and implementation of appropriate stochastic filtering solutions. To this end, different filter solutions are derived for an application in intralogistic systems and their predicted characteristics are verified experimentally. The influence of coordinate system choices for perception representations and the impact of different handling of simultaneous measurements on the resulting estimation quality is presented using the example of humanoid robot localization. Additional possibilities beyond sensor model design are presented to optimally exploit an existing perception process, such as the active control of attention and sensor orientation to optimize the gathered information with respect to the estimation process.

Based on these presented concepts and guidelines about proprioception and exteroception, filter systems are systematically designed for humanoid robot and smartphone localization, and shown to surpass current state of the art. The probabilistic robotics formulation of the indoor smartphone localization problem for location-based services is novel and particularly challenging, since no locomotion information is readily available, but intention and motion execution lie with the phone's human user and cannot be accessed by the estimation system directly. Instead, knowledge about biped walking motion generation is used to infer the underlying walking motion from acceleration measurements, and thus to provide odometry data as input for a stochastic motion model. Gaussian processes are employed as an automated Bayesian inference process to generate mappings of a building's inherent magnetic field as well as the various Wi-Fi signal strengths. The former are used to correct heading information in the motion model, while the latter are used in the sensor model to predict means and variances, and thus likelihoods of received Wi-Fi signals. The resulting localization filter proves the superiority of this concept over conventional indoor localization techniques for smartphones based on Wi-Fi localization.

A novel approach to Gaussian mixture filtering is presented for the localization of humanoid soccer robots. By analyzing the behavior of Gaussian mixture filtering under constraints of limited processing power, i.e. with only few Gaussian terms in the mixture and aggressive pruning of growth during updates, parallels can be drawn to particle filtering with a low number of particles. Consequently, techniques from particle filtering are integrated into a multi-hypotheses Kalman filter, which allows separate Gaussian terms to act as maximum likelihood trackers, while maintaining the systems overall ability to recover from a series of incorrect observation correspondence choices, and even to re-localize globally in kidnapped robot situations. The proposed system possesses a particle filter's re-localization ability and an even surpassing robustness to systematic false positive observations, while at the same time maintaining a Kalman filter's typical position tracking accuracy and computational efficiency.

Finally, the task of estimating a single mobile robot's location is extended to that of estimating all locations of robots in a team as well as other agents and objects in their dynamic environment. First, several possibilities are presented to exploit the heterogeneity of a complex environment's information sources by translating the differences in implied uncertainty into approximations, which significantly improve the efficiency of common SLAM algorithms. As a consequence, these algorithms, which originally aim at simultaneous localization of a single robot and mapping of static, homogeneous environment features, are applied to the task of cooperatively modeling the dynamic and heterogeneous environment of a team of robots with very limited computational resources. The benefit of this world modeling in a unified estimation state compared to the classic approach of separate localization and robot-centric tracking is presented with respect to the improved dynamic world map, and even with respect to each robot's localization quality.

In summary, this work offers a variety of general design guidelines, new concepts, and solutions for concrete applications scenarios. It provides a stochastic and algorithmic basis for the modeling of proprioceptive and exteroceptive processes and the systematic design of estimation systems.

This dissertation's results and achieved goals open the opportunity for further work in and beyond the presented application scenarios. The intralogistic tracking task, which in this work is mostly used to exemplify state space and filter design choices, demands further experiments in more complex industrial setups including an evaluation of its original intention of mapping potential failure areas. The potential of the smartphone localization approach in particular seems promising, as it surpasses other proposed approaches in achieved accuracy and its easy deployment. To enable realistic experiments over multiple days, dependencies on certain IMU characteristics need to be overcome and spontaneous re-calibration of the magnetic sensor needs to be repressed or at least monitored, which is not possible with the specific model and Android version used in the experiments so far. On the other hand, the availability of gyroscopes along acceleration and magnetic field sensors in all new smartphones provides possibilities to further improve the odometry estimation. Additional evaluations are needed to prove the scalability of the presented approach, and further functionality is necessary for the detection of walking up and down stairs or using elevators to allow an application in large buildings along multiple floors.

In the context of humanoid robots in RoboCup competitions, the recent upgrade of the Nao to version 4 in 2012 opens up new possibilities. This upgrade improves Nao's CPU and its cameras, even if these are still below the ones of current state of the art smartphones. In fact, the potential of the additional processing power has already been explored by implementing a more powerful localization system for the more challenging scenario of a completely symmetrical field, which has been introduced to competitions in 2012. However, since the underlying perceptions system is changed drastically with respect to its software by a new image processing module as well as its hardware by allowing access to both cameras simultaneously, which are already better each individually, these recent implementations cannot be compared to the presented ones and their state of the art counterparts, and are therefore omitted from this work. Nevertheless, the more challenging setup and the more capable hardware will impel the state of the art in the RoboCup Standard Platform League and offer various opportunities for research using the Nao platform, including the further exploration of cooperative modeling and estimation systems. Especially these latter ones and the benefit of applying SLAM concepts to complex dynamic environments might be applied to and evaluated in other robotic contexts beyond the strictly specified RoboCup scenario. As argued earlier, many office or domestic indoor scenarios share these characteristics of an environment which is partially known a-priori, but contains unmapped elements necessary for localization, and many dynamic and changing objects or agents. An application of the presented concepts to different robotic platforms in such scenarios will provide new insights and potential benefit.

Appendix A

Discrete Correspondences for Continuous Map Features

The robot soccer scenario in section 2.1 allows measurements of different static field features. The goal posts and the center circle can be modeled as 2-dimensional point features, as they are fully characterized by this even though they actually cover an area or extend further into the third dimension. Other observable features, namely the field lines, are continuous in nature. Even the observation of the end of a field line segment can be ambiguous, as the continuation of the field line may simply be blocked by something else which has not been recognized clearly.

On a first look, such continuous features should be handled by different sensor models than the ones introduced and compared in section 5.1. In [81] lines are incorporated into a monocular Kalman SLAM system. However, those lines are actually represented as segments by their end points, thus reducing the problem to well known point updates. Since the line detection algorithm employed in [81] recognizes lines by detecting corners first and then verifying their connecting line in between, only observations of full segments are used as input for this Kalman system and observations of parts of a line do not occur by definition.

In an SPL scenario, both ends of a line segment are rarely observed in the same frame. To reduce such a field line observation to a set of point feature measurements, one could simply project the end points of the observed line segment onto the nearest field line as a maximum likelihood approximation for the association for each pose hypothesis. However, in most situations, more than one field line is visible at the same time, and using this procedure for each line observation independently may generate correspondences which are inconsistent and thus harmful to the estimation stability when expressing the feature correlation by a full covariance matrix as described in section 5.1.2. This motivates the computation of a mutually consistent maximum likelihood correspondence generation for field line segment observations.

Finding conclusive correspondences among the different permutations of possible correspondences is a combinatorial problem. The naive approach would be to enumerate every possible combination of observed line with a line in the map, and then check each correspondence combination whether its chosen map associations could have generated the given observations. This naive solution can be improved in two aspects: As a start, the set of possible combinations of correspondences can be restricted by dividing observed lines and those in the map into two perpendicular sets each, and omitting associations of observed parallel lines to perpendicular lines in the map. Furthermore, additional combinations can be skipped during operation. This is case if the first correspondence choices already lead to expected observations which do not fit the real observations. Any combination, which does not change these first correspondences, is equally invalid and can thus be omitted from further consideration.

The resulting more efficient computation of all possible line correspondences is given by algorithm 3. The division of lines in the map into those along the longitudinal and those along the lateral axis of the field is given by the two sets $\mathcal{M}_1$ and $\mathcal{M}_2$. The division of observed lines on the other hand is done in line 2. As the projections of observed lines onto the field rarely produce only parallel or perpendicular lines, but skewed and warped angles due to errors in the transformation between camera and robot coordinate frame, i.e. the robot's proprioception (cp. chapter 4). To allow robustness to medium projection errors, the division into perpendicular line sets $\mathcal{L}_1$ and $\mathcal{L}_2$ is done using the main direction α_{main}, which is determined in line 1. This is done in the given implementation using an accumulator for the range of directions from 0° to 90°, i.e. the first quadrant. Each line has either its direction in this quadrant, or its normal, so that for example lines with directions 25°, 115°, 205° and 295° all map to 25° and consequently vote on all accumulator cells of an allowed error threshold around the cell corresponding to 25°. If not only the votes are stored, but their originating lines as well, the main direction can be estimated robust to projection errors or even to sporadic direction outliers[1].

Once the set of all observed lines $\mathcal{L}_{observed}$ is divided into two perpendicular sets $\mathcal{L}_1$ and $\mathcal{L}_2$ parallel and perpendicular to the main direction α_{main}, there are two different possibilities. In case $\mathcal{L}_1$ and $\mathcal{L}_2$ contain both at least one observation, it is possible to calculate a discrete pose for each valid correspondence choice. This is done in line 7 in algorithm 3, which is elaborated in more detail in algorithm 4. In the other case, only parallel lines are observed, so in general it is only possible to determine a continuous pose interval for each valid correspondence choice, since a translation along those

[1]The wrap around from 90° to 0° has to be considered when storing directions for calculating their average.

parallel lines is only restricted by the difference in length of the observed segment and the full line in the map.

Algorithm 3: Line correspondence computation.

```
Input: The set of observed lines $\mathcal{L}_{observed}$; two sets of lines in the
       map $\mathcal{M}_1$ and $\mathcal{M}_2$, along the longitudinal and lateral axis of
       the field, respectively.
1  Determine the main direction $\alpha_{main}$ among $\mathcal{L}_{observed}$;
2  Build line clusters $\mathcal{L}_1$ and $\mathcal{L}_2$ from $\mathcal{L}_{observed}$, parallel and
   perpendicular to the main direction $\alpha_{main}$, respectively;
3  $Poses = \emptyset$;
4  $PoseIntervals = \emptyset$;
5  $LineCorrespondences = \emptyset$;
6  if ($|\mathcal{L}_1| > 0$ AND $|\mathcal{L}_2| > 0$) then
7      Find possible poses $Poses$ and line correspondences
       $LineCorrespondences$ using $\mathcal{L}_1$, $\mathcal{L}_2$, $\alpha_{main}$, $\mathcal{M}_1$ and $\mathcal{M}_2$;
8  else if ($|\mathcal{L}_1| > 0$ OR $|\mathcal{L}_2| > 0$) then
9      Find possible pose intervals $PoseIntervals$ and
       $LineCorrespondences$ using $\mathcal{L}_1$, $\mathcal{L}_2$, $\alpha_{main}$, $\mathcal{M}_1$ and $\mathcal{M}_2$;
10 end
11 return $Poses, PoseIntervals, LineCorrespondences$
```

As mentioned above, certain combinations of line correspondences can be skipped during operation, if part of that combination already turned out to be geometrically inconclusive. In order to exploit this as much as possible, the enumeration of possible combinations should treat long segments as more significant than small segments. The two most significant associations are for the longest lines in $\mathcal{L}_1$ and $\mathcal{L}_2$, in order to be able to determine the candidate pose (p_x, p_y, p_θ) directly. Then the correspondences are checked one by one whether the given observations fit the expected ones based on the candidate pose (p_x, p_y, p_θ) in line 8. If this is the case, the pose and its line correspondence list is stored as a viable correspondence choice. In either case, line 17 sets $\mathcal{C}$ to the next correspondence combination, as long as there are any left. In order to skip the evaluation of combinations which can be marked as invalid based on a previous evaluation, *nextCorrespondenceIndex* marks the index of the correspondence in the current combination, which needs to be changed next in the enumeration to generate the next possibly valid combination. In case the current $\mathcal{C}$ is valid, this is simply the last index (cp. line 13). Otherwise, this is the first index which results in a conflict between expected observation and real observation (cp. line 15).

Algorithm 4 details line 7 in algorithm 3, i.e. when distinct poses can be inferred from every valid correspondence choice. Line 9 in algorithm 3

Algorithm 4: Find possible poses and line correspondences.

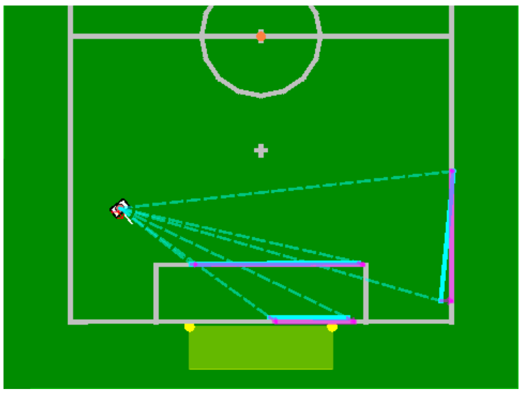
```
   Input: $\mathcal{L}_1$, $\mathcal{L}_2$, $\alpha_{main}$, $\mathcal{M}_1$, $\mathcal{M}_2$ (as defined in algorithm 3).
1  Let $\mathcal{C}$ be a list of correspondences between a fixed order of
   observations from $\mathcal{L}_1$ and $\mathcal{L}_2$ and mapped lines $\mathcal{M}_1$ and $\mathcal{M}_2$,
   respectively.;
2  $Poses = \emptyset$;
3  $LineCorrespondences = \emptyset$;
4  $p_\theta = -\alpha_{main}$;
5  for $i = 1$ to $4$ do
6      Initialize $\mathcal{C}$;
7      repeat
8          Determine $p_x$ and $p_y$ from observations $\mathcal{L}_1^{[1]}$ and $\mathcal{L}_2^{[1]}$ and their
           correspondences in $\mathcal{M}_1$ and $\mathcal{M}_2$ according to $\mathcal{C}$;
9          Check geometrical validity of all observations for the
           correspondence combination defined by $\mathcal{C}$;
10         if ($\mathcal{C}$ is geometrically possible) then
11             $Poses = Poses \cup (p_x, p_y, p_\theta)^T$;
12             $LineCorrespondences = LineCorrespondences \cup \mathcal{C}$;
13             $nextCorrespondenceIndex = |\mathcal{C}|$;
14         else
15             Set $nextCorrespondenceIndex$ to the lowest index in $\mathcal{C}$
               which results in a conflict between expected observation
               and real observation;
16         end
17         Set $\mathcal{C}$ to the next possible combination (skip all combinations
           which only differ in indices higher than
           $nextCorrespondenceIndex$;
18     until no correspondence combination left;
19 end
20 return $Poses, LineCorrespondences$
```

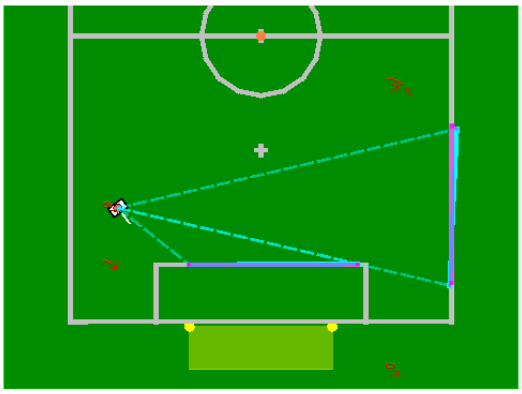
involves an computation analog to algorithm 4, but for the case that only parallel lines are observed, and thus only different continuous intervals of possible poses can be inferred for each valid correspondence choice. In this case, every additional line correspondence serves as a constraint to further restrict the interval given by the first association of line observation with one in the map.

The presented approach does not generate any possible poses or correspondences in case the set of observed lines includes a false positive. In such cases, the whole set of observations is rejected rather than to accept partial fits. While this design choice disregards potentially useful information, the harm of incorporating erroneous data is too sever in these cases which are already confirmed to contain such false information.

This algorithm's result in form of poses and correspondences for the current observations can be utilized in different ways: Either the resulting poses can be used directly as new additional pose hypotheses if the localization algorithm allows this, such as in sensor resetting (cp. section 3.4.2 or 7.3), or each hypothesis in a multi-hypotheses localization system can pick its most likely correspondence to use the set of point feature correspondences in a unified sensor update as in section 5.1.2. This last option is possible directly with point features and a sensor model as described in section 5.1.1, only if perpendicular lines are observed. This is the case, since those perpendicular lines and their associated lines in the map result in a single pose hypothesis as explained above, and consequently in strict correspondences between observed line segments and particular segments of lines in the map, so that sensor updates can be performed directly with all segment endpoints and their correspondences. In case of only parallel observed lines, the translation along the corresponding lines in the map remains uncertain and a different sensor model representation has to be used, which is explained in further detail in appendix B.

Figure A.1 illustrates the output of algorithm 3 for different combinations of observed field lines, which might indicate different poses on the field. As described above, those poses may be discrete as in figures A.1a and A.1b, or continuous as in figure A.1c where only parallel lines are observed. In a multi-hypotheses localization system as in chapter 7, this is supplied to each hypothesis in the localization filter, which picks the best fitting pose as a source of its correspondence choices for the different line observations. Note that the combinatoric results themselves are not used to update the filter, but only to generate correspondences between observed lines and the field line map.


(a) Observation of three lines.


(b) Observation of two perpendicular lines.

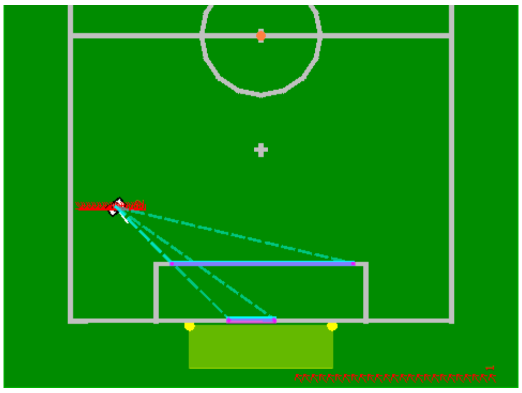

(c) Observation of two parallel lines.

Figure A.1: Output (red poses) of the graph matching algorithm for observations of field lines, and the resulting line correspondences (pink/blue) for the current pose estimation.

Appendix B

Kalman Sensor Update Using Line Observations

As explained in appendix A, observations of single or multiple parallel lines cannot be handled by sensor models for point features as those discussed in chapter 5. This appendix therefore derives a sensor model for Kalman updates with pure line features, i.e. without knowledge about which exact segment is observed.

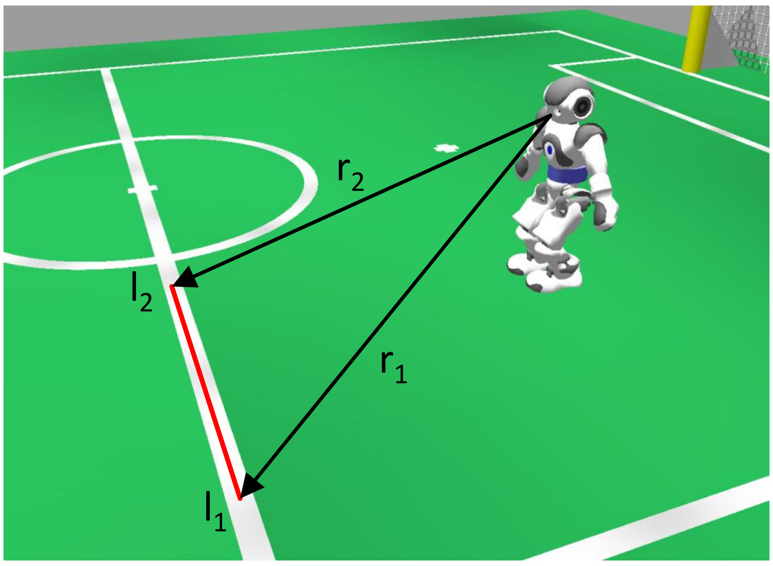


Figure B.1: Line observation.

In projective geometry, the projection of an observed line onto the camera plane can be represented conveniently by the normal vector of the plane given by the observed points on the line and the camera center. An example for a line observation is given in figure B.1. Given a robot pose $x = (p_x, p_y, p_\theta)^T$, the height of the camera h_{camera}, and the start and end points of a line in global coordinates, $l_1 = (l_{x_1}, l_{y_1})^T$ and $l_2 = (l_{x_2}, l_{y_2})^T$ respectively, the expected line observation can be derived as follows. Let r_1 and r_2 be the rays from the camera center to the line end points l_1 and l_2 in world coordinates:

$$r_i = \begin{pmatrix} l_{x_i} - p_x \\ l_{y_i} - p_y \\ -h_{camera} \end{pmatrix}, i \in \{1, 2\}. \tag{B.1}$$

A vector, which is normal to the plane spanned by r_1 and r_2, is thus given by:

$$\begin{aligned} n' &= r_1 \times r_2 \qquad \text{(B.2)} \\ &= \begin{pmatrix} h_{camera}(l_{y_2} - p_y) - h_{camera}(l_{y_1} - p_y) \\ h_{camera}(l_{x_1} - p_x) - h_{camera}(l_{x_2} - p_x) \\ (l_{x_1} - p_x)(l_{y_2} - p_y) - (l_{x_2} - p_x)(l_{y_1} - p_y) \end{pmatrix} \\ &= \begin{pmatrix} -h_{camera} * (l_{y_1} - l_{y_2}) \\ h_{camera} * (l_{x_1} - l_{x_2}) \\ (l_{x_1} - p_x) * (l_{y_2} - p_y) - (l_{x_2} - p_x) * (l_{y_1} - p_y) \end{pmatrix} \end{aligned}$$

Given this, the expected line measurement representation as a unit normal vector in robot coordinates can be derived as in equation B.3

$$\begin{aligned} h(x, h_{camera}, l_1, l_2) &= \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \\ &= \begin{pmatrix} \cos(p_\theta) & \sin(p_\theta) & 0 \\ -\sin(p_\theta) & \cos(p_\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} * \frac{n'}{|n'|} \qquad \text{(B.3)} \\ &= \begin{pmatrix} -h_{camera}(\cos(p_\theta)(l_{y_1} - l_{y_2}) - \sin(p_\theta)(l_{x_1} - l_{x_2})) \\ h_{camera}(\cos(p_\theta)(l_{x_1} - l_{x_2}) + \sin(p_\theta)(l_{y_1} - l_{y_2})) \\ s_1 \end{pmatrix} s_2^{-1/2} \end{aligned}$$

with the abbreviations

$$s_1 = l_{x_1} l_{y_2} - l_{x_2} l_{y_1} - l_{x_1} p_y + l_{y_1} p_x + l_{x_2} p_y - l_{y_2} p_x \tag{B.4}$$

$$s_2 = h^2_{camera}(l_{x_1} - l_{x_2})^2 + h^2_{camera}(l_{y_1} - l_{y_2})^2 + s_1^2. \tag{B.5}$$

The derivatives for the measurement Jacobian are then given by the following equations:

$$H = \begin{pmatrix} \frac{\partial n_x}{\partial p_x} & \frac{\partial n_x}{\partial p_y} & \frac{\partial n_x}{\partial p_\theta} \\ \frac{\partial n_y}{\partial p_x} & \frac{\partial n_y}{\partial p_y} & \frac{\partial n_y}{\partial p_\theta} \\ \frac{\partial n_z}{\partial p_x} & \frac{\partial n_z}{\partial p_y} & \frac{\partial n_z}{\partial p_\theta} \end{pmatrix} \tag{B.6}$$

$$
\begin{aligned}
\frac{\partial n_x}{\partial p_x} &= h_{camera}(l_{y_1} - l_{y_2})(\cos(p_\theta)(l_{y_1} - l_{y_2}) - \sin(p_\theta)(l_{x_1} - l_{x_2}))s_1 && \cdot s_2^{-3/2}\\
\frac{\partial n_y}{\partial p_x} &= -\,h_{camera}(l_{y_1} - l_{y_2})(\cos(p_\theta)(l_{x_1} - l_{x_2}) + \sin(p_\theta)(l_{y_1} - l_{y_2}))s_1 && \cdot s_2^{-3/2}\\
\frac{\partial n_z}{\partial p_x} &= h_{camera}^2(l_{y_1} - l_{y_2})((l_{x_1} - l_{x_2})^2 + (l_{y_1} - l_{y_2})^2) && \cdot s_2^{-3/2}\\
\frac{\partial n_x}{\partial p_y} &= -\,h_{camera}(l_{x_1} - l_{x_2})(\cos(p_\theta)(l_{y_1} - l_{y_2}) - \sin(p_\theta)(l_{x_1} - l_{x_2}))s_1 && \cdot s_2^{-3/2}\\
\frac{\partial n_y}{\partial p_y} &= h_{camera}(l_{x_1} - l_{x_2})(\cos(p_\theta)(l_{x_1} - l_{x_2}) + \sin(p_\theta)(l_{y_1} - l_{y_2}))s_1 && \cdot s_2^{-3/2}\\
\frac{\partial n_z}{\partial p_y} &= \;-\,h_{camera}^2(l_{x_1} - l_{x_2})((l_{x_1} - l_{x_2})^2 + (l_{y_1} - l_{y_2})^2) && \cdot s_2^{-3/2}\\
\frac{\partial n_x}{\partial p_\theta} &= h_{camera}(\cos(p_\theta)(l_{x_1} - l_{x_2}) + \sin(p_\theta)(l_{y_1} - l_{y_2})) && \cdot s_2^{-1/2}\\
\frac{\partial n_y}{\partial p_\theta} &= h_{camera}(\cos(p_\theta)(l_{y_1} - l_{y_2}) - \sin(p_\theta)(l_{x_1} - l_{x_2})) && \cdot s_2^{-1/2}\\
\frac{\partial n_z}{\partial p_\theta} &= 0.
\end{aligned}
$$

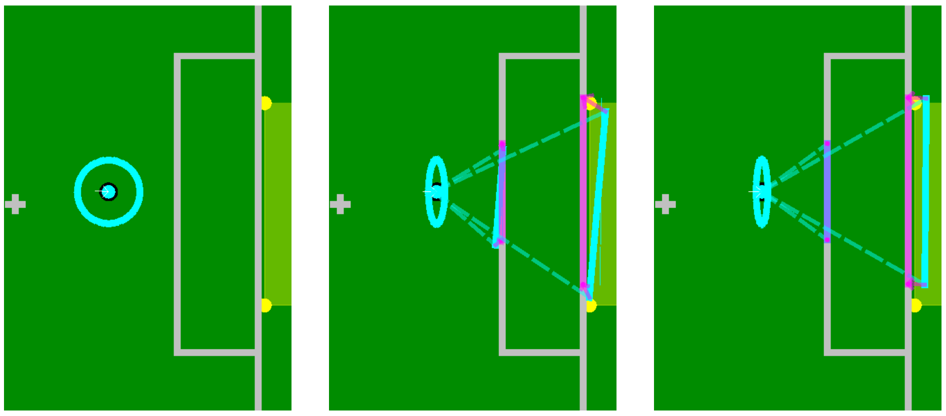

(a) A robot with an isotropic pose uncertainty. (b) Belief after the first observation of two parallel lines. (c) Belief after the second sensor update.

Figure B.2: Localization belief when updating with line observations.

This sensor model can be used for Kalman updates in situations such as the one depicted in figure A.1c, in which only parallel lines are observed. The recommendations in section 5.1.2 are still valid for this sensor model, i.e. that it is beneficial to handle multiple feature observations in a single frame in unified sensor updates. Figure B.2 illustrates the showcase of a robot with an isotropic pose uncertainty, which then observes two parallel lines and corrects its belief accordingly. Note that the uncertainty remains high along the direction of the observed lines. This would not have been the case if sensor update would be done with the line segment's associated end points as point features.

Appendix C

Jacobi Matrix Derivations of Local Tracking

The local tracking measurement equations for observations in angular coordinates, as used in the percept aggregation in section 8.1.3, are given in equations 8.1 and 8.2 and repeated here for convenience:

$$h_{\alpha_1}(\mu) = \operatorname{atan2}(h_{camera}, |\mu'|) \tag{C.1}$$
$$h_{\alpha_2}(\mu) = \operatorname{atan2}(m_y, m_x). \tag{C.2}$$

As defined in section 8.1.3, $\mu = (m_x, m_y, m_{v_x}, m_{v_y})^T$ describes an object's model consisting of a 2-dimensional spacial component and a corresponding velocity component, which are denoted $\mu' = (m_x, m_y)^T$ and $v = (m_{v_x}, m_{v_y})^T$ when used separately.

The measurement Jacobian is then given by

$$H = \frac{\partial h(\mu)}{\partial \mu} = \begin{pmatrix} \frac{\partial h_{\alpha_1}(\mu)}{\partial m_x} & \frac{\partial h_{\alpha_1}(\mu)}{\partial m_y} & \frac{\partial h_{\alpha_1}(\mu)}{\partial m_{v_x}} & \frac{\partial h_{\alpha_1}(\mu)}{\partial m_{v_y}} \\ \frac{\partial h_{\alpha_2}(\mu)}{\partial m_x} & \frac{\partial h_{\alpha_2}(\mu)}{\partial m_y} & \frac{\partial h_{\alpha_2}(\mu)}{\partial m_{v_x}} & \frac{\partial h_{\alpha_2}(\mu)}{\partial m_{v_y}} \end{pmatrix} \tag{C.3}$$

for which the separate entries can be derived as shown here. Note that the step from equation C.4 to C.5 assumes that the target is not inside the robot itself, i.e. that $|\mu'| \neq 0$. This is the case for the given application.

$$
\begin{aligned}
\frac{\partial h_{\alpha_1}(\mu)}{\partial m_x} &= \frac{\partial \text{atan2}(h_{camera}, |\mu'|)}{\partial m_x} && \text{(C.4)}\\
&= \frac{\partial \text{atan}(\frac{h_{camera}}{|\mu'|})}{\partial m_x} && \text{(C.5)}\\
&= \frac{1}{\frac{h_{camera}^2}{|\mu'|^2}+1} \cdot \frac{\partial \frac{h_{camera}}{|\mu'|}}{\partial m_x} && \text{(C.6)}\\
&= \frac{1}{\frac{h_{camera}^2}{|\mu'|^2}+1} \cdot -h_{camera} \cdot \frac{1}{|\mu'|^2} \cdot \frac{1}{2}\frac{1}{|\mu'|} \cdot 2m_x && \text{(C.7)}\\
&= -\frac{h_{camera}\, m_x}{|\mu'|^3 + h_{camera}^2\, |\mu'|} && \text{(C.8)}
\end{aligned}
$$

The derivative for m_y follows analogically:

$$
\begin{aligned}
\frac{\partial h_{\alpha_1}(\mu)}{\partial m_y} &= \frac{\partial \text{atan2}(h_{camera}, |\mu'|)}{\partial m_y} && \text{(C.9)}\\
&= -\frac{h_{camera}\, m_y}{|\mu'|^3 + h_{camera}^2\, |\mu'|}. && \text{(C.10)}
\end{aligned}
$$

For the horizontal angle α_2 their is a singularity for observations behind the robot, i.e. from $-\pi$ to π. For the Nao robot, this can be neglected, as its head can not turn enough to observe anything in this angle, but otherwise this angle periodicity has to be considered and handled appropriately. The presented derivation assumes in the step from equation C.11 to C.12 that the target is not exactly behind the robot, i.e. $m_x > 0$ or $m_y \neq 0$.

$$
\begin{aligned}
\frac{\partial h_{\alpha_2}(\mu)}{\partial m_x} &= \frac{\partial \text{atan2}(m_y, m_x)}{\partial m_x} && \text{(C.11)}\\
&= \frac{\partial \text{atan}(\frac{m_y}{m_x})}{\partial m_x} && \text{(C.12)}\\
&= \frac{1}{\frac{m_y^2}{m_x^2}+1} \cdot \frac{\partial \frac{m_y}{m_x}}{\partial m_x} && \text{(C.13)}\\
&= \frac{1}{\frac{m_y^2}{m_x^2}+1} \cdot -\frac{m_y}{m_x^2} && \text{(C.14)}\\
&= -\frac{m_y}{m_x^2 + m_y^2} && \text{(C.15)}
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial h_{\alpha_2}(\mu)}{\partial m_x} &= \frac{\partial \mathrm{atan2}(m_y, m_x)}{\partial m_x} && \text{(C.16)}\\
&= \frac{\partial \mathrm{atan}(\frac{m_y}{m_x})}{\partial m_y} && \text{(C.17)}\\
&= \frac{1}{\frac{m_y^2}{m_x^2}+1} \cdot \frac{\partial \frac{m_y}{m_x}}{\partial m_y} && \text{(C.18)}\\
&= \frac{1}{\frac{m_y^2}{m_x^2}+1} \cdot \frac{1}{m_x} && \text{(C.19)}\\
&= \frac{m_x}{m_x^2+m_y^2} && \text{(C.20)}
\end{aligned}$$

Thus follows the resulting measurement Jacobian presented in equation 8.3.

The derivation of the Jacobian F for the process model $f(\mu_{t-1})$ in equation 8.6 is simplified by the fact that for a linear function $f(x) = Ax$, where A is constant (and thus independent of x), the Jacobian is $F = A$. Equation 8.6 can be rearranged as in equation C.21, so that it is obvious that this applies to parts of the Jacobian.

$$\overline{\mu}_t = f(\mu_{t-1}) = \begin{pmatrix} \Omega(-\Delta_\theta) & \Delta_t\Omega(-\Delta_\theta) \\ 0 & V \end{pmatrix} \mu_{t-1} - \begin{pmatrix} \Delta_x \\ \Delta_y \\ 0 \\ 0 \end{pmatrix} \qquad \text{(C.21)}$$

$$\begin{aligned}
&= \begin{pmatrix} \Omega(-\Delta_\theta) & \Delta_t\Omega(-\Delta_\theta) \\ 0 & 0 \end{pmatrix} \mu_{t-1} \\
&\quad + \begin{pmatrix} 0 & 0 \\ 0 & V \end{pmatrix} \mu_{t-1} - \begin{pmatrix} \Delta_x \\ \Delta_y \\ 0 \\ 0 \end{pmatrix} && \text{(C.22)}
\end{aligned}$$

In the process model Jacobian in equation 8.8, all but the $\frac{\partial f_{m_{v_{x/y}}}(\mu_{t-1})}{\partial m_{v_{x/y}}}$ part is linear and thus easily derived:

$$\begin{aligned}
F &= \frac{\partial f(\mu_{t-1})}{\partial \mu} && \text{(C.23)}\\
&= \begin{pmatrix}
\frac{\partial f_{m_x}(\mu_{t-1})}{\partial m_x} & \frac{\partial f_{m_x}(\mu_{t-1})}{\partial m_y} & \frac{\partial f_{m_x}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_x}(\mu_{t-1})}{\partial m_{v_y}}\\
\frac{\partial f_{m_y}(\mu_{t-1})}{\partial m_x} & \frac{\partial f_{m_y}(\mu_{t-1})}{\partial m_y} & \frac{\partial f_{m_y}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_y}(\mu_{t-1})}{\partial m_{v_y}}\\
\frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_x} & \frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_y} & \frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_{v_y}}\\
\frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_x} & \frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_y} & \frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_{v_y}}
\end{pmatrix} && \text{(C.24)}\\
&= \begin{pmatrix}
\Omega(-\Delta_\theta) & \Delta_t\Omega(-\Delta_\theta) & \\
0 & \frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_{v_x}}(\mu_{t-1})}{\partial m_{v_y}}\\
 & \frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_{v_x}} & \frac{\partial f_{m_{v_y}}(\mu_{t-1})}{\partial m_{v_y}}
\end{pmatrix} && \text{(C.25)}
\end{aligned}$$

For $|v_{t-1}| < |k\Delta_t|$, this last part of the Jacobian is zero. Otherwise, these last partial derivatives are calculated as shown in equation C.26 (cp. equation 8.9). Equations 8.10 to 8.12 can be derived similarly.

$$\begin{aligned}
\frac{\partial f_{m_{v_x}}}{\partial m_{v_x}} &= \frac{\partial\Big[\Big(1+\frac{\phi\ \Delta_t}{|v|}\Big)\big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)\Big]}{\partial m_{v_x}} && \text{(C.26)}\\
&= \left(1+\frac{\phi\ \Delta_t}{|v|}\right)\frac{\partial\big[\big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)\big]}{\partial m_{v_x}}\\
&\quad + \frac{\partial\Big[\Big(1+\frac{\phi\ \Delta_t}{|v|}\Big)\Big]}{\partial m_{v_x}}\big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)\\
&= \left(1+\frac{\phi\ \Delta_t}{|v|}\right)\cos(-\Delta_\theta)\\
&\quad + \frac{\partial\left[\frac{1}{\sqrt{m_{v_x}^2+m_{v_y}^2}}\right]}{\partial m_{v_x}}\ \phi\ \Delta_t\ \big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)\\
&= \left(1+\frac{\phi\ \Delta_t}{|v|}\right)\cos(-\Delta_\theta)\\
&\quad - \frac{1}{2\sqrt{m_{v_x}^2+m_{v_y}^2}^3}\ 2m_{v_x}\ \phi\ \Delta_t\ \big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)\\
&=\\
&= \left(1+\frac{\phi\ \Delta_t}{|v|}\right)\cos(-\Delta_\theta)\\
&\quad - \frac{\phi\ \Delta_t\ m_{v_x}\big(\cos(-\Delta_\theta)m_{v_x}-\sin(-\Delta_\theta)m_{v_y}\big)}{|v|^3}
\end{aligned}$$

List of Figures

List of Symbols

t	Specifies a discrete moment in time, page 23
$t_1 : t_2$	Specifies the time interval of all discrete moments between (and including) t_1 and t_2, page 25
x_t	The state x at time t, page 23
n	Dimensionality of the state x, page 23
m_i	One element in the environment map m, page 24
$(m_{i,x}, m_{i,y})^T$	The 2-dimensional position of an object m_i, which is an element of a map m, page 24
$v = (m_{i,v_x}, m_{i,v_y})^T$	The 2-dimensional velocity of an object m_i, which is an element of a map m, page 24
z_t	An observation z made at time t, page 25
k	Dimensionality of the measurement z, page 25
u_t	A motion control u made at time t, page 25
$(p_x, p_y, p_\theta)^T$	The robot's pose given by its 2-dimensional position and its orientation, page 23
$bel(x_t)$	A robot's belief or knowledge about the state x_t, page 26
$\overline{bel}(x_t)$	A robot's prediction about the state x_t, based on information prior to t and on the current control data u_t, page 26
$p(a\|b)$	The conditional probability of event a given event b, page 26
$N(\mu, P)$	Gaussian probability distribution with mean μ and covariance P, page 29
μ	The mean of a Gaussian probability distribution, page 29
P	The covariance of a Gaussian probability distribution, page 29

ϵ	The noise term involved in the process or motion model uncertainty; often assumed to be Gaussian distributed with zero mean and covariance R, i.e. $N(0, R)$, page 30
R	The covariance of a Gaussian process noise term, page 30
δ	The noise term involved in the measurement or sensor model uncertainty; often assumed to be Gaussian distributed with zero mean and covariance Q, i.e. $N(0, Q)$, page 30
Q	The covariance of a Gaussian measurement noise term, page 30
$f(x_{t-1}, u_t)$	The process or motion model $f()$ describes the prediction of $\overline{bel}(x_t)$ based on $bel(x_{t-1})$ and u_t, page 31
$h(x_t)$	The measurement or sensor model $h()$ describes the prediction of a measurement based on the current belief state $bel(x_t)$, page 31
F	The Jacobian matrix of partial derivatives of the process model $f()$, page 32
H	The Jacobian matrix of partial derivatives of the sensor model $h()$, page 32
K	The Kalman gain, page 32
$\mathcal{X}, \mathcal{W}$	A list of elements (vectors or numbers), e.g. sigma points, particles, or weights, page 33
$\mathcal{X}^{[i]}, \mathcal{W}^{[i]}$	One element in the list, page 33
$S(p(x))$	The entropy of a probability distribution $p(x)$, page 93
$E(p(x))$	The expected value based on a probability distribution $p(x)$, page 95
m	A map of the environment, page 23
Ω	A rotation matrix, page 83

References

[1] D. Alspach and H. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximations. *Automatic Control, IEEE Transactions on*, 17(4):439 – 448, August 1972.

[2] T. Arbel and F. Ferrie. Entropy-based Gaze Planning. In *Proceedings of the Second IEEE Workshop on Perception for Mobile Agents*, 1999.

[3] C. Beder and M. Klepal. Predicting the Expected Accuracy for Fingerprinting-based WiFi Localisation Systems. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation*, IPIN, Guimarães, Portugal, September 2011.

[4] M. Betke and L. Gurvits. Mobile Robot Localization using Landmarks. *Robotics and Automation, IEEE Transactions on*, 13(2):251 –263, April 1997.

[5] W. Burgard, D. Fox, and S. Thrun. Active Mobile Robot Localization. In *Proceedings of IJCAI-97. IJCAI, Inc.* Morgan Kaufmann, 1997.

[6] H. I. Christensen and G. D. Hager. Sensing and Estimation. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 87–107. Springer Berlin Heidelberg, 2008.

[7] D.E. Clark, K. Panta, and B.-N. Vo. The GM-PHD Filter Multiple Target Tracker. In *Information Fusion, 2006 9th International Conference on*, pages 1 –8, july 2006.

[8] S. Czarnetzki, D. Hauschildt, S. Kerner, and O. Urbann. BreDoBrothers Team Report for RoboCup 2008. Technical report, Robotics Research Institute, TU Dortmund University, 2009.

[9] S. Czarnetzki, M. Hegele, and S. Kerner. Odometry Correction for Humanoid Robots Using Optical Sensors. In Javier Ruiz-del Solar, Eric Chown, and Paul Plöger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 48–59. Springer Berlin / Heidelberg, 2011.

[10] S. Czarnetzki, S. Kerner, and D. Klagges. Combining Key Frame Based Motion Design with Controlled Movement Execution. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 58–68. Springer Berlin / Heidelberg, 2010.

[11] S. Czarnetzki, S. Kerner, and M. Kruse. Real-Time Active Vision by Entropy Minimization Applied to Localization. In Javier Ruiz-del Solar, Eric Chown, and Paul Plöger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 266–277. Springer Berlin / Heidelberg, 2011.

[12] S. Czarnetzki, S. Kerner, and P. Szcypior. Learning from Demonstration - Automatic Generation of Extended Behavior Networks for Autonomous Robots from an Expert's Demonstration. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 394–400. INSTICC Press, 2011.

[13] S. Czarnetzki, S. Kerner, and O. Urbann. Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems*, 57(8):839 – 845, 2009. Humanoid Soccer Robots.

[14] S. Czarnetzki, S. Kerner, and O. Urbann. Applying Dynamic Walking Control for Biped Robots. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 69–80. Springer Berlin / Heidelberg, 2010.

[15] S. Czarnetzki, S. Kerner, O. Urbann, J. Abelev, C. Bökkerink, M. Hofmann, A. Ibisch, R. Jalali, M. Kuhnert, M. Ollendorf, P. Schorlemmer, S. Stumm, and M.-A. Weber. Nao Devils Dortmund Team Report for RoboCup 2009. Technical report, Robotics Research Institute, TU Dortmund University, 2009.

[16] S. Czarnetzki, S. Kerner, O. Urbann, M. Hofmann, S. Stumm, and I. Schwarz. Nao Devils Dortmund Team Report 2010. Technical report, Robotics Research Institute, TU Dortmund University, 2010.

[17] S. Czarnetzki and C. Rohde. Handling Heterogeneous Information Sources for Multi-Robot Sensor Fusion. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010)*, pages 133 – 138, Salt Lake City, Utah, September 2010.

[18] A. J. Davison, W. W. Mayol, and D. W. Murray. Real-Time Localisation and Mapping with Wearable Active Vision. In *ISMAR '03:*

Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, page 18, Washington, DC, USA, 2003. IEEE Computer Society.

[19] A.G. Dempster, Binghao Li, and I. Quader. Errors in Deterministic Wireless Fingerprinting Systems for Localisation. In *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, pages 111 –115, May 2008.

[20] M. Dietl, J.-S. Gutmann, and B. Nebel. Cooperative Sensing in Dynamic Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1706–1713, 2001.

[21] T. Duckett and U. Nehmzow. Mobile robot self-localisation using occupancy histograms and a mixture of Gaussian location hypotheses. *Robotics and Autonomous Systems*, 34(2-3):117 – 129, 2001.

[22] G. Dudek and M. Jenkin. Inertial Sensors, GPS, and Odometry. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 477–490. Springer Berlin Heidelberg, 2008.

[23] J. Eggert. *Empirische Entwicklung eines applikationsadaptiven Antriebskonzeptes am Beispiel eines intralogistischen Fördersystems.* PhD thesis, TU Dortmund University, 2011.

[24] J. Eggert, D. Wieczorek, and B. Künne. *Zustandsbeurteilung in der Intralogistik mit Hilfe von verteilten und stationären Messeinheiten.* Internationales Forum Mechatronik 2009, Linz, Austria, 2009.

[25] N. Fairfield and D. Wettergreen. Active Localization on the Ocean Floor With Multibeam Sonar. In *Proceedings of MTS/IEEE OCEANS*, 2008.

[26] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.

[27] D. Fox, W. Burgard, and S. Thrun. Active Markov Localization for Mobile Robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[28] D. Göhring, H. Mellmann, and H.-D. Burkhard. Constraint Based World Modeling in Mobile Robotics. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2009*, pages 2538–2543, 2009.

[29] A. Goswami. Foot Rotation Indicator (FRI) point: A New Gait Planning ToolltoEvaluate Postural Stability of Biped Robots. In *IEEE International Conference on Robotics and Automation*, pages 47–52, 1999.

[30] D. Gouaillier, C. Collette, and C. Kilner. Omni-directional closed-loop walk for NAO. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 448 –454, dec. 2010.

[31] C. Graf, A. Härtl, T. Röfer, and T. Laue. A Robust Closed-Loop Gait for the Standard Platform League Humanoid. In Changjiu Zhou, Enrico Pagello, Emanuele Menegatti, Sven Behnke, and Thomas Röfer, editors, *Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots*, pages 30 – 37, Paris, France, 2009.

[32] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 454 – 459 vol.1, 2002.

[33] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom '04, pages 70–84, New York, NY, USA, 2004. ACM.

[34] D. Hähnel, D. Schulz, and W. Burgard. Map Building with Mobile Robots in Populated Environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[35] D. Hauschildt, S. Kerner, S. Tasse, and O. Urbann. Multi Body Kalman Filtering with Articulation Constraints for Humanoid Robot Pose and Motion Estimation. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 415–426. Springer Berlin / Heidelberg, 2012.

[36] G. W. Hein. From GPS and GLONASS via EGNOS to Galileo - Positioning and Navigation in the Third Millennium. *GPS Solutions*, 3:39–47, 2000.

[37] M. Hernández, J. Cabrera, A. C. Domínguez, M. C. Santana, C. Guerra, D. Hernández-Sosa, and J. Isern. DESEO: An Active Vision System for Detection, Tracking and Recognition. In *ICVS*, pages 376–391, 1999.

[38] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The Development of Honda Humanoid Robot. In *International Conference on Robotics and Automation (ICRA)*, pages 1321–1326, 1998.

[39] A. Howard. Multi-robot Simultaneous Localization and Mapping using Particle Filters. *The International Journal of Robotics Research*, 25(12):1243–1256, December 2006.

[40] Q. Huang, S. Kajita, N. Koyachi, K. Kaneko, K. Yokoi, H. Arai, K. Komoriya, and K. Tanie. A High Stability, Smooth Walking Pattern for a Biped Robot. In *International Conference on Robotics and Automation (ICRA)*, pages 65–, 1999.

[41] G. Jochmann, S. Kerner, S. Tasse, and O. Urbann. Efficient Multi-Hypotheses Unscented Kalman Filtering for Robust Localization. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 222–233. Springer Berlin / Heidelberg, 2012.

[42] S. J. Julier and J. K. Uhlmann. New Extension of the Kalman Filter to Nonlinear Systems. In Ivan Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193. SPIE, 1997.

[43] S.J. Julier and J.K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401 – 422, March 2004.

[44] S. Kajita and B. Espiau. Legged Robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 361–389. Springer Berlin Heidelberg, 2008.

[45] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa. A Realtime Pattern Generator for Biped Walking. In *International Conference on Robotics and Automation (ICRA)*, pages 31–37, 2002.

[46] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[47] R. Kaune. Gaussian Mixture (GM) Passive Localization using Time Difference of Arrival (TDOA). In Stefan Fischer, Erik Maehle, and Rüdiger Reischuk, editors, *GI Jahrestagung*, volume 154 of *LNI*, pages 2375–2381. GI, 2009.

[48] E. Kiel. *Drive Solutions: Mechatronics for Production and Logistics.* Springer Verlag, Berlin, Heidelberg, Germany, 2008.

[49] J.-Y. Kim, I.-W. Park, and J.-H. Oh. Experimental Realization of Dynamic Walking of Biped Humanoid Robot KHR-2 using ZMP Feedback and Inertial Measurement. *Advanced Robotics*, 20(6):707 – 736, June 2006.

[50] T. Koshizen, P. Bartlett, and A. Zelinsky. Sensor Fusion of Odometry and Sonar Sensors by the Gaussian Mixture Bayes' Technique in Mobile Robot Position Estimation. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 4, pages 742 –747 vol.4, 1999.

[51] P. Krauthausen and U. D. Hanebeck. Regularized Non-Parametric Multivariate Density and Conditional Density Estimation. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010)*, pages 180 – 186, Salt Lake City, Utah, September 2010.

[52] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large Scale Graph-based SLAM using Aerial Images as Prior Information. In *Proceedings of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, June 2009.

[53] B. Künne, J. Eggert, and S. Czarnetzki. Condition Monitoring in Intralogistic Systems by the Utilization of Mobile Measurement Units. In *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL 2010)*, Hong Kong, China, August 2010. IEEE.

[54] B. Künne and J. Eggert (a). *Vergleich von simulierten Daten und realen Daten ausgewählter Belastungsprofile eines intralogistischen Transportsystems.* Belastungsabhängige Auslegung, Überwachung und Steuerung von intralogistischen Systemen; Verlag Praxiswissen, Dortmund, Germany, 2008.

[55] B. Künne and J. Eggert (b). *Belastungsprofile eines intralogistischen Fördersystems auf der Basis von Nutzungsprofilen.* TU Dortmund University, Dortmund, Germany, 2008.

[56] T. Laue, K. Spiess, and T. Röfer. SimRobot - A General Physical Robot Simulator and its Application in RoboCup. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, number 4020 in Lecture Notes in Artificial Intelligence, pages 173–183. Springer, 2006.

[57] S. Lenser and M. Veloso. Sensor Resetting Localization for Poorly Modelled Mobile Robots. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1225 –1232 vol.2, 2000.

[58] M. Lötzsch, M. Risler, and M. Jüngel. XABSL - A Pragmatic Approach to Behavior Engineering . In *Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 5124–5129, Beijing, China, 2006.

[59] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy. Precise Indoor Localization using Smart Phones. In *Proceedings of the International Conference on Multimedia*, MM, pages 787–790, Firenze, Italy, October 2010. ACM, ACM Press.

[60] J. Minguez, F. Lamiraux, and J.-P. Laumond. Motion Planning and Obstacle Avoidance. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 827–852. Springer Berlin Heidelberg, 2008.

[61] P. Misra and P. Enge. *Global Positioning System: Signals, Measurements, and Performance.* Ganga-Jamuna Press, 2006.

[62] M. Montemerlo and S. Thrun. Large-Scale Robotic 3-D Mapping of Urban Structures. In Jr. Ang, MarceloH. and Oussama Khatib, editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 141–150. Springer Berlin Heidelberg, 2006.

[63] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[64] National Research Council (U.S.). Committee on the Future of the Global Positioning System and National Academy Of Public Administration. *The Global Positioning System: A Shared National Asset: Recommendations for Technical Improvements and Enhancements.* National Academy Press, 1995.

[65] W. Nisticò and M. Hebbel. Temporal Smoothing Particle Filter for Vision Based Autonomous Mobile Robot Localization. In *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume RA-1, pages 93–100. INSTICC Press, 2008.

[66] W. Nisticò and M. Hebbel. Particle Filter with Temporal Smoothing for Mobile Robot Vision-Based Localization. In Juan Andrade Cetto, Jean-Louis Ferrier, and Joaquim Filipe, editors, *Informatics in Control, Automation and Robotics*, volume 37 of *Lecture Notes in Electrical Engineering*, pages 167–180. Springer Berlin Heidelberg, 2009.

[67] W. Y. Ochieng and K. Sauer. Urban Road Transport Navigation: Performance of the Global Positioning System after Selective Availability. *Transportation Research Part C: Emerging Technologies*, 10(3):171–187, June 2002.

[68] P. Pfaff, C. Plagemann, and W. Burgard. Gaussian Mixture Models for Probabilistic Localization. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 467 –472, May 2008.

[69] J. M. Porta, B. Terwijn, and B. Kröse. Efficient Entropy-Based Action Selection for Appearance-Based Robot Localization. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2842–2847, 2003.

[70] M. Quinlan and R. Middleton. Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 276–287. Springer Berlin / Heidelberg, 2010.

[71] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2005.

[72] T. Röfer, J. Brose, D. Göhring, M. Jüngel, T. Laue, and M. Risler. GermanTeam 2007 - The German national RoboCup team. In *In RoboCup 2007: Robot Soccer World Cup XI Preproceedings. RoboCup Federation*, 2007.

[73] T. Röfer, T. Laue, M. Weber, H. D. Burkhard, M. Jüngel, D. Göhring, J. Hoffmann, B. Altmeyer, T. Krause, M. Spranger, O. v. Stryk, R. Brunn, M. Dassler, M. Kunz, T. Oberlies, M. Risler, U. Schwiegelshohn, M. Hebbel, W. Nisticò, S. Czarnetzki, T. Kerkhof, M. Meyer, C. Rohde, B. Schmitz, M. Wachter, T. Wegner, and C. Zarges. GermanTeam RoboCup 2005. Technical report, Universität Bremen, Humboldt-Universität zu Berlin, TU Darmstadt, Universität Dortmund, 2005.

[74] A. Roxin, J. Gaber, M. Wack, and A. Nait-Sidi-Moh. Survey of Wireless Geolocation Techniques. In *Globecom Workshops, 2007 IEEE*, pages 1 –9, Nov. 2007.

[75] D. B. Rubin. Using the SIR Algorithm to Simulate Posterior Distributions. In M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 3.* Oxford University Press, 1988.

[76] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. Tracking Multiple Moving Objects with a Mobile Robot. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:371, 2001.

[77] O. Serrano, J. M. Canas, V. Matellan, J. M., C. V. Matellán, and L. Rodero. Robot Localization using WiFi Signal without Intensity Map. In *Proceedings of the V. Workshop on Physical Agents*, 2004.

[78] R. Siegwart and I.R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents. The MIT Press, 2004.

[79] P. W. Singer. *Wired for War: The Robotics Revolution and Conflict in the 21st Century*. The Penguin Press HC, January 2009.

[80] J. Small, A. Smailagic, and D. P. Siewiorek. Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure. Technical Report 040201, Institute for Complex Engineered Systems at Carnegie Mellon University, December 2000.

[81] P. Smith, I. Reid, and A. Davison. Real-Time Monocular SLAM with Straight Lines. In *Proc. British Machine Vision Conference*, Edinburgh, 2006.

[82] J. Strom, G. Slavov, and E. Chown. Omnidirectional Walking Using ZMP and Preview Control for the NAO Humanoid Robot. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 378–389. Springer Berlin / Heidelberg, 2010.

[83] A. Stroupe, M. Matrin, and T. Balch. Distributed Sensor Fusion for Object Position Estimation by Multi-Robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2001*, 2001.

[84] S. Tasse, M. Hofmann, and O. Urbann. On Sensor Model Design Choices for Humanoid Robot Localization. In Xiaoping Chen, Peter Stone, LuisEnrique Sucar, and Tijn Zant, editors, *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 380–390. Springer Berlin Heidelberg, 2013.

[85] S. Tasse, M. Hofmann, and O. Urbann. SLAM in the Dynamic Context of Robot Soccer Games. In Xiaoping Chen, Peter Stone, LuisEnrique Sucar, and Tijn Zant, editors, *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 368–379. Springer Berlin Heidelberg, 2013.

[86] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[87] S. Thrun, D. Fox, and W. Burgard. Monte Carlo Localization with Mixture Proposal Distribution. In *Proc. of the National Conference on Artificial Intelligence*, 2000.

[88] B. Upcroft, S. Kumar, M. Ridley, L. L. Ong, and H. Durrant-whyte. Fast Re-Parameterisation of Gaussian Mixture Models for Robotics Applications. In Nick Barnes and David Austin, editors, *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*. Australian Robotics and Automation Association, 2004.

[89] O. Urbann, S. Kerner, and S. Tasse. Rigid and Soft Body Simulation Featuring Realistic Walk Behaviour. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranli, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 126–136. Springer Berlin / Heidelberg, 2012.

[90] R. Van der Merwe and E.A. Wan. The Square-Root Unscented Kalman Filter for State and Parameter-Estimation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 6, pages 3461 –3464 vol.6, 2001.

[91] M. Vukobratović and B. Borovac. Zero-moment Point – Thirty Five Years of its life. *International Journal of Humanoid Robotics*, 1(1):157–173, 2004.

[92] M. Vukobratović, B. Borovac, and V. Potkonjak. Towards a Unified Understanding of Basic Notions and Terms in Humanoid Robotics. *Robotica*, 25(1):87–101, 2007.

[93] M. G. Wing, A. Eklund, and L. D. Kellogg. Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability. *Journal of Forestry*, 103(4):169–173, June 2005.

[94] J. Yamaguchi, E. Soga, S. Inoue, and A. Takanishi. Development of a Bipedal Humanoid Robot: Control Method of Whole Body Cooperative Dynamic Biped Walking. In *International Conference on Robotics and Automation (ICRA)*, pages 368–374, 1999.

[95] Y. F. Zheng and J. Shen. Gait Synthesis for the SD-2 Biped Robot to Climb Sloping Surface. *IEEE Journal of Robotics and Automation*, 6(1):86–96, 1990.

[96] X. S. Zhou and S. I. Roumeliotis. Multi-Robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1785–1792, Beijing, China, October 2006.