UNIVERSITÄT ZU LÜBECK

From the Institute of Medical Informatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Heinz Handels

"Advanced Sensor Fusion Methods with Applications to Localization and Navigation"

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by

Toni Fetzer
from Rothenburg ob der Tauber

Lübeck 2024

First referee:	Prof. Dr. Marcin Grzegorzek
Second referee:	Prof. Dr. Georg Schildbach
Date of oral examination:	27.09.2024
Approved for printing:	Lübeck, 18.03.2025

Acknowledgements

This work would not have been possible without the incredible support of universities, colleagues, friends, and family. My academic career began during my bachelor studies at the Institute for Design and Information Systems at the University of Applied Sciences Würzburg-Schweinfurt (THWS). My supervisor Prof. Dr. Frank Deinzer, the head of the institute Prof. Dr. Karsten Huffstadt, the deans Prof. Dr. Michael Müßig and Prof. Dr.-Ing. Peter Braun as well as all responsible persons in the Faculty of Computer Science and Business Information Systems supported by in a way that was a novelty at the THWS at that time. I was able to present my work worldwide, teach hundreds of students the basics of computer science, and learn from incredibly brilliant minds. I will always be grateful to THWS and its staff for this privilege.

Anyone who does science knows that top-notch research can only be done as part of a team. The fact that I have received so many prizes and awards during my scientific life is undoubtedly due to my colleagues. My special thanks go to Dr. Frank Ebner and Markus Bullmann. Frank has been a driving force behind my research from the very beginning and has significantly influenced the experiments in this thesis with his countless software tools. Anyone who knows me is aware that patience is not necessarily my greatest strength, so the often very spontaneous phone calls with Markus about new approaches, methods, or all the other topics that such a PhD entails have a very special value for me. I would also like to take this opportunity to thank all my other companions like Lukas, Marcel, Seb, Max, André, Matthias, Robin, Markus E., Steffen, and many others.

Behind a strong research team there are also strong institutions and their heads. With the Medical Data Science Lab of Prof. Dr. habil. Marcin Grzegorezk and the Center for Artificial Intelligence and Robotics of Prof. Dr. Magda Gregorová, I had an environment that offered me every opportunity. I would also like to thank all the institutions, foundations and companies that supported us financially and with data sets. These include the City of Rothenburg, the Fraunhofer Institute for Manufacturing Engineering and Automation, the Landesstelle für die nichtstaatlichen Museen in Bayern, the Sparkassenstiftung, the City of Würzburg, and many more. Probably the biggest influence on my research, career and professional development was my PhD supervisor Prof. Dr. Frank Deinzer and his Steinbeis Transfer Center New Media and Data Science. As a project manager there, I transferred research directly into practice, a process that requires a whole range of skills. Thank you, Frank, for teaching me all this with the necessary trust and patience.

Anyone who has written a thesis knows it has no office hours. It always hangs over you like the sword of Damocles. Without the backing and support of my family, I probably would have

crumbled under the pressure. I would like to thank and apologize to my wife Alena and my daughter Fenna. They had to spend countless evenings, weekends and even holidays without their husband and father. Thanks also to all my friends who kept reminding me that I had to write this thesis, what would I be without you. Last but not least, I would like to thank my parents Andrea and Fritz, who have supported me in everything I have done, my brother Peter, who has always been there to help and advise me, and my sister Eva, on whom I can rely in all situations.

To my grandma Inge, who has always shown us grandchildren nothing but love.

Abstract

We use sensors to track how many steps we take during the day or how well we sleep. Sensor fusion methods are used to draw these conclusions. A particularly difficult application is indoor localization, i.e. finding a person's position within a building. This is mainly due to the many degrees of freedom of human movement and the physical properties of sensors inside buildings. Suitable approaches for sensor fusion for the purpose of self-localization using a smartphone are the subject of this thesis.

To best address the complexity of this problem, a non-linear and non-Gaussian distributed state space must be assumed. For the required position estimation, we therefore focus on the class of particle filters and build a novel generic filter framework on top of it. The special feature of this framework is the modular approach and the low requirements towards the sensor and movement models. In this work, we investigate models for Wi-Fi and Bluetooth RSSI measurements using radio propagation models, the relatively new standard Wi-Fi FTM, which is explicitly designed for localization purposes, the barometer to determine floor changes as accurately as possible, and activity recognition to find out what the pedestrian is doing, e.g., ascending stairs. The human motion is then modeled in a movement model using IMU data. Here we propose two approaches: a regular tessellated grid graph and an irregular tessellated navigation mesh.

From these we formulate our proposal for an *indoor localization system* (ILS). However, some fundamental problems of the particle filter lead to critical errors. These can be a multi-modal density to be estimated, unbalanced sensor models or the so-called sample impoverishment. Compensation, or in the best case elimination, of these errors by advanced sensor fusion methods is the main contribution of this thesis. The most important approach in this context is our adaptation of an *interacting multiple modal particle filter* (IMMPF) to the requirements of indoor localization. This results in a completely new approach to the formulation of an ILS. Using quality metrics, it is possible to dynamically switch between arbitrarily formulated particle filters running in parallel. Furthermore, we explicitly propose several approaches from the field of *particle distribution optimization* (PDO) to avoid the sample impoverishment problem. In particular, the *support filter approach* (SFA), which is also based on the IMMPF principle, leads to excellent position estimates even under the most difficult conditions, as extensive experiments show.

Zusammenfassung

Mit Hilfe von Sensoren verfolgen wir, wie viele Schritte wir am Tag gehen oder wie gut wir schlafen. Um diese Schlüsse zu ziehen, werden Methoden der Sensordatenfusion eingesetzt. Ein besonders schwieriger Anwendungsfall ist die Indoor-Lokalisierung, also das Auffinden der eigenen Position innerhalb eines Gebäudes. Dies liegt vor allem an den vielen Freiheitsgraden der menschlichen Bewegung und den physikalischen Eigenschaften der Sensoren innerhalb von Gebäuden. Geeignete Ansätze der Sensordatenfusion zum Zweck der Selbstlokalisierung mittels Smartphone sind Gegenstand dieser Arbeit.

Um der Komplexität dieser Problemstellung bestmöglich gerecht zu werden, muss von einem nichtlinearen und nichtgaußverteilten Zustandsraum ausgegangen werden. Für die erforderliche Positionsschätzung konzentrieren wir uns daher auf die Klasse der Partikelfilter und bauen darauf ein neuartiges generisches Filterframework auf. Das Besondere daran ist der modulare Ansatz und die geringen Anforderungen an die Sensor- und Bewegungsmodelle. Im Rahmen dieser Arbeit untersuchen wir Modelle für Wi-Fi und Bluetooth RSSI Messungen mittels Radiopropagationsmodellen, den relativ neuen Standard Wi-Fi FTM, der explizit für Lokalisierungszwecke entwickelt wurde, das Barometer, um möglichst genaue Stockwerkwechsel bestimmen zu können und eine Aktivitätserkennung, um herauszufinden, was der Fußgänger gerade tut, z.B. Treppen steigen. Die menschliche Bewegung wird dann in einem Bewegungsmodell mit Hilfe von IMU-Daten modelliert. Hier schlagen wir zwei Ansätze vor: ein gleichmäßiges tessellierter Gittergraph und ein unregelmäßiges tesselliertes Navigationsnetz.

Daraus formulieren wir unseren Vorschlag für ein *Indoor-Lokalisierungssystem* (ILS). Einige grundlegende Probleme des Partikelfilters führen jedoch zu kritischen Fehlern. Diese können eine zu schätzende Dichte sein, die multimodal ist, unausgeglichene Sensormodelle oder das sogenannte Sample Impoverishment. Die Kompensation, im besten Fall sogar die Eliminierung durch fortgeschrittene Methoden der Sensorfusion ist der Hauptbeitrag dieser Arbeit. Der wichtigste Ansatz in diesem Zusammenhang ist unsere Anpassung eines *Interacting Multiple Modal Particle Filters* (IMMPF) an die Anforderungen einer Indoor-Lokalisierung. Damit schaffen wir einen völlig neuen Ansatz zur Formulierung eines ILS. Mit Hilfe von Qualitätsmetriken ist ein dynamischer Wechsel zwischen parallel laufenden, beliebig formulierten Partikelfiltern möglich. Darüber hinaus schlagen wir mehrere Ansätze aus dem Bereich der Partikelverteilungsoptimierung explizit zur Vermeidung des Sample Impoverishment Problems vor. Insbesondere der ebenfalls auf dem IMMPF-Prinzip basierende *Support Filter Approach* (SFA) führt auch unter schwierigsten Bedingungen zu guten Positionsschätzungen, wie umfangreiche Experimente zeigen.

Contents

Chapter 1

Introduction

In recent years, sensors have become a constant companion in our daily lives. They are on our wrists, in our pockets, at our workstations, and sometimes even drive around our homes vacuuming. From bicycles to toothbrushes, everything is made intelligent by a multitude of sensors, large and small, that monitor different entities such as temperature, speed, or the relative change of the magnetic field. Of particular interest to recent computer science research are multisensor systems that combine different sensing modalities to solve challenging problems. A prominent example of a multisensor system is an autonomous car. Cameras around the vehicle allow the detection of objects such as pedestrians or road signs using computer vision and machine learning. Various proximity sensors measure the distance to other road users, while an *inertial measurement unit* (IMU) keeps track of the car's own orientation and speed. A *light detection and ranging* (LiDAR) scanner performs *simultaneous localization and mapping* (SLAM), providing an absolute position for the car. This list can probably be extended indefinitely, but according to the current state of the art, none of these technologies alone can guide a car autonomously through traffic [Yur20]. Thus, only a combination of different sensor modalities is able to provide true autonomous behavior. Such a combination of very different sensor types is known as sensor fusion and is the basis for most multisensor systems.

1.1 Towards Sensor Fusion Systems

Computational sensor fusion systems were first used in the military and aerospace industries for guidance, navigation, and control of vehicles, particularly aircraft and spacecraft. One of the first and probably most famous systems was implemented in the navigation computer of the Apollo program, which sent the first humans to the moon by continuously tracking and adjusting the position and orientation of the space shuttle [GA10]. There are classic books

on sensor fusion such as “Mathematical Techniques in Multisensor Data Fusion” by [HM04], “Multisensor Data Fusion” by [WL90], or “Data Fusion in Robotics & Machine Intelligence” by [Meu92], which propose different definitions of the term sensor fusion. A precise definition in relation to this work is given by [HM04]. According to him, sensor fusion “seeks to combine data from multiple sensors to perform inferences that may not be possible from a single sensor alone”. The term sensor has a broad meaning, ranging from classical raw data (e.g., acceleration in $\mathrm{m\,s^{-2}}$) to high-level or context-aware results (e.g., object recognition). As seen in the autonomous car example above, combining such a high versatility of sensors is a necessary requirement for many modern applications.

It should be noted that there is some confusion in the terminology for fusion systems in the literature. Besides the term sensor fusion, there are data fusion, information fusion, and multisensor data fusion [Elm02]. In this thesis we follow the definitions given by [Das01], where information fusion is used as an umbrella term for fusion of any kind of data. This includes, for example, applications related to data mining and database integration. Information fusion is then defined by [Das01] as “Information fusion encompasses the theory, techniques, and tools designed and used to exploit the synergy in the information acquired from multiple sources (sensors, databases, information gathered by humans, etc.) in such a way that the resulting decision or action is in some sense qualitatively or quantitatively (in terms of accuracy, robustness, etc.) better than would be possible if these sources were used individually without such synergy exploitation.” It follows that the other terms are subsets of information fusion. However, in this thesis only the problem of sensor fusion is considered, and thus all other terms with ambiguous meaning are ignored.

When compared to a single sensor system, a fusion of multiple units offers a number of significant advantages. Fusion of multiple independent measurements of the same property allows for improved resolution and accuracy of the resulting value, e.g., using a variety of temperature sensors instead of a single one. By deploying such sensors at different locations and allowing them to collect data unsynchronized, a sensor fusion system is then able to extend the spatial and temporal coverage. Even a failure of one sensor element does not necessarily result in a loss of perception, allowing the system to continue to provide information. Another benefit is increased confidence, since the measurement of one sensor is confirmed by another sensor covering the same domain. As seen in the autonomous car example, sensor fusion systems are able to cover more high-level and context-aware problems by combining very different entities. As a result, the system becomes less vulnerable to interference due to the increased dimensionality of the measurement space (e.g., measuring the desired quantity with optical sensors and ultrasonic sensors) [Elm02]. From a more practical perspective, sensor fusion is able to reduce

system complexity. Since most state-of-the-art methods have a unified representation of the sensor output as a probability, it is easy to add new sensors in a modular way.

However, in contrast to these - and likely many more - benefits, a sensor fusion system faces several challenges. As mentioned above, sensor fusion increases the susceptibility to interference. However, this is only true if all sensors are working as expected and do not produce unusual measurements due to some damage. In such a case, the accuracy could be reduced to the point where a single well-functioning sensor would provide better overall results. Another challenge comes from the large differences between some sensors (e.g. Wi-Fi and barometer). Here, the respective model parameters (e.g. sensor noise) must be optimally adjusted to avoid unwanted interactions between the sensors. This is not easy for complex scenarios, especially in a time-sequential domain without future knowledge [TBF05]. Once again, a very practical aspect must be emphasized. Especially in mobile devices, battery life plays a central role and is increased by the use of multiple sensors.

In the literature, there are several categorizations for sensor fusion systems. However, most authors distinguish three basic levels of fusion: data-level fusion, feature-level fusion, and decision-level fusion [Elm02; Gra17; Yan14]. Data-level fusion combines multiple heterogeneous sources of raw sensor measurements (e.g., 3-axis accelerometer) to produce new data that is more accurate and informative than the original source. As the name suggests, feature-level fusion combines multiple features (e.g., edges, peaks, or color) from multiple sensors to create a new high-dimensional feature vector that is the input for classification and pattern recognition steps. For most scenarios, it is useful to apply feature selection algorithms (e.g., *principal component analysis* (PCA)) to the new vector to obtain a most significant feature vector [Sze22]. The last and most important category of sensor fusion in the context of this thesis is decision-level fusion. The output is a high-level decision (e.g., emergency braking of an autonomous car) obtained from information that has already been abstracted to a certain level by preliminary sensor data or feature-level processing (e.g., obstacle detection and radar). More generally, decision-level fusion is the process of selecting (or generating) a hypothesis from the set of hypotheses generated by individual (local and often weaker) decisions of multiple sensors [Gra17]. Of course, all these different categorizations of sensor fusion are not mutually exclusive and allow for hybrid architectures depending on the scenario and application. However, in this work we mainly consider decision-level fusion, except for some pattern recognition tasks.

As described above, sensor fusion combines a set of sensors to make inferences that may not be possible with a single sensor alone. In a mathematical context, an inference is represented as an environment, process, or quantity whose true value, situation, or state is unknown. The knowledge is then obtained indirectly from the associated (noisy) sensors. It is obvious that such a process suffers from a multitude of uncertainties based on the imperfection and incom-

pleteness of the sensors and the methods used. Therefore, the main task within a sensor fusion system is the development of appropriate models that describe the uncertainty of the system. By providing an explicit measure of uncertainty, the information provided by the sensors can be fused in an efficient and predictable manner [DWH16].

There are many methods to describe this uncertainty, with probabilistic methods being by far the most widely used in the literature [DJ09; DWH16; Gra17; HM04; Sä13; TBF05; Yan14]. As in everyday life, the word probability describes a conditional measure of uncertainty associated with the occurrence of a particular event, given the available information and accepted assumptions. This naturally leads to ideas of sensor fusion and how to use sensor information to make inferences about such an event. The probabilistic method that captures this behavior is Bayesian inference. It uses the principle of Bayesian statistics to update the probability of a hypothesis as more evidence or information becomes available. Because new sensor data becomes available over time, sensor fusion problems are often formulated in a time-sequential dynamical system, also known as a state-space model. To avoid theoretical limitations in such systems, a Markov process with unobserved states is assumed (*Hidden Markov Model* (HMM)), leading to the broad class of recursive Bayesian filtering. The so-called Bayes filter is a general probabilistic approach that estimates an unknown probability density function that describes the uncertainties of the system. The density is updated recursively over time using two probabilistic models: The transition model represents the dynamics of the system and its uncertainties as a Markov sequence, while the evaluation model estimates a probability of how the current sensor measurement, including sensor noise, depends on the unknown state [DJ09].

Given a probabilistic interpretation for a certain sensor fusion problem, the goal is to provide successive estimates of the current most likely hypothesis, which is often represented by a mode of the probability density. This procedure is also called recursive state estimation. In this context, a state (hypothesis) describes the unknown measure that represents the inference obtained by combining different sensors, e.g. the position of a car on the road by a *Global Navigation Satellite Systems* (GNSS), radar and IMU. However, state estimation is only one, and in many cases the simplest, way to interpret a probability density. What is often neglected in common sensor fusion applications is that one can make inferences by interpreting more of the density than just its modes.

Although the Bayes filter provides an optimal solution for recursive density estimation, it is not possible to compute it in closed form for most (realistic) problems. Especially for high-dimensional problems, it is very difficult to find analytic solutions for the densities. They exist only in rare cases [DJ09]. Therefore, a large group of different (numerical) approximation methods has been developed in the last decades, which allow probabilistic sensor fusion on digital computers. One of the oldest and probably the most studied solution for implementing the

Bayes filter is the *Kalman filter* [Kal60]. It assumes linear evaluation and transition functions while representing the probability density and sensor noise as a multivariate Gaussian distribution. As a result, the Kalman filter belongs to the class of Gaussian filters and results in a linear Gaussian system. Over the last 50 years, several extensions to the standard Kalman filter have been developed to overcome the linearity assumption. For example, the *extended Kalman filter* uses a first-order Taylor approximation to provide a linearization of the transition and evaluation functions [TBF05].

An alternative for applications that require a non-Gaussian representation are non-parametric filters. Compared to Gaussian filters, they allow for the approximation of any given probability density function. Such filters are often used in complex scenarios requiring multi-modal densities or complex dynamics, such as multi-target tracking or pedestrian localization [Dei05; Ebn15]. A well-established and widely used class of non-parametric filters is *Sequential Monte Carlo* (SMC), also known as *particle filter* in the Bayesian context. As the name suggests, the filter uses the Monte Carlo method to obtain numerical results. A set of weighted random samples (particles) is used to solve any problem with a probabilistic interpretation. Each of these particles then represents a possible hypothesis (unknown state) of the problem statement. New particles are drawn according to an importance distribution, often represented by the state transition that models the dynamics of the system. These particles are then weighted by the state evaluation given different sensor measurements. A resampling step is used to avoid that only a small number of particles have a significant weight [DJ09]. There are countless extensions and improvements to SMC, making it the most dominant method for sensor fusion in recent years [DJ09; Sä13; TBF05] as well as within this thesis. The easy adaptation to a wide range of time-sequential problems as well as the high degree of freedom have led to this success. As mentioned above, the state space can be non-linear and non-Gaussian, while the prior and sensor noise distributions can take any form. It is very easy to generate new samples and thus provide state estimation. Finally, SMC is perfect for parallelization on modern computers or graphics cards.

Besides the broad class of probabilistic methods, some alternative techniques such as fuzzy set theory and evidential reasoning have been proposed to describe the uncertainty of a system. They deal with perceived limitations of probabilistic methods, such as complexity, inconsistency, precision of models, and uncertainty about uncertainty [DWH16]. Sometimes, sensor fusion systems consist of a large number of different sensors, which then requires specifying a large number of probabilities to be able to use methods like SMC, leading to high dimensionality and thus complexity. Unlike Bayesian inference, evidential reasoning allows each source to contribute information at different levels of detail. For example, one sensor may provide information to distinguish individual entities, while other sensors may provide information to

distinguish classes of entities. In addition, methods of evidential reasoning are able to deal with incomplete information or qualitative assessments of uncertainty, and provide a method for capturing ignorance or an inability to distinguish between alternatives [Kha13a].

Of course, the choice of a suitable method depends strongly on the problem and the inferences that need to be derived from it. However, there is no perfect solution, but rather a selection of suitable methods and combinations of these methods. Much depends on the required quality measures such as accuracy, resolution, flexibility and robustness. For example, SMC is the de facto standard for target tracking due to its high accuracy and robustness to conflicting sensor readings, while evidential reasoning offers a more flexible formulation and thus a much richer representation of belief, leading to successes especially in the area of automated reasoning [BHM97; Kha13a].

1.2 Pedestrian Localization and Navigation

One of the most active research areas in sensor fusion is pedestrian localization and navigation. Both problems are part of the larger class of target tracking. Pedestrian (self-)localization is the problem of estimating the position of a person walking in a known or unknown environment, while navigation combines the former with path finding methods. We divide environments into two categories: outdoor and indoor. Outdoors, GNSS such as Global Positioning System (GPS) or Galileo are mainly used as a source of information. Today, they have an accuracy in the public sector of up to four meters, which is sufficient for most outdoor applications, such as finding a nearby coffee shop or following hiking trails [Oeh09]. Using specialized technologies like *Real-time kinematic positioning* (RTK) an accuracy of a few centimeters can be achieved in free field. This reduces the last really open problems in the outdoor sector with GPS to localization in difficult scenarios such as urban canyons, mountains or dense forests [Cri17; TV09; ZG21]. Indoors, however, the situation is very different. The high-frequency GPS signal is severely attenuated by solid objects such as walls or ceilings that block the line of sight to the satellite. This leads not only to a significant loss of signal strength, but also to multiple reflections off surfaces causing multipath propagation, which ultimately leads to uncontrollable errors [DH10]. Especially in multi-storey buildings made of brick or concrete, GPS gives an accuracy level of tens of meters and it sometimes takes several minutes to get the first position fix [Kjæ10]. As a result, GNSS does not play a dominant role in indoor localization and is even ignored in most state-of-the-art systems due to its unfavorable characteristics.

As mentioned above, determining a position in an indoor environment is a very challenging task due to a number of significant limitations. To date, the architecture of most buildings has not been designed with localization in mind. Sophisticated or historic facilities often consist

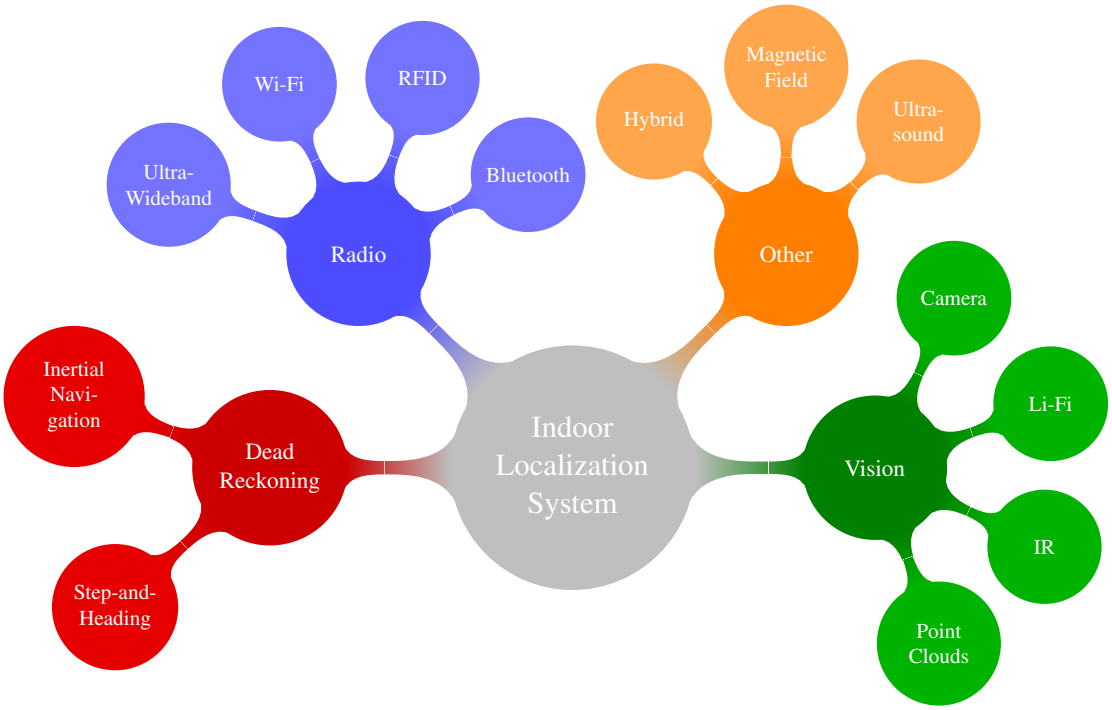


Figure 1.1: Overview of the most common indoor localization technologies. A more comprehensive and detailed version can be found in Appendix A.

of irregularly shaped spatial structures, while modern multi-story office buildings are made of highly attenuating materials such as reinforced concrete and metalized windows, posing problems for localization systems using wireless technologies. Compared to outdoor navigation, much higher accuracy must be achieved, especially in environments with many small separated areas, such as shopping malls or trade fairs. Another limitation is that many technologies make inferences based on the pedestrian's movement. Unlike the well-known behavior of robots, human motion is difficult to predict and rarely follows linear conditions, especially when considering environmental constraints such as not walking through walls. Until now, no localization standard has been able to establish itself in the market. On the contrary, new approaches, methods and technologies are constantly being introduced to solve the problem. There is no "one and only technology" like GNSS for outdoor applications. This leads to a variety of different sensors and technologies to choose from, making the development of such systems even more difficult.

1.2.1 State of the Art

In recent years, many *indoor localization systems* (ILS) have been introduced to meet the above requirements. Figure 1.1 provides a categorized overview of the most common solutions. Of

course, ILS can not only be used for pedestrian localization, but also cover a broader spectrum such as robotics or object tracking. Especially in the field of robotics, many technologies have found their way into our everyday life, such as industrial warehouses, vacuum or service robots. However, this thesis focuses on the more difficult topic of pedestrian localization and navigation, as introduced above. For the sake of clarity, we will use the term ILS as a synonym for pedestrian localization in indoor environments.

One of the first ILS was presented in the pioneering work of [BP00] using triangulation based on Wi-Fi radio-frequencies. Since then, radio-based approaches have become one of the most researched and promising technologies for localization [Guo20; Liu07]. They are able to provide an absolute position estimate relative to some wireless base station (e.g., Wi-Fi access points or RFID readers). Of course, the person to be tracked must also carry an appropriate mobile device for communication. Many different wireless technologies have been implemented in ILS. Because of their commercial success as anti-theft and barcode replacements, RFID chips seem to be a good fit. There are two types of RFID tags: active tags, which are powered by a local power source such as batteries, and passive tags, which are powered and activated by waves from nearby RFID readers. Passive tags are mostly used for so-called proximity methods because of their short range of 1 to $2\,\mathrm{m}$. Here, the pedestrian's position is determined by the nearest RFID reader that detects the tag being carried. However, this forces the user to constantly interact with the environment, limiting the use of this technology to certain applications. For example, [Kie] deployed arrays of passive RFID tags under a carpet to provide route guidance for visually impaired and blind people using a foot-mounted reader. However, it is clear that the accuracy of such a solution is proportional to the number of tags deployed. Active transceivers, on the other hand, are heavier and more expensive, but provide long detection ranges of $20\,\mathrm{m}$ or more. This allows them to use more advanced location techniques such as multilateration or fingerprinting based on *received signal strength indicators* (RSSI). Well-known ILS systems that use active RFID with RSSI are SpotOn [HWB00] and LANDMARC [Ni03].

Fingerprinting is one of the oldest, but still one of the most popular solutions for indoor localization. This is confirmed by the continuous development over the last decade [EK10; Sol20; Yiu17]. During a one-time offline phase, a large number of RSSI reference measurements are taken throughout the building to build a radio map. During the online phase, the pedestrian's location is then inferred by comparing the radio map with live measurements. There are many different ways to create a radio map and obtain a position estimate from it. They range from a simple deterministic representation of the reference measurements, which provides the best estimate using the nearest neighbor of the current received signal strength, to a probabilistic representation of the radio map, which chooses the most likely position based on Bayesian in-

ference. For example, in the 2014 Microsoft Indoor Localization Competition, an ILS using neural network-based Wi-Fi fingerprinting is reported to have an average location error of only $1.56\,\mathrm{m}$, taking second place [LL17b]. However, despite the very good accuracy achieved, fingerprinting approaches suffer from huge setup and maintenance times. The use of robots instead of human workers may therefore be a viable option [Hat21; Yeh09].

Multilateration, on the other hand, estimates the position of a mobile object by measuring its distances from multiple base stations (e.g., RFID readers or Wi-Fi access points), three or more at most. Thus, only the positions of the base stations are required, which makes the setup and maintenance effort manageable. The distance can be derived using time-based measures such as *time of arrival* (TOA), *time difference of arrival* (TDOA) or *round-trip time* (RTT) [Liu07]. As the names suggest, the calculations are based on different travel times from the transmitter to the receiver as well as the speed of the radio signal. Under optimal *line of sight* (LOS) conditions, such techniques can achieve very high accuracies. Obviously, such optimal conditions rarely exist in indoor environments. In reality, walls and other obstacles cause non-line of sight (NLOS), which then leads to error-causing effects such as path loss and multipath propagation. To compensate for this, an error model is integrated into the distance calculation [Bul22; Rac11]. In addition, time-based solutions usually require accurate time synchronization between base stations, which demands specialized hardware. A promising technology based on such hardware is *ultra-wideband* (UWB). Due to its very high accuracy (up to $0.2\,\mathrm{m}$), it has become the focus of attention in recent years, dominating the 2016 Microsoft indoor localization competition [LL17a]. UWB uses a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum, depending on the policies and regulations enforced by different countries. For Europe and USA the bandwidth is limited to 3.1 - $10.6\,\mathrm{GHz}$ and can only be used indoors [Ala16]. Due to the combination of very high time resolution and very short pulses, typically less than a nanosecond, UWB is able to accurately and robustly measure the *time of flight* (ToF) of signals. These characteristics make it particularly robust to multipath effects such as reflection and scattering. In practice, there may be potential interference with existing systems, for example WiMAX, Wi-Fi at $5\,\mathrm{GHz}$ and US digital TV operate within the UWB spectrum. In addition, indoor environments with many reinforced concrete walls can cause extreme NLOS conditions, which can result in signal attenuation and therefore propagation delay up to hundreds of nanoseconds [BCX19]. Nevertheless, UWB is one of the most promising indoor technologies of the future, not necessarily for pedestrian navigation in leisure areas such as shopping malls, but especially in the industrial field of warehouse management and vehicle monitoring [Fet22].

In contrast, the aforementioned RSSI values are provided by most commercial devices out of the box, making it the most available technology for estimating distance in multilateration.

RSSI-based methods estimate distance by calculating the attenuation of the transmitted signal strength, which increases robustness in NLOS conditions [Liu07]. This is done using radio propagation models. Different propagation models have been developed to account for different attenuation conditions [SR92]. For example, the wall-attenuation-factor model introduces an additional parameter to the well-known log-distance model [Fri71], which takes into account obstacles (e.g. walls, ceilings or floors) between the base station and the location in question. Of course, this requires an intersection test of each obstacle with the line-of-sight, which is costly for larger buildings. Various solutions such as using an optimization scheme for the attenuation parameters or pre-computing a signal ray-tracing are proposed in the literature [Ebn17; Tay09].

The most interesting wireless technologies today are Wi-Fi and *Bluetooth Low Energy* (BLE), since most of their commercially available chipsets provide RSSI readings. Today, most buildings have a Wi-Fi infrastructure that is supported by a wide range of mobile devices, especially smartphones. Because BLE devices are inexpensive and can even run on batteries, the necessary infrastructure can be set up quickly and cost-effectively. This promises a huge market potential for such solutions, affecting the development of many ILS [Ebn15; Fet23a; Kna17; Nur13; Pip19; SP20]. Wi-Fi and BLE frequencies are very sensitive to reinforced concrete walls and reflecting windows. To achieve optimal results, advanced radio propagation models are required, which then require an accurate description of the building's architecture and construction materials. In most cases, however, it is very difficult to create such a description, leaving many unknown factors. This also leads to multipath effects such as shadowing, reflection, refraction and absorption [Mau12]. Nevertheless, even when using weak propagation models, RSSI multilateration methods based on Wi-Fi or BLE are able to provide accuracies between 2 to $5\,\mathrm{m}$, while keeping the setup and maintenance effort low compared to other solutions such as fingerprinting [Fet18].

In 2016, a first attempt was made to bring together both worlds, the accuracy of time-based methods and the widespread adoption of Wi-Fi devices. The IEEE 802.11-2016 specification, also known as IEEE802.11mc, introduced the Wi-Fi *Fine Time Measurement* (FTM) protocol. It determines a distance by measuring the signal's *round-trip time* (RTT) between an initiating device (such as a smartphone) and a responding station (such as an access point). This eliminates the need for synchronized clocks between nodes. Compared to UWB, Wi-Fi FTM is intended to operate in the common Wi-Fi frequency bands $2.4\,\mathrm{GHz}$ and $5.0\,\mathrm{GHz}$. To sample a signal for time measurement, the sampling rate is critical. This is directly related to the bandwidth of the Wi-Fi channel, i.e. the larger the bandwidth, the more accurate the FTM distance measurement. For example, at $5.0\,\mathrm{GHz}$ with large bandwidths of $80\,\mathrm{MHz}$, sub-meter accuracy could be reported [Hor20b]. In contrast, the more common $2.4\,\mathrm{GHz}$ with a small bandwidth of $20\,\mathrm{MHz}$ is more susceptible to signal propagation errors, with results ranging from 1 to $5\,\mathrm{m}$

[Bul20b; Fet22]. As noted by [Bul20b] and [Hor20b], many manufacturers have begun to incorporate FTM into their devices, although implementation of FTM is optional. The potential for FTM is huge, since all that is required is a refresh of a building's Wi-Fi access points, which is necessary after some time anyway. In addition, many flagship smartphones running Android are already capable of the protocol [Hor22].

Other technologies that provide absolute position information are vision-based solutions. Especially in robotics, such methods are used because they promise very high accuracies (mm to cm) and allow procedures like *simultaneous localization and mapping* (SLAM) to be carried out very thoroughly [BDW06; DWB06; Li20]. Here, the optical sensor (e.g., camera or LIDAR) is mounted on the robot, resulting in self-localization based on its environment. As mobile devices have become smaller and more powerful, such systems have become interesting for pedestrian localization and navigation. Of course, it doesn't make much sense to strap a LIDAR to someone's head, although this has been done in the past [HB22]. Instead, most approaches use standard CMOS sensors or depth cameras such as stereo vision or structured light. Regardless of the sensor, feature-based methods are typically used to estimate the pedestrian's position by comparing a live image to a set of pre-recorded data. While depth cameras try to find features in a 3D point cloud of the building, ordinary CMOS images are compared to a sequence of images taken from a video or database and assigned a fixed position. There are countless different techniques for feature detection, ranging from simple template matching to advanced methods like *scale-invariant feature transform* (SIFT) or thoroughly trained neural networks [Sze22]. Once the correct point or image sequence has been found, the distance can be derived either by transforming the relative depth information to an absolute position in the case of depth sensors, or by estimating the view geometry using *angle of arrival* (AOA) multilateration for CMOS images.

For example, [WKM11] uses images taken with a standard smartphone camera for both positioning and database creation. It extracts features using *speed up robust features* (SURF) and calculates the position by applying a geometric position correction scheme to the most appropriate image from the database and the current live image. Using high resolution images with an extensive feature search results in an average accuracy of $0.68\,\mathrm{m}$, while low resolution images still perform with $2.85\,\mathrm{m}$. A major disadvantage of such solutions is the large amount of data generated. For large buildings, this can only be handled by external servers, which requires a permanent high-speed wireless network connection. Finally, constant use of the camera consumes a lot of power, which makes it questionable.

In contrast to a moving camera, static vision sensors (e.g. surveillance cameras) analyze the position of a person using object detection methods. Detecting and tracking people in video or images is a classic problem in computer vision. A large number of different solutions exist, with

convolutional neural networks having achieved considerable results in recent years [Meh17; Meh20]. Compared to surveillance cameras, depth sensors can provide more accurate results, but have a much shorter range and a higher price [Fet14]. In addition, many public buildings are already equipped with surveillance technology. However, there is still one major open problem: the unambiguous identification of the localized person. This would require very complex and questionable systems [Bow04].

Another vision-based technology is *light fidelity* (Li-Fi). It is more of a wireless communication technology like Wi-Fi, but it communicates in the visible spectrum. Li-Fi uses *visible light communication* (VLC) by switching LED lamps on and off at very high speed, reaching data rates up to $200\,\mathrm{Gbit/s}$. Of course, it requires LOS conditions, but no direct LOS is needed, and even light reflected from walls can reach up to $70\,\mathrm{Mbit/s}$ [DH15]. These properties, as well as the basic presence of lighting in buildings, make this technology very interesting for ILS. Although this is a very promising approach, the technology is still a niche player in indoor localization research. According to Google Scholar, [Hua16]'s work is among the most relevant papers in the field, with 32 citations. They add Li-Fi to support a multilateration approach using Wi-Fi RSSI with a propagation model. If a Li-Fi signal is measured, they identify the transmitter LED lamp and then use its distance to nearby access points to re-calibrate the propagation model. They report an improvement in accuracy from $3\,\mathrm{m}$ to $0.62\,\mathrm{m}$, but without revealing many details about the experiments, so this should be considered a simplified lab test. Other more recent approaches can be found in the literature, but most of the sources do not meet the scientific quality of the author of this thesis.

In addition to Li-Fi for localization, other VLC techniques such as *on-off keying* (OOK), IEEE 802.15.7 or other home-grown solutions are presented in the literature. For example, [Kuo14a] uses off-the-shelf LEDs that transmit their position and identifier with OOK. The optical pulses are then captured by a single front-facing smartphone camera using the rolling shutter effect of the CMOS sensor. An image processing pipeline checks for the presence of certain frequencies and decodes the data of the LED transmissions if they are present. Given the image coordinates of the LEDs and their fixed position, an AOA algorithm determines the position and orientation of the phone in the building. The accuracy of the system depends on good LOS conditions, the camera resolution and the number of detected LEDs. Nevertheless, in a small test environment [Kuo14b] achieves decimeter position error and 3° orientation error. Of course, determining the position of all LED lights in advance can be very time-consuming, especially for large buildings.

So far, all of the ILS presented require the mobile device or user to somehow interact with hardware installed in the environment. However, not all buildings are able to provide such solutions everywhere, or there are repeated radio gaps in certain areas. Independence from

this can be achieved by relative positioning methods, also known as pedestrian dead reckoning (PDR). In PDR, the user's movement is tracked by accumulating the change in position from a given starting point. This is done by estimating a step length (or walking speed) and a ground trajectory (or walking direction) for incoming sensor measurements. Prior activity detection can help classify the user's movements into appropriate categories such as walking, driving, or climbing stairs. This information is obtained using an IMU, which measures a body's specific force with an accelerometer and angular rate with a gyroscope for each of the three axes: pitch, roll, and yaw. It is now common to equip smartphones with an IMU to check the position of the device to switch between screen modes or to perform activity recognition tasks such as detecting fitness exercises. Sometimes a magnetometer is integrated to measure the surrounding magnetic field or even a barometer to measure atmospheric pressure.

Step detection and length estimation are mostly based on the assumption that pedestrian motion is cyclic. Thus, various methods such as thresholding and peak detection, autocorrelation, and spectral analysis are used to search for repeating patterns within the measured acceleration. For example, [KGD14] presents a simple yet efficient method that uses a probabilistic representation for steps. After applying a low-pass filter to the vertical acceleration, they search for local minima and maxima between adjacent measurements and calculate the delta between them. A probability for each delta is then estimated using a gamma distribution. This allows for more robust results than a simple threshold-based solution. A very good comparison between different methods can be found in [BH13] and [Wan22].

Note that many implementations claim to use only vertical acceleration and do not compensate for changes in sensor pose during walking, i.e., the smartphone screen always facing up [DP17]. However, this condition cannot be met in practice, e.g. when the smartphone is temporarily placed in a pocket or held to the ear to make a phone call. There are several approaches to satisfy such conditions. Some extend the above to allow for arbitrary poses and carrying locations [PL15], others constantly track the IMU orientation to determine the vertical acceleration vector [SS10], and still others perform activity recognition and adapt different algorithms for different motions [Mar18].

The direction the pedestrian is walking can be determined using a gyroscope. Most PDR systems compute the IMU's orientation with respect to an Earth-fixed global system, also called the *east-north-up* (ENU) frame. This can be described by a transformation matrix that rotates the gyroscope from the body frame to the global frame. After the rotation, integration over the gyro's z-axis for a predefined time interval provides the user's heading change (yaw) [Ebn15]. In combination with one of the above mentioned step detection methods, a classical PDR for ILS is obtained. However, determining the transformation matrix in the first place is a difficult task that requires frequent updates from other auxiliary sources, such as magnetometers and

accelerometers [DP17; Mad10]. Due to magnetic field distortions in indoor environments and slow updates, standard magnetometers, such as those found in modern smartphones, are not reliable. For this reason, approaches that use the accelerometer instead are often preferred. Here it is assumed that the acceleration during walking is cyclic, and thus the average acceleration over several cycles must be close to zero. This allows the direction of gravity to be measured and used to construct the transformation matrix. Especially with cheap and therefore inaccurate IMUs, the matrix has to be updated at very short intervals to maintain good results [DP17].

PDR systems for pedestrian localization have an obvious disadvantage. Even small measurement errors add up during integration, resulting in potentially cubic growth over time (drift) and thus highly erroneous position estimates. While military, aerospace, or naval applications use high-precision sensors that keep the error small and allow tracking for many hours, most consumer devices use cheap microelectromechanical system (MEMS) based IMUs. These MEMS IMUs usually only allow valid positions to be estimated for a short time (single digit minutes) before the drift eventually dominates [Wan22]. There are many ways to limit the drift. One of the most commonly proposed approaches for static (e.g. foot-mounted) IMUs is *zero velocity updates* (ZUPT). A ZUPT algorithm detects when the system is in a stationary phase, i.e. when the system has a constant position and attitude, and counteracts the drift by performing either a hard or soft update of the IMU. As the name implies, a hard update resets the position, velocity, and yaw to zero and re-initializes them using the gravity-based transformation matrix. Soft updates, on the other hand, use a model that uses knowledge of how the errors have evolved over time to obtain an estimate of the accumulated errors since the last update. The accumulated error estimate is then used to correct and calibrate the PDR system [WS21]. The authors of [SNH10] evaluated several approaches and the best ones achieved a position error between $0.14\,\%$ and $0.20\,\%$ of the distance traveled.

As the above shows, even these approaches cannot solve the drift problem, they can only delay it. A more promising approach is to use additional (absolute) position information, such as that provided by Wi-Fi RSSI or depth cameras. These technologies are able to correct for drift with each incoming measurement, eliminating the need for an initial position. At the same time, PDR allows human motion to be integrated into the system. Such ILS approaches are known as hybrid systems. They are probably the largest and most promising group, because they combine the solutions discussed above using high-level sensor fusion methods. In particular, extended Kalman filters or particle filters are used to estimate the current position using several of the individual technologies presented earlier. Given the conclusions in section 1.1, such an approach seems very reasonable.

Of course, solutions such as UWB and LIDAR already provide very high accuracy. However, they are either based on specialized non-commercial hardware or are not suitable for nor-

mal pedestrians. Based on the technologies discussed so far, two basic types of hardware can be distinguished for ILS: Commercial hardware that is openly available to everyone, i.e. mobile devices such as smartphones and fitness trackers, but also stationary devices such as cameras and Wi-Fi access points, and specialized hardware that is not established on the market or in the average household, often forcing pedestrians to carry an extra device just for localization. ILS based on commercial hardware thus aim to solve the problem by using established technology that is already carried by pedestrians, and thus follow an approach that is practicable today, while specialized hardware takes a very technical and solution-oriented view of the problem.

Most applications aiming for commercial success or immediate usability consider smartphones and their integrated technologies as the only source of information. As already discussed, smartphone sensors are based on a low-cost and battery-saving design, which does not allow for high accuracy. Therefore, hybrid systems are of particular interest. Thanks to the use of sensor fusion methods, they benefit from advantages such as compensation for a sensor failure or the higher reliability due to different technologies that have the same problem domain. Since many manufacturers install different types and models of sensors, there is no alternative but to design a system that is as modular as possible in order to achieve a comprehensive solution. Probabilistic fusion methods are particularly well suited for this purpose, as they allow individual sensor models to be switched on or off.

An obvious choice for a hybrid system is to integrate the magnetometer as an absolute information source. By using it as a compass, it works well with PDR approaches, compensates for the aforementioned drift, and does not require any additional hardware inside the building. The compass provides the absolute azimuth based on the Earth's geomagnetic field. This can be used to determine the direction of the pedestrian and update the parameters of the gyroscope. For example, the authors of [RWRC14] present a very comprehensive PDR system that even includes different movement patterns of the user. Based on the compass measurements, they form inference vectors and rotate them until they all point to the same object causing the magnetic disturbance, now the magnetic north can be determined based on the original measurement. In addition, after an initial phase of a few steps, the (relative) angle measurements of the gyroscope can be used to compensate for larger jumps of the compass. In this way, the redundancy of the information significantly improves reliability.

As discussed earlier, magnetometers inside buildings are subject to perturbations and deflections of the magnetic field, mostly caused by metals in the building material. One approach that takes advantage of this to obtain absolute location information is magnetic matching. Similar to radio fingerprinting, a large number of reference measurements are taken throughout the building to create a magnetic map. Assuming that the magnetic field inside a building is approximately static, each reference has its own unique signature of magnetic flux density. The

current location is determined by comparing the current flux density with the flux density values stored in the map [OAM22].

A hybrid system combining magnetic matching and PDR is presented by [Sol16] using a particle filter (SMC) for fusion and state estimation. Here the particle filter has very practical advantages. The state transition, i.e. what models the system dynamics and thus the movement of the pedestrian, can be represented by PDR, while magnetic matching weights the particles in the state evaluation. Another major advantage of this approach is the easy integration of other sensors into the evaluation process, assuming statistical independence. For example, [Guo16] also added Wi-Fi fingerprinting to the above system, as [Sol16] suggested in their conclusion. This modularity, as well as the properties already discussed in section 1.1, such as non-linear and non-Gaussian state space representation, make particle filters the de facto standard for target tracking and thus pedestrian localization and navigation.

The success of systems using particle filters can be confirmed by looking at the results of past competitions for smartphone-based solutions, organized by the EvAAL group and hosted at the International Conference on Indoor Positioning and Indoor Navigation in different places around the world [Eva]. In contrast to other competitions, the organizers attach great importance to a realistic environment, where only commercial devices can be used in unprepared multi-storey buildings. Impartial test persons then walk several hundred meters long routes from which the accuracy of the system can be determined [Pot17a]. The best result so far in a 3D scenario was achieved with our system based on a particle filter [Ebn15; Fet18]. Similar approaches have dominated the field in recent years. Some other notable competitors are the ILS of [Kna17; Mos13] and [PLP21].

All of these systems share the lowest common denominator of embedded sensors. They all use some combination of Wi-Fi or BLE, PDR, and knowledge of the building, such as the position of walls and doors. In hybrid systems, it is common to consider environmental constraints within the state transition. This allows to model the potential movement of a person inside the building in a more realistic way, e.g. to prevent walking through walls. Thus, one is interested in only allowing positions that are actually feasible. The simplest transition model simply rejects all movements where the LOS between the current position and the potential destination is blocked by an obstacle. However, the computational complexity of such intersection tests is quite high, as it depends on the number of particles used as well as the architecture of the buildings [ARC12a]. A potential method to avoid these (online) intersection tests altogether is to reduce the possible movements to a graph. This of course leads to a more discrete approximation of the transition, which only allows movement over edges from one vertex to another of the graph. The graph can be generated from the floorplan of the building in a one-time offline phase, ignoring all edges that intersect a wall. There are quite different types of graphs, ranging

from topological descriptions, such as a navigation graph [YW15], having a vertex for each room or point-of-interest, to regular or irregular tessellated cells, describing the indoor space with a set of polygons [Ebn21].

1.2.2 Open Problems in Hybrid Systems

Looking again at the last mentioned ILS based on particle filtering, it can be seen that the sensors used are often very similar, but the systems differ greatly in the way the sensor models are designed and how they are fused within the particle filter. For example, [Fet18] and [Kna17] combine Wi-Fi and IMU as the main source of knowledge. While the former uses RSSI multilateration in combination with PDR on an equidistant grid, the latter uses fingerprinting and motion based on truly continuous PDR, which is required for intersection testing. Furthermore, [Kna17] uses discrete ground changes based on the RSSI readings and [Fet18] allows for continuous ones using the smartphone's barometer. However, the basic structure and methodology of SMC remains the same and is being adopted by many other ILS due to recent successes. As recent work has shown, particle filters for target tracking suffer from a number of profound problems, implying that most of these systems are also affected [Bul18; Ebn22; Fet16; Fet17; Fet18; Fet23a; Guo20; Li14].

One problem that has a dramatic impact, and can even cause state estimation to lose track or stall altogether, is sample impoverishment. Consider a particle filter as described at the end of section 1.1. After transition and evaluation, a resampling step is often used to prevent weight degeneracy, i.e., that the majority of the weight is concentrated on only a few particles. It discards low weighted particles and duplicates high weighted ones. This results in a new equally weighted set of particles for the next filter update. However, as noted in [Fet17], this also "leads to a decreasing diversity of particles" and therefore a high concentration of them at similar positions. In many cases, this leads to a poor approximation of the underlying probability density and worse estimation results, which is then called sample impoverishment.

Thus, in the context of indoor localization, it refers to a situation where the filter is unable to sample enough particles in the relevant regions of the building due to a high concentration of misplaced particles. As with resampling, this is due to restrictive transition models, such as not allowing particles to pass through walls. This restricts particles from moving freely within the dynamic system, which can lead to particles getting stuck in the corner of a room, for example, due to an access point malfunction or a wrongly detected turn. As the localizing pedestrian then moves further away from that corner, the Wi-Fi measurements become increasingly inappropriate, resulting in equal weighting of all particles, and the PDR continues to cause movement leading to a final misalignment. A very simple solution would be to randomly place a small

number of particles throughout the building. However, this leads to unpredictable behavior and increases uncertainty. There is a need for more robust and better methods.

Another common problem are multimodalities. A multimodality exists when the probability density has two or more highly probable hypotheses, called modes. In the context of state estimation, the challenges with such a density are obvious. It is necessary to ask the question "Which hypothesis (mode) is the correct one that best represents the inferences to be made?" or, in the context of this work, "Which hypothesis best represents the position of the pedestrian to be localized?".

One way to answer this question is to look at the state estimation procedure itself. Using a particle filter, the presumed best estimate can be found in many different ways. The standard procedures are either to calculate the weighted average over all particles or to find the particle with the largest weight. In the case of multimodalities, both procedures have obvious problems. While the latter represents the peak of a mode or a single displaced particle, the former calculates an estimate between modes. Of course, computing the full probability density function could solve the above, but as discussed in Section 1.1, it is an almost intractable task to find an analytical solution, which is the reason for using SMC in the first place, namely particle filtering. Numerical solutions, such as a *kernel density estimation* (KDE), are therefore a viable alternative, providing the "real" most likely state given by the maximum of the density estimation and thus avoiding the aforementioned problems. Unfortunately, such methods tend to consume a large amount of computational time, making them impractical for real-time scenarios [Bul18]. This creates a need for fast approximation algorithms.

Another way to deal with multimodalities is to prevent them in the first place. This means that measures are taken during the execution of the particle filter to prevent the density from splitting into multiple modes. Unfortunately, such procedures are rarely discussed in the literature and sometimes even deliberately ignored, e.g. in multi-target tracking. A very simple approach would be to assume an always Gaussian distributed state transition, which in the end is only an approximation of the Kalman filter and severely restricts the incorporation of environmental knowledge. One of the more commonly used methods is backtracking [WKB08]. In backtracking, the trajectories of all particles are stored in a history for a predefined time τ, also called the lag. If a particle makes a restricted movement at the current time t, e.g. crossing a wall, its trajectory is removed from the history. This improves the state estimation afterwards, because multimodalities up to $t - \tau$ are removed. The estimate must therefore be recalculated τ times per update, and the removed particles are redrawn at the next transition step. However, backtracking is based on very weak theoretical assumptions and violates several conditions of SMC. Moreover, it does not work for all systems, since the filtering procedure has to be modified and environmental constraints have to be taken into account.

A more general and profound approach is Bayesian smoothing, also known as backward filtering [DJ09; Sä13]. In contrast to backtracking, smoothing is able to use all knowledge that lies before the currently viewed time $t - \tau$. By running backwards in time, it allows to recursively optimize the successive densities, thus solving problems like multimodalities or other misbehavior that only become apparent with future knowledge. Although smoothing is mathematically well formulated, the application of these methods in the context of target tracking is still an open question with very few answers. One of the main reasons for this is probably the high computational complexity of most smoothing algorithms, which makes them unattractive for real-time systems.

The last open point is the widely discussed problem of parameter selection. Finding suitable or even optimal parameters (e.g., sensor noise) for the filter's probabilistic models depends on many factors. First of all, the quality of the used sensors and the chosen probabilistic representation (e.g. Gaussian distribution), but also the problem statement itself. In the context of indoor localization, the human factor in particular is an unknown quantity. For example, it can refer to the step length, the walking behavior, or the way the smartphone is held by the pedestrian. Most ILS developers choose their parameters based on heuristics or experience. A viable and widely used alternative is to estimate the parameters. Such estimation is classically approached by optimization methods such as gradient descent or least squares, expectation-maximization algorithms, or of course the filtering methods already discussed. Since parameter estimation is of great interest in many areas of science, the literature offers a wide range of solutions and approaches. A very good overview can be found in [Bos07] or more recent ideas in [CP20]. The author of [Sä13] offers a special consideration of such techniques in the context of Bayesian filtering.

As has been shown in the past, another factor is very important when selecting parameters, namely a balanced configuration between the individual sensor models. Consider a state evaluation procedure that combines a Wi-Fi multilateration and a magnetic matching sensor model. Now, the pedestrian walks into an isolated corridor where the Wi-Fi measurements are highly attenuated, while the magnetic fingerprints continue to provide reliable results. The combination of the two would still lead to a worse, or rather more uncertain, state space representation and thus a worse position estimation than the magnetic matching alone. A similar effect can also be observed with heterogeneous sensor types, which can only be integrated into the system in irregular and unpredictable intervals, e.g. with Wi-Fi and surveillance cameras. Although it is possible to counteract this phenomenon by parameterizing the respective sensor model, e.g. applying a weighting according to the area of the building, providing such parameters is often not generalizable and depends on the actual scenario [Fet23a].

A more promising approach is based on the concept of an interactive multiple model (IMM) [BSWT11]. Using a Markov transition matrix, it is possible to make a sensor model more or less influential based on previously selected quality features, such as the current Wi-Fi attenuation. In the context of sensor fusion, the broad class of such approaches is referred to as multiple model fusion. In a time-sequential, non-linear, and non-Gaussian dynamic system, this can be achieved by mixing between different, independently running particle filters using a method called interacting multiple model particle filter (IMMPF) [BD03]. It is described as a Markov chain process that provides a probability for each filter and a transition matrix for switching between them. This allows it to draw new particles from all available filters, not just its own. The final state estimate can then be drawn from the joint posterior distribution of the IMMPF, including all filters, with respect to some drawing strategy.

Despite its good properties, to the best of our knowledge, the IMMPF is rarely used for practical applications in sensor fusion research. For example, the authors of [CT18] use an IMMPF for visual tracking of single pedestrians in videos. By combining multiple movement models and visual cues, they were able to outperform other state-of-the-art approaches in this research area. In [Wan12] and [KMK16], the IMMPF is used for maneuvering target tracking in wireless sensor networks. They both mix three different movement models for velocity, rotation, and acceleration to model the motion patterns of the target. Despite the rare use of IMMPF, the potential for indoor localization is immense because modularity can be achieved not only at the sensor level, but also at the filter level. This means that any ILS based on particle filters can be combined as long as a Markov transition matrix can be specified. Therefore, it should be of interest to adapt the IMMPF approach to localization problems and to propose appropriate strategies.

1.3 Summary of Contributions

In the context of this work, the above mentioned problems are analyzed in great detail and a wide range of solutions based on established and new methods are proposed. The theoretical quality of these approaches always plays an important role to ensure their universal applicability. Furthermore, it is a stated goal that existing systems following the general approach of particle filtering can be easily extended by the methods presented here. In addition to solving these problems, improving the general estimation results is also an important aspect. In order to test the practical applicability, several application scenarios are presented and different solution approaches are discussed based on the theoretical considerations of the presented methods.

In the context of sensor fusion for indoor localization and navigation, the present work contributes innovations in the areas of particle filtering, multiple model fusion, and sample

impoverishment. Furthermore, a comprehensive smartphone-based ILS is presented for the analysis and validation of the proposed methods. The following contributions can be expected:

- **Indoor Localization System:** The methods presented in this thesis have the primary goal to improve the robustness and accuracy of an ILS. Therefore, we first present several techniques, methods, and models that can be used to create a hybrid ILS as a basis for all the improvements proposed in this work. In order to describe a wide range of systems, a general particle filter was chosen as the basis for position estimation. To represent the environment, i.e. the building, and to model only those movements that are actually feasible, e.g. no walking through walls, spatial models are integrated. We introduce a discrete movement model based on a regular tessellated graph and a continuous model based on an irregular tessellated navigation mesh. By applying them in the state transition, a PDR can be simulated. They require sensor information about the current motion of the pedestrian. Using the smartphone's IMU, we present methods for step detection and heading estimation that provide this motion. Furthermore, we present several novel sensor models for state evaluation. Besides the classical radio-based methods of Wi-Fi and BLE RSSI multilateration, we also present models for Wi-Fi FTM. To reduce setup and calibration time for access points or beacons, we will present data-driven models using optimization schemes. Especially in 3D scenarios, floor changes, e.g. by ascending stairs or riding an elevator, are a challenging scenario. To handle such situations, we present a sensor model for the barometer and two activity detection approaches. Together with the underlying spatial model and topological information, this enables continuous and smooth floor changes.

- **Multiple Model Fusion:** In the course of this work, we present a novel sensor fusion scheme for indoor localization by adapting an IMMPF to this type of dynamic systems. It promises absolute modularity and thus a very flexible and broad field of application. To achieve this, a balanced configuration of sources providing inferences about the pedestrian's position is achieved by treating each sensor, combinations of sensors, or even entire ILS independently as a single particle filter. For an overall estimation of the position, the respective filters are then combined using a mixing scheme as well as a novel joint state estimation procedure. To serve a real-time scenario, the matrix is implemented as a non-trivial Markov transition matrix that is frequently updated. It consists of different probabilistic quality factors of the involved sensors (e.g. current Wi-Fi quality), which then describe their influence in the overall system. These quality measures are established for the previously presented sensor models. To avoid numerical and temporal problems

such as sample degeneration, we extend the classical IMMPF approach and improve its dynamical properties.

- **Sample Impoverishment:** Finally, this work is dedicated to the elimination of sample impoverishment. As described before, sample impoverishment is a crucial problem especially in localization and navigation. The current literature provides only a few general approaches to deal with such situations. Therefore, we present an additional method based on the KDE and resampling that allows to sample from the real density instead of just the small subset of particles. Especially when the particle set is a bad approximation, this provides a slightly better dispersion. However, in the case of environmental constraints, e.g. when a system is stuck, the solutions usually fail or even make things worse. For this reason, some alternative strategies and procedures for avoiding and eliminating sample impoverishment in indoor environments are presented. These approaches can be categorized by effort and range from simple heuristic filter extensions using the underlying floorplan to the introduction of an additional support filter that monitors the current state of impoverishment using the Kullback-Leibler divergence.

1.4 Thesis Outline

As seen in chapter 1, a brief introduction of sensor fusion system as well as the current state of the art in pedestrian localization and navigation was given. We discussed the open problems from which most ILS suffer today and gave a small glimpse in this thesis contributions.

Chapter 2 deals with the theoretical foundations and concepts of sensor fusion using a probabilistic representation. First, some classical approaches are presented to highlight the advantage of Bayesian methods in the field of target tracking. Starting with the basic concept of Bayes' filter and Hidden Markov Model, the chapter goes on to introduce the large class of SMC methods and the necessary methods for state estimation and resampling. Then, the well-known CONDENSATION filter is examined in detail, and the most popular particle filter approaches are discussed. The chapter concludes with an application of the above to the use case of localization and navigation, explaining the theoretical foundations of our hybrid ILS.

The techniques, methods and models of the ILS are then presented in chapters 3 and 4. First, the used sensors and their probabilistic models are explained in detail to form the state evaluation process. To provide an implementation of the state transition, two motion models based on different spatial descriptions of a building floorplan are introduced in Chapter 4.

The general theory behind multi-model fusion is reviewed in Chapter 5. Then, the detailed derivation of the IMMPF is shown and its special properties are discussed in detail. Based on

this, we adapt it to indoor localization and present some novel extension for dynamic systems. Finally, we introduce sensor quality metrics for use within the time-sequential Markov process in the mixing stage of the IMMPF.

To tackle the problems occurring from sample impoverishment, different approaches for avoiding and eliminating them are presented in Chapter 6. Similar to the other chapters, we first present some state-of-the-art approaches that our own follow. Here, we distinguish between the categories general, usable without any information about the problem statement, and localization-specific methods.

Chapter 7 presents the experiments and evaluations of the methods presented above. As a basis, we use different configurations of the ILS in several different buildings. We focus on a realistic setting to first explain the occurring errors and phenomenons that can be observed in complex scenarios and then how they can be solved using the respective contributions of this thesis.

The thesis concludes with a summary and outlook on future work in chapters 8 and 9.

Chapter 2

Sensor Fusion

Consider again the definition of sensor fusion as given by [HM04]. According to him, sensor fusion "seeks to combine data from multiple sensors to perform inferences that may not be possible from a single sensor alone". In this context, an inference is represented as an environment, process, or quantity whose true state is unknown. In a more mathematical sense, the state $\boldsymbol{q}$ is an element of a corresponding state space $\mathcal{Q}$ of the system. This means that the state space is a set of all possible configurations of the system, compromised by the environmental model and any knowledge of how elements of this set are related. Depending on the scenario, the state space can be finite or infinite, while the state is given by a tuple of real numbers (a vector). From an indoor localization point of view, the state describes the position of the pedestrian and the state space all possible positions within the building.

Given the respective sensor data that yields an observation $\boldsymbol{o}$, the goal of sensor fusion is to infer the underlying state $\boldsymbol{q}$ using some system function f that maps the observations to the state; $f(\boldsymbol{o}) \mapsto \boldsymbol{q} \in \mathcal{Q}$. Since the state often changes with time and incoming observations $\boldsymbol{o}$, such functions are usually described by a time-sequential dynamical system [Sä13]. This means that for a given time interval only one future state can follow from the current state; $f(\boldsymbol{o}, \boldsymbol{q}_t) \mapsto \boldsymbol{q}_{t+1} \in \mathcal{Q}$. Depending on the problem to be solved or the assumptions made, such a system model has certain properties, e.g., linear, nonlinear, or Gaussian representation. It describes the most abstract level of the sensor fusion process and therefore takes into account all associated information, especially the uncertainty of the state and the respective sensor observations. This again leads to the conclusion that the main task in sensor fusion is the development of appropriate models to describe the uncertainty of the system. By providing an explicit measure of uncertainty, the information provided by sensors can be fused in an efficient and predictable manner [DWH16].

As outlined in Section 1.1, a distinction is made between probabilistic and non-probabilistic methods for describing and manipulating uncertainty. This thesis focuses on probabilistic methods, especially those based on the principles of Bayesian inference and filtering. They describe the models of the system using a Bayesian interpretation of probability. Here, probability is expressed as a degree of confidence in the underlying problem, e.g. in (self-)localization this can be formulated as the question "How likely is it that I am at this position?". Of course, this can be subject to certain properties, but this probabilistic, sometimes called belief, representation serves as a powerful and intuitive description of uncertainty. All models, as well as the sensors involved, accept this representation and try to answer the question in a statistical sense. In current research, Bayesian methods represent a large part of the solutions used for sensor fusion, and in particular for target tracking. For this reason, we distinguish between Bayesian and non-Bayesian methods.

In most scenarios, the goal of sensor fusion is to obtain a reasonable estimate of the true state $\boldsymbol{q}$ of the environment. Defined by [BSLK01]: "Estimation is the process of inferring the value of a quantity of interest from indirect, inaccurate, and uncertain observations". In a broader sense, sensor fusion can be seen as an estimation problem, where an uncertainty model is used to fuse different sensor modalities. In a probabilistic context, this often means finding the most likely state of an underlying probability density function. Again considering a time-sequential dynamic system, the estimation of a moving state is also referred to as filtering. When using prior knowledge, it is often called recursive filtering or recursive state estimation. Putting this into a (probabilistic) Bayesian context leads to the broad class of recursive Bayesian filtering, which is of particular interest in this work.

To provide the mathematical foundation for estimation, this chapter begins with a brief overview of estimation theory in statistics, especially point estimation. We will examine several fundamental principles and conclude by distinguishing between Bayesian and non-Bayesian approaches.

Starting with Section 2.2, we focus mainly on probabilistic methods that have a Bayesian interpretation. First, the principles of Bayesian inference and filtering are introduced mathematically, leading to the very important formulation of recursive Bayesian estimation. This can be seen as the foundation of all subsequent methods and thus plays an essential role in the field of sensor fusion, perhaps the most important in the current literature. It separates the system function (posterior) into three different probability densities, namely the prior, the transition, and the evaluation model. Depending on the problem, this representation allows for very different methods and properties of the respective models.

Another class of methods based on recursive Bayesian estimation is the aforementioned particle filter. It can be derived by putting the numerical approximation scheme of SMC into

a Bayesian context. As discussed, SMC is able to numerically solve any problem with a probabilistic interpretation using repeated random sampling. This allows to handle more complex scenarios involving multimodal densities or complex dynamics, especially pedestrian localization. Therefore, the remaining sections of the Chapter 2, are dedicated to SMC and the different particle filter implementations derived from it.

2.1 Estimation Theory

In a very broad sense, estimation theory is a branch of *statistical inference* that focuses on the process of estimating unknown parameters of a *statistical model* based on available data. These two fundamental concepts will be briefly introduced in the following, focusing on the application of sensor fusion. For a broader scope, the interested reader is also referred to the excellent textbooks by [Wil11] and [CB21].

Statistical inference is the process of drawing conclusions or making predictions about a population based on sample data. It involves using statistical methods to analyze and interpret the available data and to estimate the properties of the underlying population from which the data were drawn. Statistical inference can be broadly classified into two types:

1. *Estimation* involves using sample data to estimate the values of unknown parameters or characteristics of a population. For example, the average height of people in a city, the parameters of a radio propagation model or the position of a pedestrian.

2. *Hypothesis* testing uses sample data to test certain predictions, called hypotheses, about a population. It provides tools to evaluate the evidence for or against a hypothesis and quantify the level of uncertainty associated with our inferences. A hypothesis may be formulated as a test of relationship, e.g. whether there is a relationship between gender and height.

For making data-driven decisions, assessing the strength of evidence, and understanding variability and uncertainty in data, statistical inference provides a powerful set of tools and techniques. As such, it plays a central role in sensor fusion as well as in many other scientific fields, including biology, economics, and social sciences [CB21].

In the context of sensor fusion, statistical inference provides a set of models and methods for estimating the parameters of the underlying state space $\mathcal{Q}$, making predictions about the system dynamics, and assessing the uncertainty in the estimates. Therefore, the goal is to improve the accuracy, reliability, and robustness of the estimates while minimizing the effect of sensor errors and biases. Using a probabilistic representation, it provides a mathematical

framework to combine the information from multiple sensors in a statistically optimal way and to make inferences at decision-level, e.g., to decide whether an autonomous car should make an emergency stop based on multiple sensor inputs. In a nutshell, the task of statistical inference is to find an appropriate statistical model and then draw conclusions from one or more of those models.

A statistical model is a mathematical representation of a complex system, such as a sensor, state space, or psychological survey, that generates data. It embodies a set of statistical assumptions [Wil11]:

- *Distributional* assumptions refer to the assumptions made about the probability distribution of the data, especially random errors and uncertainty.
- *Structural* assumptions refer to the assumptions made about the relationships between the variables in the statistical model, such as linearity, additivity, or interaction effects.
- *Cross-variation* assumptions refer to the assumptions made about the correlations or covariances between the variables in the model, such as stationarity, autocorrelation, or independence.

The model aims to capture the underlying structure and relationships between the variables in the system, as well as the uncertainty in the data generation process. In an idealized form, both can be described by unknown model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, observing a finite set of random variables $\boldsymbol{O}_1, ..., \boldsymbol{O}_n$ with corresponding realized values (observations) $\boldsymbol{o}_{1:n} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n\}$, which satisfy an *independent and identically distributed* (i.i.d.) as well as a continuous probability distribution $p(...)$. This leads to a typical model that draws $\boldsymbol{o}_i$ from a population distributed as follows

$$\boldsymbol{o}_i \sim p(\boldsymbol{o}_i \mid \boldsymbol{\theta}) \ . \tag{2.1}$$

It describes the conditional probability of receiving an observation given $\boldsymbol{\theta}$. In the simplest case, the joint probability can be given by

$$p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) = p(\boldsymbol{o}_1, ..., \boldsymbol{o}_n \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{o}_i \mid \boldsymbol{\theta}) \ , \tag{2.2}$$

where a statistically independence between the single observations is assumed and all individual distributions are sharing an identical set of parameters. Inferences about $\boldsymbol{\theta}$ are thus based on this distribution. Depending on the underlying state space $\mathcal{Q}$, its assumptions and the uncertainty of observations $\boldsymbol{o}$, the quantity $\boldsymbol{\theta}$ can be either finite-dimensional (parametric model) or infinite-dimensional (non-parametric model) [BSLK01]. For example, if the noise of incoming

observations can be assumed to be Gaussian, the system yields a parametric model described by a normal distribution $\mathcal{N}(\mu, \sigma^2)$ and is therefore completely determined by finding $\boldsymbol{\theta} = (\mu, \sigma^2)$. In contrast, non-parametric models are determined solely by the data, and thus are usually not defined a priori. Common examples of such models are *support vector machines* (SVM), which fits a hyperplane for data separation and classification, or a *kernel density estimation* (KDE), which attempts to estimate a probability density function.

Of course, the assumptions made for (2.2) often do not apply to sensor fusion scenarios. An example would be the use of Wi-Fi access points throughout the building, where radio signals on the same channel are likely to interfere with each other and therefore interact in some way. A simple workaround is to have each sensor provide an independent model, which is then later combined by assuming statistical independence between the models, but not necessarily between the respective observations [Sä13]. The combination of these so-called sensor models is itself a statistical model that tries to infer some overarching problem. To better understand this for the purpose of indoor localization, we need to go back to the different categorizations of sensor fusion systems (cf. Section 1.1) and match this concept to the statistical models. Most sensors are not able to provide a direct measure of the pedestrian's position and preferably her heading, e.g. Wi-Fi provides RSSI measures and IMU provides acceleration and angular velocity. This is why we consider sensor fusion in the first place. Solving the localization problem can thus be referred to as decision-level fusion, since inferences cannot be made directly from observed data, but only from a combination described by an appropriate statistical model, where the unknown quantity $\boldsymbol{\theta}$ represents position and heading. On the other hand, combining an accelerometer and a gyroscope to infer a $\boldsymbol{\theta}$ that inherits the pedestrian's relative velocity and/or heading can be considered feature-level or data-level fusion. So at the lowest level, not necessarily fusion level, a statistical model of a sensor, in the sense of (2.1), tries to find a probability distribution that best describes the sensor error.

Regardless of the level of abstraction of the statistical model, a solution for the unknown $\boldsymbol{\theta}$ is always sought. The methods and tools for doing so are summarized under the term *estimation theory*. Consider again the problem of inferring $\boldsymbol{\theta} \in \mathcal{Q}$, now also called the estimand, using the i.i.d. random variables $\boldsymbol{O}_1, ..., \boldsymbol{O}_n$ with corresponding realized values (observations) $\boldsymbol{o}_{1:n} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n\}$. A point estimator $\hat{\boldsymbol{\theta}}$ is then denoted as a function $\hat{\boldsymbol{\theta}}(\boldsymbol{o}_{1:n})$ of these observations, providing a fixed value of the sample space $\mathcal{Q}$, often called the estimate. Its counterpart, the interval estimator, instead yields a range of plausible values. Thus, the goal of an estimator is to maximize the knowledge about $\boldsymbol{\theta}$ in a statistical model, i.e. maximizing the probability density in (2.2). For the sake of simplicity, we use the abbreviated notation $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(\boldsymbol{O})$ and interpret $\hat{\boldsymbol{\theta}}$ directly as a random variable.

The corresponding estimation error is defined as

$$e(\boldsymbol{o}_{1:n}) = \hat{\boldsymbol{\theta}}(\boldsymbol{o}_{1:n}) - \boldsymbol{\theta} \ , \tag{2.3}$$

denoting the basis for optimality between the estimated parameters and the actual value of the parameters. Other relevant quantified properties are the *mean squared error* (MSE) of $\hat{\boldsymbol{\theta}}$ given by

$$\mathrm{MSE}(\hat{\boldsymbol{\theta}}) = \mathrm{E}[(\hat{\boldsymbol{\theta}}(\boldsymbol{O}) - \boldsymbol{\theta})^2] = \mathrm{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2] \ , \tag{2.4}$$

the variance

$$\mathrm{Var}(\hat{\boldsymbol{\theta}}) = \mathrm{E}[(\hat{\boldsymbol{\theta}} - \mathrm{E}[\hat{\boldsymbol{\theta}}])^2] \ , \tag{2.5}$$

and the bias

$$B(\hat{\boldsymbol{\theta}}) = \mathrm{E}(\hat{\boldsymbol{\theta}}) - \boldsymbol{\theta} \ , \tag{2.6}$$

where $\mathrm{E}(\hat{\boldsymbol{\theta}})$ is the expected value of the estimator. For a finite number of realized values $\boldsymbol{o}_{1:n}$ it resembles a (probability-) weighted average, which is the case for standard sensor models. The $\mathrm{MSE}(\hat{\boldsymbol{\theta}})$ is a measure of the quality of an estimator, indicating how far a set of estimates is on average from the estimand. It can also be written as $\mathrm{MSE}(\hat{\boldsymbol{\theta}}) = \mathrm{Var}(\hat{\boldsymbol{\theta}}) + (B(\hat{\boldsymbol{\theta}}))^2$, what shows the relationship between bias and variance. In case of an unbiased estimator, i.e. the distance between the average of estimates and the estimand is zero, we can write $\mathrm{MSE}(\hat{\boldsymbol{\theta}}) = \mathrm{Var}(\hat{\boldsymbol{\theta}})$. An example where this might happen would be observations that are perfectly Gaussian distributed, and thus $\mathrm{E}(\hat{\boldsymbol{\theta}})$ is identical to the mean. Other well-known and popular quantified properties are the Fisher information, which provides a measure of the information that the observations carry about the unknown parameter $\boldsymbol{\theta}$ as well as further knowledge about the (co)variance, and the related Cramér-Rao bound, which expresses a lower bound on the variance of an unbiased estimator as well as inferences about their efficiency [CB21].

Apart from the quantified properties of an estimator, there are also behavioral properties, often using the aforementioned, to consider when selecting an appropriate estimator for a particular statistical model. One of these quantified properties is the efficiency of an estimator. It measures how well $\hat{\boldsymbol{\theta}}$ utilizes the available information in the observations $\boldsymbol{o}_{1:n}$ to estimate $\boldsymbol{\theta}$. In other words, it provides knowledge about how well the estimator uses the data to get as close to the true value of the parameter as possible. This measure is based on the choice of a particular loss function, most commonly $\mathrm{MSE}(\hat{\boldsymbol{\theta}})$. For optimality, the goal is to find the *minimum-variance unbiased estimator* (MVUE) that yields the smallest variance among all unbiased estimators, satisfying the Cramér-Rao bound for all possible values of $\boldsymbol{\theta}$, and is thus said to be efficient. In practice, however, an MVUE does not always exist, which then requires a loss function that best fits a particular context or application. This makes the property of being

unbiased particularly desirable. Other quantified properties are consistency, which measures how well an estimator approaches the true value of the parameter as the sample size increases, or robustness, which indicates how well an estimator performs in the presence of outliers or deviations from the underlying assumptions of the statistical model [Wil11].

2.1.1 Maximum Likelihood Estimation

The most widely used method for estimating parameters of statistical models is the maximum likelihood method. As explained above, the general goal of an estimator is the to determine the parameters $\boldsymbol{\theta}$ of a statistical model by finding the values that give the observed data the highest joint probability. To do this, maximum likelihood methods use the likelihood function of the joint probability density in (2.2). Consider again a statistical model denoted as (2.1) the likelihood function is then

$$\boldsymbol{\theta} \mapsto f_{\mathcal{L}}(\boldsymbol{o}_i \mid \boldsymbol{\theta}) \ . \tag{2.7}$$

It returns the probability of a given observation $\boldsymbol{o}_i$ (realized value, not random value) as a function of $\boldsymbol{\theta}$. In other words, a probability density function of $\boldsymbol{o}_i$ is given when $\boldsymbol{\theta}$ is fixed, and a likelihood function of $\boldsymbol{\theta}$ is given when $\boldsymbol{o}_i$ is fixed. Thus to prevent confusion the likelihood is commonly written as $\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_i)$. For our density of interest (2.2) this results in

$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) = \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_1, ..., \boldsymbol{o}_n) = p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) \ , \tag{2.8}$$

what we will refer to as *the* likelihood function throughout this work.

Finding the maximum of the likelihood function for a given statistical model with respect to $\boldsymbol{\theta}$ yields the *maximum likelihood estimator* (MLE).

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}} \, \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) \ . \tag{2.9}$$

That is, $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ are the estimated parameters that provide the maximum probability given the observations $\boldsymbol{o}_{1:n}$. The MLE thus belongs to the class of extremum estimators, which exhibit asymptotic behavior in the sense that it converges *almost surely* to the true value of the parameter [RC13]. This means, with an observation size n increasing to infinity the estimator $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ converges in probability to its true value, if for all $\epsilon > 0$

$$\lim_{n \to \infty} P(e(\boldsymbol{o}_{1:n})_{(2.3)} > \epsilon) = 0 \ . \tag{2.10}$$

With i.i.d. observations, the model only needs to be identifiable, compact, and continuous, which is often the case. In addition, the MLE is also an efficient estimator since it satisfies the Cramér-Rao bound as n approaches to infinity.

A common algebraic simplification for extremum estimators is to use the *log likelihood*. The logarithmic transformation of (2.9) is

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta}\in\mathcal{Q}}{\operatorname{argmax}} \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) \ . \tag{2.11}$$

This takes advantage of the fact that the logarithm of the product is the sum of the logs and that the logarithmic function does not change the extremum due to its monotonic behavior. Especially for probability density functions from the exponential family, such as Gaussian or Poisson, which are logarithmically concave, maximization is made more convenient. It is interesting to note that for the same set of densities, the standardized least squares and maximum likelihood estimates are identical [CFY76].

As an example that can be solved using an MLE consider the probability density function of a one-dimensional Gamma distribution defined as

$$\Gamma(x \mid \alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta} \ . \tag{2.12}$$

In terms of the methods introduced above, let $p(\boldsymbol{o}_i \mid \boldsymbol{\theta})_{(2.1)} = \Gamma(\boldsymbol{o}_i \mid \alpha, \beta)$ with $\boldsymbol{\theta} = (\alpha, \beta)$ be our statistical model describing the i.i.d observations $\boldsymbol{o}_{1:n} = \{o_1, ..., o_n\}, o \in (0, \infty)$. The log likelihood according to (2.8) and (2.11) is then

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) &= \log \prod_{i=1}^{n} \Gamma(o_i \mid \alpha, \beta) \\ &= \log \prod_{i=1}^{n} \frac{1}{\Gamma(\alpha)\beta^{\alpha}} o_i^{\alpha-1} e^{-o_i/\beta} \\ &= -n \log \Gamma(\alpha) - n\alpha \log \beta + (\alpha - 1) \sum_{i=1}^{n} \log o_i - \sum_{i=1}^{n} \frac{o_i}{\beta} \ . \end{aligned} \tag{2.13}$$

Assuming that α is known, the MLE of β can be found very simple by making the first derivative test, i.e. solving $\frac{\delta}{\delta\beta} \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) = 0$, what results in

$$\hat{\beta}_{\hat{\boldsymbol{\theta}}_{\text{MLE}}} = \frac{1}{n\alpha} \sum_{i=1}^{n} o_i \ . \tag{2.14}$$

If α is also unknown, solving $\frac{\delta}{\delta\alpha} \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) = 0$ in addition results in a much more difficult computation, which yields

$$\log(\alpha) - \psi(\alpha) = \log\left(\frac{1}{n}\sum_{i=1}^{n} o_i\right) - \frac{1}{n}\sum_{i=1}^{n}\log(o_i) \ , \tag{2.15}$$

where ψ is the derivative of the gamma function $\Gamma(\alpha)$, the digamma function. As can be seen, this is not an explicit solution, and even worse, there is no closed-form solution for α [RC13]. The authors of [YC17] introduced closed-form estimators for α and β derived from the likelihood equations of the generalized gamma distribution. Although these estimators have similar efficiency to MLE, they are not explicit extremum estimators and are therefore referred to as mixed-type log-moment estimators.

In most cases, there is no closed-form MLE solution for sensor fusion applications, what requires for numerical approximation. Even the properties of consistency and efficiency are of no use, because a sensor model is only an idealized form of the process generated by the data. A very famous aphorism, popularized by the statistician George Box, is that "all models are wrong, but some are useful" [Box76]. The point is that despite the usefulness and advances of statistical inference and estimation theory, a model can never account for the complexities of the real world. Nevertheless, we can try to use as many information and knowledge sources as possible to reach the desired inferences. By combining multiple sensor models to answer (2.2), we can at least get a little closer to reality. However, an important factor not considered by MLE is possible prior knowledge, e.g. about the initial distribution of the state space or its boundary behavior. In the indoor localization example, the state space is bounded by the walkable area of the building and may provide assumptions about the initial position. This leads to the next class of estimators.

2.1.2 Maximum a Posteriori Estimation

In contrast to MLE, the co-called *maximum a posterior estimator* (MAP) makes use of possible prior knowledge by assuming an a priori distribution with density $p(\boldsymbol{\theta})$. If we now treat $\boldsymbol{\theta}$ as random variables and apply the principles of Bayesian statistics, the conditional probability $p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})$ (cf. (2.2)) of the data and the a priori $p(\boldsymbol{\theta})$ can be derived by using the Bayesian formula:

$$p(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) = \frac{p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{o}_{1:n})} \ . \tag{2.16}$$

Here, $p(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n})$ is the so-called posterior distribution, which denotes a classical conditional probability distribution. Similar to (2.8) the probability $p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})$ can be interpreted as a likelihood function.

As an optimization criterion, MAP uses the posterior of the model parameters (2.16) instead of maximizing the likelihood as in (2.9), yielding

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}}\, p(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}}\, \frac{p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\boldsymbol{o}_{1:n})} = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}}\, p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \ . \quad (2.17)$$

The marginal distribution $p(\boldsymbol{o}_{1:n}) = \int p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is always positive and constant. It does not depend on $\boldsymbol{\theta}$ and can therefore be ignored in the maximization process. The marginal is also called model evidence because it can be used in model comparison to determine the contribution of the observations to the posterior of the associated model. As we will see later, it is interpreted as a normalization factor, since in most cases the posterior and the sensor models are not proper distributions, i.e. their integral does not sum to 1 [BS09]. For example, assuming a statistical independence between the respective sensor models that are combined in a scheme like $p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) = p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})_1 \cdot p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})_2$ inevitably requires a normalization. Since Bayes' Theorem can only be applied to proper distributions, (2.16) is often seen as a definition rather than an application of the theorem. However, this does not matter for the use cases discussed in this thesis.

Comparing (2.9) and (2.17), the MAP differs from the MLE only in the addition of an appropriate a priori $p(\boldsymbol{\theta})$. Consequently, using a uniform distribution $p(\boldsymbol{\theta}) \propto 1$ as a priori results in an MLE. As we have seen, the MLE is in principle optimal for obtaining an estimate from observations alone, as long as the number of observations is not limited, i.e., asymptotic properties can come into play. In the context of sensor fusion, this is obviously an untenable assumption, since the number of sensor measurements, especially in real-time applications, is fundamentally limited by factors such as temporal resolution (e.g., frequency of the sensor), physical conditions (e.g., absorption of radio signals), or spatial constraints (e.g., transmitter range). By adding the a priori probability, the absence of sensor measurements can be compensated by appropriate prior knowledge. For this reason, parameter estimation using MAP represents the state-of-the-art for sensor fusion systems in the field of indoor localization [Guo20; SVW22]. This is because, as discussed at the end of the previous section, there is a lot of prior knowledge about the posterior in this particular application.

As a simple example for MAP, consider a sequence of i.i.d observations $\boldsymbol{o}_{1:n} = \{o_1, ..., o_n\}, o \in (0, \infty)$ that are distributed according to an univariate Gaussian:

$$\mathcal{N}(o \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{o-\mu}{\sigma}\right)^2} \quad , \tag{2.18}$$

assuming the variance σ^2 is known and we want to estimate the mean μ. Thus, we can define that the unknown parameters $\boldsymbol{\theta} = \mu$, the model $p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta}) = \mathcal{N}(o \mid \mu, \sigma^2)$ and the a priori $p(\boldsymbol{\theta}) = \mathcal{N}(\mu \mid \mu_0, \sigma_0^2)$ with known initial mean μ_0 and variance σ_0^2. By ignoring the marginal $p(\boldsymbol{o}_{1:n})$, as discussed above, the maximization in (2.17) takes the form:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}} \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) p(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}} \prod_{i=1}^{n} \mathcal{N}(o_i \mid \mu, \sigma^2) \mathcal{N}(\mu \mid \mu_0, \sigma_0^2) \quad . \tag{2.19}$$

To solve this we make the first derivative test and again apply a logarithmic transformation for simplification, what leads to

$$\frac{\delta}{\delta\mu} \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n}) p(\boldsymbol{\theta}) = 0 \Leftrightarrow \frac{\delta}{\delta\mu} \sum_{i=1}^{n} \log \mathcal{N}(o_i \mid \mu, \sigma^2) + \log \mathcal{N}(\mu \mid \mu_0, \sigma_0^2) = 0 \quad . \tag{2.20}$$

After some algebraic transformations (see [Fin14] for the complete proof) and solving for μ, the MAP estimate is then given by

$$\hat{\mu}_{\text{MAP}} = \frac{\sigma_0^2 \left(\sum_{i=1}^{n} o_i\right) + \sigma^2 \mu_0}{\sigma_0^2 \, n + \sigma^2} \quad . \tag{2.21}$$

This provides an analytical solution for computing an estimate. It can be interpreted as a linear interpolation between the variance-weighted means of the a priori and the likelihood function. If the observation size $n \to \infty$ or the prior variance $\sigma_0 \to \infty$, the a priori becomes *non-informative* since it no longer affects the estimate. Consequently, the MAP will converge to the MLE; writing $\hat{\mu}_{\text{MAP}} \mapsto \hat{\mu}_{\text{MLE}}$.

Nevertheless, finding an analytical solution for realistic scenarios only happens in rare cases. One possibility is the existence of a so-called conjugate prior for the likelihood. According to the Pitman-Koopman-Darmois theorem, a statistical model has a conjugate prior if and only if there exists an a priori $p(\boldsymbol{\theta})$ that, when combined with the likelihood, yields a posterior that is in the same probability distribution family as $p(\boldsymbol{\theta})$. This is especially true for the exponential family, such as the Gaussian, Poisson, or Gamma distributions [Fin97].

Now, modifying the above example and assuming that the variance σ^2 of the observations is also unknown, we get $\boldsymbol{\theta} = (\mu, \sigma^2)$. The model $p(\boldsymbol{o}_{1:n} \mid \boldsymbol{\theta})$ and the a priori distribution

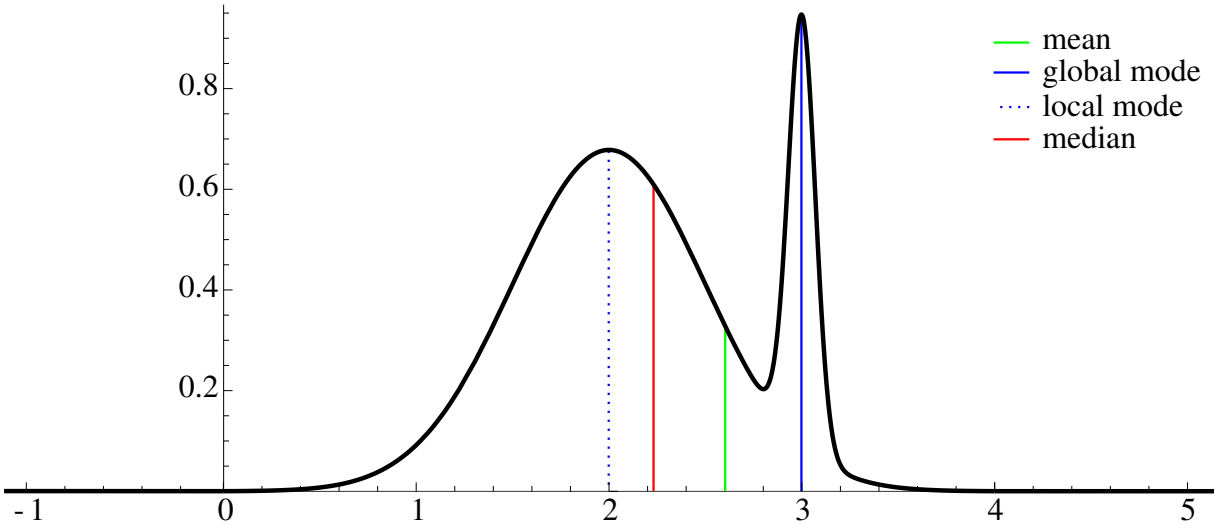


Figure 2.1: A univariate bimodal probability distribution as a posterior $p(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n})$ for point estimation. The three estimates of mean, global mode, and median are shown as solid lines. While the global maxima (MAP) represents the value with the highest probability, it neglects the other, less likely but relevant, values at the local maxima.

of the mean $p(\mu)$ still follows a standard univariate Gaussian. In contrast, the a priori of the variance $p(\sigma^2)$ is assumed to be an inverse gamma distribution with shape parameter α and scale parameter β, i.e. $\sigma^2 \sim \text{Inv-}\Gamma(\alpha, \beta)$. Having the joint a priori for the mean and the variance $p(\mu, \sigma^2) = p(\mu) \cdot p(\sigma^2)$ changes (2.19) according to

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta} \in \mathcal{Q}}{\operatorname{argmax}} \prod_{i=1}^{n} \mathcal{N}(o_i \mid \mu, \sigma^2) \mathcal{N}(\mu \mid \mu_0, \sigma_0^2) \text{Inv-}\Gamma(\sigma^2 \mid \alpha, \beta) \ . \tag{2.22}$$

By means of the Pitman-Koopman-Darmois theorem the joint a priori $p(\mu, \sigma^2)$ is not conjugate to the likelihood, as μ and σ^2 appear together in a non-factorized way in the likelihood; hence they will also be coupled together in the posterior [Mur12]. Since both the likelihood and the mean follow a Gaussian, this relationship is called semi-conjugated.

The two examples above demonstrate that a closed-form solution is simply not to be expected in practice, since even the simplest models often do not fulfill the necessary criteria. As we have already seen, the state space $\mathcal{Q}$ can quickly become very large, depending on the set of sensors to be fused, and the individual sensor models are probability density functions that cannot be parameterized. Therefore, searching for an analytical solution is futile and appropriate numerical approximation methods such as Markov Chain Monte Carlo (MCMC) sampling, regression analyses or other optimization algorithms must be applied to obtain a valid estimate.

Regardless, MAP methods often have difficulty representing complex probability densities, such as those found in sensor fusion, through estimation. Looking at the bimodal posterior

distribution from Figure 2.1 as an example, we see that the highest mode, i.e. the point where the posterior probability and thus the estimate of the MAP is largest, is not characteristic of the overall distribution. An alternative is the more versatile class of Bayesian estimator. They minimize the posterior expected value of a loss function, such as the MSE (cf. (2.4)). Using the MSE as loss results in the mean of the posterior, also known as the *minimum mean square error* (MMSE) estimator. Changing the loss function also changes the Bayes estimator, which allows inferences to be made about certain properties, e.g., robustness to outliers in the underlying model by choosing the sum of absolute deviations to minimize [RC13].

In classical textbooks such as [Ber13], it is argued that the mean is often a better estimator then the mode of a distribution. However, this might not be true for all use cases. For example, in classification, the well-known Bayes classifier chooses the class that has the largest posterior of occurrence, which is MAP. Under the theoretical assumption the posterior is known, this proofs to be the optimal classifier. A similar assertion can be made for point estimators under the assumption of optimal sensor models. Interestingly, the MAP can be easily obtained from the more general Bayes estimator by choosing a $0-1$ loss function of the form

$$f_{\text{loss}}(a, \boldsymbol{\theta}) = \begin{cases} 0, & a = \boldsymbol{\theta}, \\ 1, & \text{otherwise}, \end{cases} , \tag{2.23}$$

where $a \in \mathcal{Q}$ is the value minimizing the expected value of the loss. Based on this, it is commonly accepted that the MAP is a limiting case of the Bayes estimator, although it has been disproved for some edge cases, as shown in [BD19]

Despite the theoretical properties discussed above, point estimator can often only make limited or summary statements about problems in sensor fusion. It is obvious that by combining many (physically) different sensor models, the complexity of the posterior quickly becomes very high. This is often accompanied by an increase in the dimensions of the unknown parameters, which poses problems even for numerical estimation and optimization methods. In practice, therefore, a more sophisticated approach is often chosen, in that first the parameters of the sensor models are each determined, and the actual higher-level problem to be answered, e.g. indoor localization, is done in a subsequent step. This has the great advantage that the supposedly simpler solo sensor models can be determined independently, perhaps even based on pre-existing measurements. In the overall system, their parameters can then be considered as constants, and only an estimation on the state space of the use case relevant to the fusion needs to be performed.

Given the above, it is often useful to estimate the entire probability density function rather than a point estimate. Since the posterior is not in closed form, density estimation is achieved

using numerical sampling methods that allow drawing from arbitrary probability densities. By drawing a sufficiently large number of samples, the posterior can be approximated. Such a representation allows a variety of possibilities for the final parameter estimation. These can be heuristics, such as excluding individual samples based on prior knowledge, or analytical methods, such as a kernel density estimator. One method that makes this possible is the Monte Carlo method, which will be discussed in Section 2.3.

One factor that has not been discussed in context of estimation theory are the time-sequential properties that often underlie real-time sensor fusion models. This means that as time progresses, new measurements become available, but an estimate must be available at each point in time. This estimate, e.g., the position of a person, claims to be up-to-date, i.e., it should indicate the position at a point in time that is as recent as possible. This requires the definition of discrete-time stochastic processes, which at best are also applicable to nonlinear and non-Gaussian distributed systems. In the following section, we will introduce such processes based on Bayesian inference.

2.2 General Bayes Filter

Up to this point, we have introduced and discussed solutions for estimating arbitrary quantities of statistical sensor models. It was shown that in a Bayesian context we have the possibility to add a priori knowledge, which is a massive gain for the field of sensor fusion. This can be knowledge about the system, e.g. the building, or about the respective sensors, e.g. an initial position of an IMU. However, the way we have introduced estimation theory is limited because we can only estimate a point based on a set of measurements that must be known in advance. In the context of sensor fusion, it therefore makes sense to look for methods that can bring the advantages of Bayesian inference, as already used in MAP, into a time-sequential setting while providing an overall estimate of the posterior.

In order to delve into the stochastic processes of (time-)sequential state estimation, we must first update the mathematical description of our system that we established at the beginning of this chapter and discuss the resulting insights. Again, the goal of sensor fusion is to infer the underlying state $\boldsymbol{q}$ using some system function f that maps the observations to the state; $f(\boldsymbol{o}) \mapsto \boldsymbol{q} \in \mathcal{Q}$. According to the Bayesian context agreed on and the concepts introduced in Section 2.1, the function f can be denoted by the probability distribution (posterior) $p(\boldsymbol{\theta} \mid \boldsymbol{o}_{1:n})$. Describing this in a time-sequential manner and according to (2.16) results in

$$p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T}) = \frac{p(\boldsymbol{o}_{1:T} \mid \boldsymbol{q}_{1:T}) p(\boldsymbol{q}_{1:T})}{p(\boldsymbol{o}_{1:T})} \quad , \tag{2.24}$$

where the unknown quantity $\boldsymbol{\theta}$ is now a vector valued time series of hidden states $\boldsymbol{q}_{1:T} = \{\boldsymbol{q}_1, ..., \boldsymbol{q}_T\}$ that corresponds to noisy observations $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_T\}$. Since we are investigating indoor localization, we can assume a physical system from which inferences about a pedestrian's possible whereabouts are made. By putting this in a temporal and probabilistic context, this can be further reduced to a stochastic dynamic system. A common way of representing such systems is the state space representation [TBF05]. Now this all sounds quite meaningful, but these are simply generic terms that summarize different use cases of mathematical functions and models. Somewhat disrespectfully, one could also say that they are the buzzwords that belong to the topic of this thesis. The underlying system is obviously physical due to the sensors involved, dynamic due to human motion over time, and finally stochastic due to probabilistic considerations. The term state, instead of quantity or parameter as before, arises from the physical view, where the system assumes a time-dependent state $\boldsymbol{q}_t$ described by a point in the state space $\mathcal{Q}$.

The joint distribution in (2.24) describes the full posterior of all the states given all the observations. However, this means that for each time step and each incoming observation, the full posterior distribution must be recalculated to obtain an updated estimate. As a consequence, the dimensionality of the density and thus the complexity of the computation increases with each time step. In fact, if no further assumptions can be made or additional information is available, the problem is so severe that the computations eventually become intractable, regardless of the computing power available [DJ09]. Instead of solving the entire posterior function per time step as in (2.24), it is common to determine only the marginal distributions of the states to simplify the computations by an order of magnitude.

In real time applications the calculation of $p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T})$ would be unnecessary anyway, since we are basically only concerned with finding the current state $\boldsymbol{q}_t$. The trajectory $\boldsymbol{q}_{1:T}$ is then automatically obtained over time by storing the respective states in a history. This means that for real-time sensor fusion we are interested in a marginal distribution of the form

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}), \quad t = 1, 2, ..., T \ . \tag{2.25}$$

This is commonly known as the filtering distribution. Here, the time index t is discrete and the *current and previous* observations $\boldsymbol{o}_{1:t} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_t\}$ have equally-spaced time intervals. In order to perform the marginalization from (2.24) to (2.25), special probabilistic properties must be assumed for the dynamical system. A stochastic process that satisfies these properties is the well-established Markov model. In the presence of a dynamical system, this leads to the class of *hidden Markov models* (HMM), which will be presented in the next chapter. Bringing all this back into a Bayesian context, we finally get the general probabilistic approach of Bayesian

filters, whose goal is to estimate an unknown probability density function for a distribution like (2.25).

2.2.1 Hidden Markov Model

Markov models are well established methods for making probabilistic inferences about a series of events, where the probability of each event depends only on the states attained in the previous event. This means that instead of making a prediction about the entire sequence of events, only the preceding states are considered. When only the immediate predecessor $\boldsymbol{q}_{t-1}$ is used, it is called a first-order Markov process that satisfies the Markov property, i.e. given the state $\boldsymbol{q}_{t-1}$, the subsequent $\boldsymbol{q}_t$ does not depend on any $\boldsymbol{q}_{1:t-2}$. It can be visualized as a finite state automaton with edges between any pair of states which are labeled with transition probabilities characterized as follows:

$$P(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1}) = P(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) \ . \tag{2.26}$$

Using a conditional probability density as in the state estimators in the previous section, statistical independence holds between all prior states $\boldsymbol{q}_{1:t-2}$ and also between observations $\boldsymbol{o}_{1:t-2}$. This results in the simplified probability distribution:

$$P(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1}, \boldsymbol{q}_{1:t}) = P(\boldsymbol{o}_t \mid \boldsymbol{q}_t) \ . \tag{2.27}$$

A brief introduction in the topic of Markovian stochastic with many examples is provided by [Gag17].

The use of a Markov models is particularly advantageous when it can be assumed that the probabilistic state changes of the system to be modeled, previously referred to as dynamics, only affect each other over a limited period of time. This is often the case with sensor fusion, as measurements become outdated over time and no longer accurately represent the current state. This increases the uncertainty. Especially in real-time systems with high-dimensional and/or continuous state spaces, outdated measurements are usually undesirable and considered an error factor. For example, in the self-localization use case described in this thesis, the position estimation should be based on recent measurements so that a pedestrian moving in the building can get the most accurate result possible without significant delay. On the other hand, if the parameters of a stationary object are estimated, or if the state describes a static probability density, then the system is less complex and the Markov assumption can be dropped in favor of classical estimation methods, which in the best case converge with increasing amount of measurement data. At first glance, one might think that a Markov model is a limitation, but

we will see that for our use case there are almost exclusively advantages in the mathematical description and numerical implementation.

The mathematical formulation and the properties of the Markov model depend on whether the state space representation and the time series each take a discrete or continuous form [Gag17]. In applications that attempt to make inferences about a physical object in a real-world environment, such as target tracking of missiles, the state space $\mathcal{Q}$ is usually considered to be continuous, since the object can take any arbitrary position. A more simple example is the angular position of an undamped pendulum [AP90]. In contrast a discrete consideration of $\mathcal{Q}$ is present, when the number of states is finite (or countable like $\mathbb{Z}$). For example, in the game of chess, where the effective state space is the set of positions that can be reached by moves that are legal in the game. In some applications, however, continuous state spaces can be simplified to discrete state spaces. For example, if for localization purposes we are not interested in the most accurate position estimation possible, but only in individual areas within a building, or if the sensors used are not capable of mapping with such high accuracy, a equidistant grid can be used. The boundaries are given by the size of the building, resulting in a finite and countable set of grid-cells. Each cell then describes a possible state as a position of localization. In the literature, this state space model is also referred to as an occupancy grid [TBF05].

In the time domain, the distinction between discrete and continuous can be more clearly defined. In short, a discrete-time signal always has a countable domain, such as natural numbers, while a continuous signal is a varying quantity whose function domain is an uncountable set. Thus, the latter is represented by a function that describes the behavior of the system considering time as a continuous variable. These systems are often used to model phenomena that occur in a continuous and uninterrupted manner, such as theoretical models that explain the behavior of physical systems. Very simple examples are the trigonometric functions over time, i.e. $f(t) = \sin(t)$. Discrete time, on the other hand, applies to a system in which variables change only at specific points in time, represented by sequences of observed or measured values. Consequently, in the context of sensor data, continuous time refers to the underlying physical phenomenon being measured, while discrete time refers to the recorded measurements $\boldsymbol{o}_{1:T}$ from the sensor at specific times t. This process of measuring the continuous-time signal at discrete time intervals is commonly known as sampling.

For the sensor fusion system considered at the beginning of Section 2.2, this generally results in a discrete time series, while the state space depends on the specific application. Due to the mostly trivial mapping from a continuous to a discrete state space, we only have to assume that it can be measured. This results in a time-discrete Markov model with measurable state space [Gag17].

The careful reader will have noticed that when we introduced $\boldsymbol{q}_{1:T}$, we referred to them as hidden states. In our functional description of sensor fusion, we defined the goal to infer $\boldsymbol{q}$ from observations $\boldsymbol{o}$ of one or more sensors; $f(\boldsymbol{o}) \mapsto \boldsymbol{q} \in \mathcal{Q}$. For example, in Section 2.1.1 we tried to estimate the parameters of a Gamma distribution using a set of observations. The parameters define a statistical model that directly describes the sensor uncertainties. This is called an observable state. If $\boldsymbol{q}$ cannot be observed directly, but can only be inferred from indirect measurements, one speaks of a latent state, or more generally, without the context of a state space, of latent variables.

Latent states, which can represent aspects of physical reality, encompass both hidden states and hypothetical states in the context of sensor fusion. The latter, also known as hypothetical constructs, refer to abstract concepts that are not directly observable but are inferred from observable data. These variables can include categories, behavioral or mental states, or data structures. For instance, in a sensor fusion system for autonomous driving, a hypothetical variable could be "road friction coefficient", which indicates the level of grip between the vehicle's tires and the road surface. This variable cannot be directly measured but can be estimated based on sensor inputs, such as wheel slip, accelerometer data, or visual observations, allowing the autonomous vehicle to make informed decisions regarding traction control or braking [MUH04].

On the other hand, hidden states are meaningful factors that could theoretically be measured but are not observable due to practical limitations. In sensor fusion, they can arise from factors like inaccessible data, limitations in measurement techniques, or inherent system uncertainties. For instance, in indoor localization, the exact position of a pedestrian might be a hidden variable due to the limitations of the sensor resolution or environmental obstructions. Although the position cannot be directly observed, it can be inferred through fusion of multiple sensor inputs, such as radar, LiDAR, and camera data, to enhance situational awareness and tracking accuracy.

Putting the above in the context of the sensor fusion categories of data-, feature-, and decision-level, we can conclude that especially the more abstract concept of decision-level fusion has latent states. In contrast, systems categorized as data-level and feature-level may also have a directly observable state. In a autonomous setting, i.e. with i.i.d. observations and no further non-probabilistic dependencies, a latent state of a time-discrete Markov model with measurable state space leads to the so-called *hidden Markov model* (HMM), while an observable state leads to the well-known Markov chain [Gag17]. Based on our previous conclusion that the indoor localization problem belongs to the decision-level and the introduced concepts for estimating probability density functions, the HMM is naturally chosen as the most suitable option for modeling the system dynamics. Despite its name, the HMM is capable of making inferences about both hidden and hypothetical states. For this reason, we will only use the term hidden state, since both can be used interchangeably in this context.

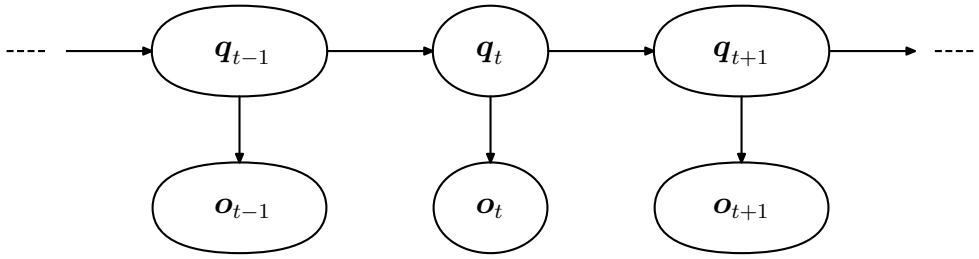


Figure 2.2: Bayesian network graph of a first-order *hidden Markov model* (HMM). The arrows illustrate the conditional dependencies inherit by the state evaluation (vertical) and transition (horizontal). A state $\boldsymbol{q}_t$ depends only on the previous state $\boldsymbol{q}_{t-1}$ and an observation $\boldsymbol{o}_t$ depends only on $\boldsymbol{q}_t$, satisfying the Markov property.

The HMM is completely described by:

- A discrete time interval $1{:}\,T = [1, T] = \{t \in \mathbb{N} \mid 1 \leq t \leq T\}$.
- A finite set of states in a measurable state space; $\boldsymbol{q}_{1:T} = \{\boldsymbol{q}_1, ..., \boldsymbol{q}_T\} \in \mathcal{Q}$, giving the non-observable (hidden) Markov process and satisfying the Markov property.
- A finite set of i.i.d. random variables $\boldsymbol{O}_1, ..., \boldsymbol{O}_T$ with corresponding realized values (observations) $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_T\}$.
- An initial (a priori) probability distribution of states $p(\boldsymbol{q}_1)$ at time $t = 1$
- State transition probabilities $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ (cf. (2.26)), often written in a matrix of the form $A = \{a_{i,j} \mid a_{i,j} = p(\boldsymbol{q}_t = j \mid \boldsymbol{q}_{t-1} = i)\}$.
- State evaluation probabilities $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ (cf. (2.27)), which define the model outputs, i.e. they provide the probability of a given observation under the conditions of a state.

If observations had a symbolic (scalar) form like $\boldsymbol{o}_{1:T} = \{o_1, ..., o_T\}$, the above continuous probability distributions of the HMM can be described in terms of discrete ones; $p(\dots) \mapsto P(\dots)$ [Fin14]. In Figure 2.2, the general architecture of a first-order HMM is shown as a graph, also called Bayesian network. The arrows illustrate the conditional dependencies inherit by the state evaluation and transition. The Markov property is easy to verify because each state $\boldsymbol{q}_t$ depends only on the previous state $\boldsymbol{q}_{t-1}$. Similarly, the value of the observation depends only on the value of the corresponding state, both at time t. Thus, the time-sequential process is defined by the state transition between two successive points in time $t - 1$ and t.

Putting the above properties in context of a dynamic system, the joint distribution of state sequences and observations can be factorized as follows:

$$p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T}) = p(\boldsymbol{q}_1)p(\boldsymbol{o}_1 \mid \boldsymbol{q}_1) \prod_{t=2}^{T} p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) \ . \tag{2.28}$$

This would be sufficient as a system function $f(\boldsymbol{o}) \mapsto \boldsymbol{q} \in \mathcal{Q}$. However, it does not involve Bayesian principles. Therefore, the state estimation relies on MLE rather then MAP. Consequently, there is no consideration of prior knowledge at each time step t, despite having an initial probability density $p(\boldsymbol{q}_1)$. This behavior is specific to Bayes' statistics, as shown in equations (2.9) and (2.17). In time-discrete dynamic contexts, this leads to recursive behavior, which gives rise to the class of Bayes filters that are ubiquitous in sensor fusion.

2.2.2 Recursive Bayesian Estimation

Given all this, we are finally able to derive the marginalization of the joint posterior described in (2.25). The basic idea is to combine the Bayesian principles of the MAP estimator (a priori and likelihood) with the HMM as a dynamical system (transition and evaluation). The derivation starts again with Bayes' rule applied to the marginal posterior distribution $p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t})$ as given by (2.16). We can write with respect to all previous observations $\boldsymbol{o}_{1:t-1}$

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) = \frac{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{o}_{1:t-1})p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1})}{p(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1})} \ . \tag{2.29}$$

The HMM always satisfies the Markov property, and thus no past measurement would provide additional information or influence the stochastic process of future steps. This allows to further simplify the above using the principles of conditional independence as follows:

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) = \frac{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1})}{p(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1})} \ , \tag{2.30}$$

where by applying the law of total probability, we get the predictive and recursive behavior out of

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1}) = \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1})d\boldsymbol{q}_{t-1} \ . \tag{2.31}$$

In the literature, (2.30) is often referred to as the update step and (2.31) as the prediction step. Due to the recursive behavior inherent in MAP principles, the above is known as recursive Bayesian estimation or Bayes filter. The most general version of the Bayes filter is represented by Algorithm 1 in pseudo-algorithmic form. The a priori distribution $p(\boldsymbol{q}_1)$ is given as input.

Algorithm 1 Bayes Filter

Input: A Priori $p(\boldsymbol{q}_1)$

1: **for all** $\boldsymbol{q}_t$ **do**
2: $p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1}) = \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})\, p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1}) d\boldsymbol{q}_{t-1}$ ▷ (2.31)
3: $p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) = \eta\, p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)\, p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1})$ ▷ (2.30)
4: **end for**

The prediction step is performed separately in line 2, followed by the update step in line 3, which calculates the posterior. The denominator of (2.30) is represented as normalization constant η.

The joint posterior $p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T})$ (cf. (2.24)) then satisfies the recursion

$$p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T}) = p(\boldsymbol{q}_{1:T-1} \mid \boldsymbol{o}_{1:T-1}) \frac{p(\boldsymbol{q}_T \mid \boldsymbol{q}_{T-1}) p(\boldsymbol{o}_T \mid \boldsymbol{q}_T)}{p(\boldsymbol{o}_T \mid \boldsymbol{o}_{1:T-1})} \ . \tag{2.32}$$

By integrating out $\boldsymbol{q}_{1:T-1}$ with $t = T$ in the above equation, the marginal posterior from (2.30) also follows as shown by [DJ09]. If (2.32) and thus (2.30) can be computed sequentially, the marginal distribution $p(\boldsymbol{o}_{1:t})$, which we also introduced as model evidence, can be recursively given by

$$p(\boldsymbol{o}_{1:T}) = p(\boldsymbol{o}_1) \prod_{t=2}^{T} p(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1}) \ . \tag{2.33}$$

where

$$p(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1}) = \int p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1}) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) d\boldsymbol{q}_{t-1:t} \ . \tag{2.34}$$

As discussed in the derivation of MAP (cf. (2.17)), the model evidence is often neglected and/or interpreted as a normalization constant, as we have done in Algorithm 1 [Sä13]. In practice, it is often sufficient to normalize the final result of a filter update to 1 instead of computing the integral [DJ09; Fet23a; TBF05].

For a more detailed discussion, we split the Bayes filter in (2.30) into three blocks: evaluation, transition, and recursion, instead of the previously mentioned update and prediction. Neglecting the model evidence and assuming a normalization step results in the following:

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) \propto \underbrace{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)}_{\text{evaluation}} \int \underbrace{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})}_{\text{transition}} \underbrace{p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1}) d\boldsymbol{q}_{t-1}}_{\text{recursion}} \ . \tag{2.35}$$

In the context of sensor fusion, this notation is very important because it describes the necessary models to make inferences about $\boldsymbol{q}_t$ without requiring a specific implementation. We call them models because the required probability density function is often described by modeling the uncertainty of a sensor (e.g., radar data) or the dynamics of the system (e.g., human motion).

What is not so obvious after looking at (2.35) is the need for an a priori distribution $p(\boldsymbol{q}_1)$, which is still relevant as shown in (2.32) or Algorithm 1. In the following we will discuss these models in more detail.

The *prior model*, denoted by the a priori distribution $p(\boldsymbol{q}_1)$, represents initial beliefs about an uncertain state $\boldsymbol{q}$ before any observations $\boldsymbol{o}$ are made. It characterizes the possible and impossible values that the parameters of the state can take, as well as their relative probabilities in the absence of any evidence. Selecting a prior is a critical step in Bayesian filtering, as it plays a fundamental role in incorporating existing knowledge or assumptions about the system [Jay68]. The choice of an appropriate $p(\boldsymbol{q}_1)$ depends on several factors, including the available information, the specific problem domain, and the desired behavior of the analysis. It is therefore an essential component of Bayesian inference and a critical factor in improving the accuracy of the filtering process.

Various approaches to model the a priori distribution exist. A profound overview can be found in [BS09] or [Gel13]. We have already discussed the class of conjugate priors in Section 2.1.2, which are more of an algebraic convenience that a closed-form expression for the posterior exists. However, this is only true for simple posteriors in the exponential family, such as the Gaussian, Poisson, or Gamma distributions [Fin97].

Another commonly used type is an informative prior, which incorporates specific and known information about the system. For example in meteorology, an informative prior can be constructed based on historical weather patterns or climate models, such as the knowledge that a certain region is prone to experiencing heavy rainfall during a particular season. Here, it can be designed to reflect this information and would bias the forecast towards higher probabilities of rain during that season. Considering indoor localization, an informative prior could be a known starting position at the entrance of a building. One way to model this is to use a Gaussian distribution, where the starting position serves as the expected value and the width of the entrance determines the variance [Ebn22]. In a more general sensor fusion setting, a prior can be used to encode sensor characteristics or known relationships between sensor measurements. This helps to improve the accuracy and reliability of the fusion process [Fet23a].

In contrast, uninformative priors are used when there is a lack of specific knowledge about the system. These priors express general information without making strong assumptions. A classic example is the uniform distribution, which assigns equal probability to all possible parameter values. Another well-known uninformative prior in Bayesian statistics is Jeffreys' prior [Jef46]. Its key idea is to choose a prior that is invariant under reparameterization, i.e., the prior distribution remains unchanged when the state is expressed in a different parameterization or scale. This property ensures that the choice of the prior does not depend on the specific state used and avoids introducing bias due to the choice of parameter representation. However, for

complex applications such as sensor fusion, Jeffreys' prior is not always available in closed form and may require numerical approximation methods to compute. In addition, since it does not incorporate much knowledge about the state and thus minimizes the amount of a priori influence on the posterior distribution, other prior models may be more appropriate when a priori knowledge is available [Gel13].

The *transition model* denotes the probability density $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ that describes the system dynamics and its uncertainties as a Markov sequence, defined in terms of the transition from $\boldsymbol{q}_{t-1}$ to $\boldsymbol{q}_t$. Depending on the context, it is often referred to as a dynamic or predictive model because it makes statements about the future of the system. For example, the authors of [WLG18] implement a transition model to predict the shape α and scale β parameters of a Gamma distribution that models limit order books in financial markets. The new state $\boldsymbol{q}_t = (\alpha, \beta)$ is drawn from another Gamma distribution with the previous state $\boldsymbol{q}_{t-1}$ as the expected value and a data-driven a priori standard deviation. For applications based on physical laws, such as calculating the trajectory of a missile, a physical model can be used for prediction. For pedestrian tracking, the possible movement patterns of a human, such as walking direction, speed, or step length, must be taken into account. Since the number of degrees of freedom and influencing variables for both a missile and a human cannot be exactly determined, approximate physical models are used. The exciting thing about these models is that they can represent not only pure physical motion, but also external influences such as obstacles that are not easily defined in state space. For example, in Section 4 we will introduced several concepts of modeling the human movement inside buildings, incorporating knowledge about walls and other objects.

The *evaluation model* is defined as a probability density function $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$. We first encountered it in (2.1) as a statistical model that draws observations from a population defined by unknown model parameters $\boldsymbol{\theta} \sim \boldsymbol{q}$. In the context of MLE, it was reinterpreted as a likelihood function in (2.8), representing the observation as a function of the state. In a Markovian setting, it then becomes the so-called evaluation model, keeping most of its already introduced properties. When considering sensor fusion, the evaluation holds the sensor models and is thus largely responsible for the fusion process. As discussed in the derivation of MAP (cf. Section 2.1.2) as well as in the introduction of estimation theory (cf. Section 2.1), many real-world applications require the assumption of statistical independence between the respective sensor models. This allows to omit costly correlation calculations and greatly simplifies the entire fusion process to a joint distribution of the form

$$p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) = \prod_{i=1}^{N} p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)_i \tag{2.36}$$

using the concepts of (2.2) as well as (2.28) in a marginalized form and applying the multiplication rule for independent events. Here N denotes the number of sensor models involved and i their identifying index. Each $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)_i$ can be chosen very individually. The only requirement is that it has a probabilistic representation. Since we interpret Bayes' theorem as a definition rather than an application of the theorem, it does not even have to be a proper distribution (see the discussion below (2.17)). The only thing that needs to be strictly addressed is to perform a normalization step that ensures that the entire evaluation model in (2.36) sums to 1 [BSLK01].

Of course, ignoring the correlation between respective sensor models is a simplification that comes with a cost. According to [BSC86], sensor models that estimate the same target (state) have a dependent error due to a common process noise. For example, if sensors are exposed to the same external noise that biases their measurement, and these dependencies are not accounted for, the estimate may suffer from over/under confidence and even a divergence in the fusion process, e.g. multi-modalities in the posterior [Kha13b]. To solve such a problem, often referred to as data incest, there are several solution strategies. One way is to eliminate the correlation directly at the measurement level (before fusion). This can be done in a straightforward way by eliminating an affected measurement, or more implicitly by reconstructing the measurements using a decorrelation process [MEK05]. Another strategy, especially in the presence of an unknown correlation, is to account for the correlation in the fusion process. Several different approaches are presented in the literature, such as *covariance intersection* (CI) [JU97] or the *internal ellipsoid approximation* (IEA) [ZL08] in a Gaussian setting and *generalized Chernoff fusion* (GCF) [Hur02] for arbitrary probability density functions. The latter is especially promising, as it is also able to consider an unbounded number of sensor models.

Note that statistical independence is assumed not only between the individual sensor models, but also between the evaluation and the transition. The Markov assumption only ensures that the noise of the transition and the evaluation are independent over time, but not between $\boldsymbol{q}_t \leftrightarrow \boldsymbol{o}_t$ and $\boldsymbol{q}_{t-1} \leftrightarrow \boldsymbol{o}_t$ (cf. Figure 2.2). This dependence is best explained by an example. A moving object is to be tracked by a radar. Even if the sensor error of the radar is completely independent of the motion of the object, the transition in (2.16) causes an additional noise contribution to the evaluation just because of this motion. The papers [SG12] and [GS10] deal with this phenomenon in detail, which is especially important for nonlinear and non-Gaussian problems. If all sensor models i are based on the same parametric distribution, e.g. a Gaussian, and the state space is linear, the correlated noise can be handled. The most famous implementation of (2.35), which applies this by assuming a Gaussian distribution of all sensor noise, is the Kalman filter [Kal60]. However, even though the correlation can be accounted for by maintaining cross-coverage between updates, it is often omitted because it has been shown to scale quadratically with the number of updates [Kha13b].

As shown in Section 2.1.1 and Section 2.1.2 finding a closed-form solution for the posterior or even the likelihood (evaluation) is not to be expected in practices. Especially in applications with a high-dimensional state space, the problem is exacerbated, which is why it is often referred to as the curse of dimensionality [DJ09]. Thus, implementing Algorithm 1 requires finding a numerical solution for the integration in line 2 and the product in line 3. The Kalman filter mentioned above is one such numerical solution approach. In its original form, it makes two basic assumptions: that the dynamical system is linear and that the noise is Gaussian distributed. If the real system has exactly these properties, i.e., the evaluation and transition model reality accurately, and the covariance of the noise is known exactly, the Kalman filter provides an optimal state estimate in the sense of MMSE [BSLK01].

Obviously, these assumptions of linear and Gaussian distributed state estimation are often too limiting for many real-world problems. A very simple example is the mathematical pendulum, which has a nonlinear equation of motion (transition) because it contains trigonometric functions. Considering the complexity of human locomotion in three-dimensional space, alternative methods such as the *extended Kalman filter* (EKF) or the *unscented Kalman filter* (UKF) are required [JU04]. The former linearizes the nonlinear system equations around the current estimate and then applies the Kalman filter to the linear model. However, this linearization can quickly lead to inaccuracies, as it is only reliable for systems that are almost linear on the time scale of the updates [TBF05]. The UKF, on the other hand, approximates the probability distribution of the state by a set of points (sigma points) and propagates them through the nonlinear equations. This results in a better approximation of the nonlinear dynamics than the EKF and makes it a promising approach.

Although the UKF is able to map nonlinearities directly, it is still subject to the Gaussian assumption. As a consequence, like all other generalizations and extensions of the Kalman filter class, it is not able to map multimodal densities, i.e. it is not possible to build a multi-target tracking system with a single UKF. Finally, the Kalman filter has low robustness to outliers, since they introduce a significant bias that is always incorporated into the state estimation.

For a generally defined sensor fusion system like the one developed in this thesis, the Gaussian assumption is thus a too strong constraint. It may often be sufficient for many physical sensor noises, but fails for more complex models, especially virtual sensor models such as the activity detection presented in Section 3.5. Allowing arbitrary probability densities to be represented satisfies our desire to be able to combine arbitrary probabilistic sensor models. While this can lead to a cumbersome and complex mixed density for the posterior, especially since multimodal densities may not be desirable for state estimation unless multi-target tracking is involved, it allows for a variety of interesting options for analyzing the state space covered by

the posterior. One class of methods that deal with nonlinear systems and provide a numerical solution for almost arbitrary probability densities are the broad class of Monte Carlo methods.

2.3 Sequential Monte Carlo

Essentially, we have so far been introduced to two classes of problems in statistical inference that require a numerical solution. One is optimization problems, usually associated with the likelihood approach, and the other is integration problems, associated with Bayesian approaches. This is not a strict classification, as we can see by comparing the examples in Section 2.1, but they are commonly used in the literature because of the specific tasks they are designed to solve [RC13]. Sampling, i.e., the generation of random samples from probability densities that cannot be drawn directly, is a third class of problems we will encounter in the course of this chapter.

Although all three classes of problems have their own algorithms and mathematical principles, they can be solved numerically using the same basic idea: repeated random sampling from a known probability density. Despite its simplistic nature, it allows to provide a solution to any problem that has a probabilistic representation. The group of numerical algorithms using this concept is generally referred to as *Monte Carlo* (MC) methods. The starting point, called the target distribution $\tau(\boldsymbol{q})$, for the approximation is usually a complex, high-dimensional and/or intractable probability density, an optimization problem, or an integral. In a very generic sense it can be written as

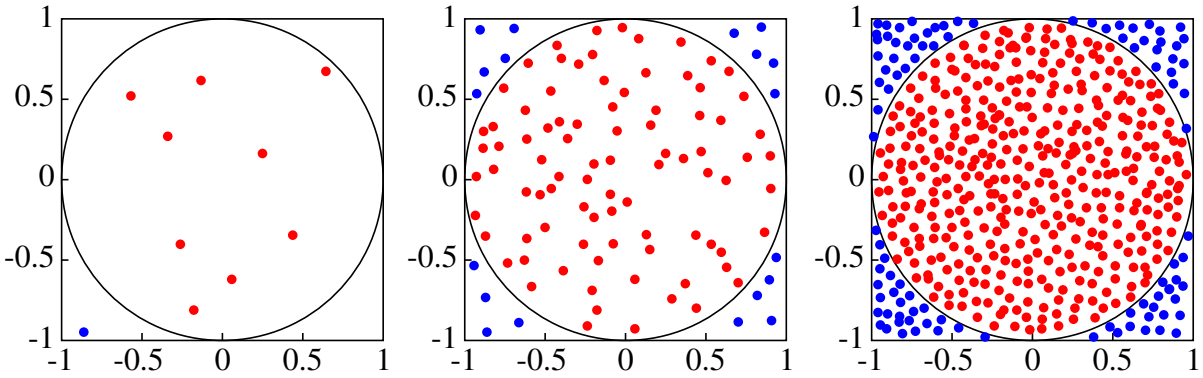
$$\tau(\boldsymbol{q}) = \frac{\gamma(\boldsymbol{q})}{Z} \ , \tag{2.37}$$

where the target $\tau(\boldsymbol{q})$ is defined on the state space $\mathcal{Q}$ and $\gamma(\boldsymbol{q}) : \mathcal{Q} \to \mathbb{R}^+$ is required to be known point-wise. Thus, the normalizing constant $Z = \int \gamma(\boldsymbol{q}) \ d\boldsymbol{q}$ might be unknown. MC methods try to approximate $\tau(\boldsymbol{q})$ and estimate Z by sampling N i.i.d. random variables, $\boldsymbol{X}^i \sim \tau(\boldsymbol{q})$ for $i = 1, \ldots, N$ by the empirical measure

$$\tau(\boldsymbol{q}) \approx \frac{1}{N} \sum_{N}^{i=1} \delta_{\boldsymbol{X}^i}(\boldsymbol{q}) \ , \tag{2.38}$$

where $\delta_{x_0}(x)$ denotes the Dirac delta mass located at x_0. For any test function $\varphi : \mathcal{Q} \to \mathbb{R}$ the expectation is thus denoted by

$$\mathrm{E}^{\mathrm{MC}}(\varphi) := \int \varphi(\boldsymbol{q})\tau(\boldsymbol{q})d\boldsymbol{q} \approx \frac{1}{N} \sum_{i=1}^{N} \varphi(\boldsymbol{X}^i) \ . \tag{2.39}$$



(a) $N = 10, \pi \approx 3.6$ (b) $N = 100, \pi \approx 3.24$ (c) $N = 500, \pi \approx 3.2$

Figure 2.3: A simple example of how the *Monte Carlo* (MC) method can be used to approximate π. It is determined by quadrupling the probability that a i.i.d. random sample within the square falls inside the circle. Due to the law of large numbers, the variance of the result decreases as the number of samples increases.

In essence, MC exploits the law of large numbers, i.e. as the number of samples increases, the empirical average converges to the expected value. Therefore, the variance of the approximation error decreases at a rate of $\mathcal{O}(1/N)$ regardless of the dimension of the space $\mathcal{Q}$ [DJ09].

As a simple example, consider a unit square of side length 2, centered at the Cartesian origin $(0, 0)$, and a circle of radius 1, also centered at the origin, such that $\boldsymbol{q} = (x, y)$ and $\mathcal{Q} = \{x, y \in [-1, 1]\}$. The target distribution $\tau(\boldsymbol{q}) = \pi \approx \hat{\pi}$ and a sample $\boldsymbol{X}$ is drawn according to a uniform distribution $\mathcal{U}(-1, 1)$ for each dimension of $\boldsymbol{q}$. The MC method estimates the value of π by randomly generating samples within the square and counting the number M of points that fall within the circle according to:

$$\pi \approx \hat{\pi} = 4 \cdot \frac{M}{N} \quad . \tag{2.40}$$

Here, the ratio of M to N approximates the ratio of the areas, which is $\frac{\pi}{4}$ since the area of the square is 4 and the area of the circle is π. The value M can be easily determined using Pythagorean theorem, resulting in a test function

$$\varphi(\boldsymbol{q}) = 4 \cdot \begin{cases} 1, & x^2 + y^2 \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad , \tag{2.41}$$

which by substitution in (2.39) then gives (2.40). As can be seen in Figure 2.3, as the number of samples increases and the process is repeated, the approximation becomes more accurate because the law of large numbers ensures convergence.

Algorithm 2 Markov chain Monte Carlo - Metropolis-Hastings

Input: A Priori $\boldsymbol{X}_0 \in \mathcal{Q}$

1: **for** $i = 1$ **to** N **do**
2: Sample $\boldsymbol{X}_i \sim q(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1})$ ▷ Proposal Distribution
3: Compute $P_{\text{accept}} = \dfrac{p(\boldsymbol{q}_i \mid \boldsymbol{o}_{1:n})}{p(\boldsymbol{q}_{i-1} \mid \boldsymbol{o}_{1:n})} = \dfrac{p(\boldsymbol{o}_{1:n} \mid \boldsymbol{X}_i)p(\boldsymbol{X}_i)}{p(\boldsymbol{o}_{1:n} \mid \boldsymbol{X}_{i-1})p(\boldsymbol{X}_{i-1})}$ ▷ Acceptance Ratio
4: Sample $u \sim \mathcal{U}(0, 1)$
5: **if** $u > P_{\text{accept}}$ **then**
6: $\boldsymbol{X}_i = \boldsymbol{X}_{i-1}$ ▷ Reject Sample
7: **end if**
8: Write $\boldsymbol{X}_i$ in file ▷ Approximation
9: **end for**

In practice, however, a summation according to (2.38) is often not efficiently feasible because the state space is too large [And03]. For this reason, methods have been developed that better "fill" the space and sample the most important points more frequently, often converging to the integral more quickly. *Markov chain Monte Carlo* (MCMC) is one such class of methods and provides algorithms like Metropolis–Hastings or Gibbs sampling [RC13]. Given an *observable* state $\boldsymbol{q}$, the general procedure for a target distribution that is our posterior in question, $\pi(\boldsymbol{q}) = p(\boldsymbol{q} \mid \boldsymbol{o}_{1:n})$, is shown in Algorithm 2.

The algorithm starts with an initial sample $\boldsymbol{X}_0^0 \in \mathcal{Q}$ and then generates a new candidate $\boldsymbol{X}_i^k$ by sampling from a proposal distribution $q(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1})$ for each iteration step i. As $k = i$, we can simplify $\boldsymbol{X}_i = \boldsymbol{X}_i^k$. The proposal distribution is an important mechanic, since obviously drawing from $\tau(\boldsymbol{q})$ is often intractable. As its name suggests, it *proposes* new samples from a distribution that should be easy to sample from, have enough flexibility to explore different regions of the state space, and cover the support of the target distribution [And03]. In many cases, a Gaussian with a mean of $\boldsymbol{q}_{i-1}$ and a standard deviation of 1 meets these requirements. In line 3, the acceptance probability P_{accept} for the candidate state is computed based on the ratio of the target distribution evaluated at the candidate $\boldsymbol{X}_i$ and $\boldsymbol{X}_{i-1}$. Note that the normalization, in this case the model evidence of Bayes' theorem (see (2.16)), can be ignored, as it is canceled out in the equation. If P_{accept} is greater than a random value between 0 and 1, the candidate state is accepted and saved; otherwise it is rejected. This allows the Markov chain to explore the state space guided by the proposal distribution, while ensuring convergence to the target $\tau(\boldsymbol{q})$. The algorithm is then repeated for a sufficient number of iterations to allow the Markov chain to converge. The stored parameters provide an empirical approximation of the target distribution $\tau(\boldsymbol{q})$.

Convergence of the Markov chain is typically evaluated by examining diagnostic statistics such as autocorrelation and effective sample size. It is also important to discard an initial portion of the chain, known as the burn-in period, to ensure that the samples are drawn from the stationary distribution. However, MCMC is not well suited to sequential simulation. At a given time t, one must wait for the Markov chain with a kernel K_t to reach its stationary distribution, as it is commonly solved using parallel running MCMC algorithms [DMDJ06].

In a sequential setting, (2.37) is denoted by

$$\tau(\boldsymbol{q}_{1:t}) = \frac{\gamma(\boldsymbol{q}_{1:t})}{Z_t} \ , \tag{2.42}$$

with the approximation (2.38) and the expectation (2.39) changing accordingly. The normalizing constant changes to $Z_t = \int \gamma(\boldsymbol{q}_{1:t}) \ d\boldsymbol{q}_{1:t}$ and the samples $\boldsymbol{X}^i_{1:t} \sim \tau(\boldsymbol{q}_{1:t})$ are now indexed by i over the time interval. For ease of reading, we have chosen a simplified notation so that $\tau(\dots) = \tau_t(\dots)$. Solving this problem with MCMC is not straightforward and has several drawbacks. The assumed stationarity of the posterior may make it difficult for the MCMC to adapt and accurately track the evolving distribution, leading to suboptimal or biased estimates. Another drawback is the lack of data reuse, as traditional MCMC methods require sampling from scratch at each time step, ignoring the information gained from previous iterations. However, both of these factors are important for estimation according to the sensor fusion requirements we defined at the beginning of this chapter. An even more crucial point: As discussed in Section 2.2.1, methods relying on classic Markov chains require that $\boldsymbol{q}$ is observable. In most sensor fusion scenarios, the state can only be determined indirectly from the sensor data, which is why we introduced the HMM in the first place. This leads to the ultimate requirement that the desired approximation method using the principles of MC must be compatible with the HMM and thus the Bayes filter. In order to achieve this, a general framework for *sequential Monte Carlo* (SMC) methods will be presented in the following.

2.3.1 Sequential Importance Sampling

In most cases, the target distribution $\tau(\boldsymbol{q}_{1:t})$ is a complex high-dimensional probability distribution where direct sampling is intractable. In sensor fusion applications, the problem is exacerbated because the state $\boldsymbol{q}$ is hidden, resulting in the posterior being the estimand. To address this, an *importance density* function $q(\boldsymbol{q}_{1:t})$ is introduced such that

$$\tau(\boldsymbol{q}) > 0 \Rightarrow q(\boldsymbol{q}_{1:t}) > 0 \ ,$$

from which it is easy to draw samples. The concept is quite similar to the proposal distribution of the MCMC introduced in the previous chapter, so some authors use the names interchangeably. Thus, in theory, they are different. The importance density aims to effectively capture the dynamics and uncertainties of the underlying system, taking into account both prior knowledge and incoming observations. In contrast, the proposal distribution focuses only on the dynamics, i.e., it determines how the chain moves from the current state to a new candidate state, and thus guides the exploration of the state space to estimate properties of the target distribution. Nevertheless, for the sake of simplicity, a function of the form $q(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1})$ with a Gaussian distributed error is often assumed for both in practice. As we will show throughout this thesis, such an assumption has severe limitations, that require for more sophisticated alternatives.

Using an importance density for drawing samples $\boldsymbol{X}^i \sim q(\boldsymbol{q}_{1:t})$ is widely known as *importance sampling* (IS) technique and is a fundamental tool in MC [Sä13]. The basic idea is to weight each $\boldsymbol{X}^i$ to obtain an estimate of the desired quantities associated with the target distribution. The unnormalized weight function

$$w(\boldsymbol{q}_{1:t}) = \frac{\gamma(\boldsymbol{q}_{1:t})}{q(\boldsymbol{q}_{1:t})} \tag{2.43}$$

describes the ratio of the target $\tau(\boldsymbol{q}_{1:t})$ to the importance $q(\boldsymbol{q}_{1:t})$ given the IS identities from (2.42) as

$$\tau(\boldsymbol{q}_{1:t}) = \frac{w(\boldsymbol{q}_{1:t})q(\boldsymbol{q}_{1:t})}{Z_t} \tag{2.44}$$

$$Z_t = \int w(\boldsymbol{q}_{1:t})q(\boldsymbol{q}_{1:t})d\boldsymbol{q}_{1:t} \tag{2.45}$$

Consequently, the MC approximation is now carried out for $q(\boldsymbol{q}_{1:t})$, what is according to (2.38) the empirical measure of the samples $\boldsymbol{X}^i$. Substituting this into (2.44) and (2.45) we obtain the IS approximation for the target

$$\tau(\boldsymbol{q}_{1:t}) \approx \sum_{i=1}^{N} W_t^i \delta_{\boldsymbol{X}_{1:t}^i}(\boldsymbol{q}_{1:t}) \ , \tag{2.46}$$

and normalization factor

$$Z_t \approx \frac{1}{N} \sum_{i=1}^{N} w(\boldsymbol{X}_{1:n}^i) \ . \tag{2.47}$$

Here

$$W_t^i = \frac{w(\boldsymbol{X}_{1:t}^i)}{\sum_{j=1}^{N} w(\boldsymbol{X}_{1:t}^j)} \tag{2.48}$$

denotes the normalized weight of the corresponding sample, which can be written as a tuple $\{W_t^i, \boldsymbol{X}_t^i\}$, commonly known as particle. Consequently, a set of particles is completely described by $\{W_{1:t}^i, \boldsymbol{X}_{1:t}^i\}_{i=1}^N$. For a test function $\varphi : \mathcal{Q} \to \mathbb{R}$ the expectation is simply denoted as

$$\mathrm{E}^{\mathrm{IS}}(\varphi) := \int \varphi(\boldsymbol{q})\tau(\boldsymbol{q})d\boldsymbol{q}_{1:t} \approx \sum_{i=1}^{N} W_t^i \varphi(\boldsymbol{X}_{1:t}^i) \ . \tag{2.49}$$

In contrast to (2.39), this estimate is biased for a finite number of particles, as we sample from the importance density instead of the target. As proven by several authors, $\mathrm{E}^{\mathrm{MC}}(\varphi)$ is a asymptotically consistent estimator that satisfies the central limit theorem with the bias and the variance both being $\mathcal{O}(1/N)$ [BSLK01; DJ09].

In a nutshell, IS uses a finite number of particles $\{W_{1:t}^i, \boldsymbol{X}_{1:t}^i\}_{i=1}^N$ to approximate a probability distribution, where each particle is a weighted representation of a possible system state $\boldsymbol{q}$. This is a fundamental concept in numerical approximation and takes an important role throughout this work. The advantages of using weights are quite obvious, since with a classical MC approach, as illustrated in Figure 2.3, the target $\tau(\boldsymbol{q})$ is approximated based on the density of samples in a particle region of the state space. For large state spaces, the number of samples required often becomes intractable. This is known as the curse of dimensionality, where the sample size required to achieve a given level of accuracy grows exponentially with the dimensionality of the state space. In contrast, IS allows for a more efficient allocation of computational resources. This approach helps mitigate the exponential growth in the number of samples required, making the estimation process more feasible in large state spaces.

If we would use the above sampling scheme in a sequential setting, the computational complexity is at least linear in size of time steps t, i.e. linearly increasing with each t. This may not sound scary at first, but consider the simple example of approximating π. In Figure 2.3c we get a value of 3.2 with a sample size of $N = 500$, whereas to get a numerically stable value of 3.14 we need at least a million samples. This was the reason we considered MCMC in the first place. For IS, we tackle the complexity problem by considering an importance distribution that samples recursively with increasing time t, i.e. $\boldsymbol{X}_1^i \sim q(\boldsymbol{q}_1)$ at time 1 and then $\boldsymbol{X}_k^i \sim q(\boldsymbol{q}_k \mid \boldsymbol{X}_{1:k-1}^i)$ for time $k = 2, \ldots, t$. As a result, the importance distribution is then structured as follows

$$\begin{aligned} q(\boldsymbol{q}_{1:t}) &= q(\boldsymbol{q}_1) \prod_{k=2}^{t} q(\boldsymbol{q}_k \mid \boldsymbol{q}_{1:k-1}) \\ &= q_{t-1}(\boldsymbol{q}_{1:t-1}) q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1}) \ , \end{aligned} \tag{2.50}$$

Algorithm 3 Sequential Importance Sampling

Input: A Priori $q(\boldsymbol{q}_1)$

1: **for** $i = 1$ **to** N **do**
2: **if** time $t = 1$ **then** ▷ Initialization
3: Sample $\boldsymbol{X}_1^i \sim q(\boldsymbol{q}_1)$
4: Compute the weights $W_1^i \propto w_1(\boldsymbol{X}_1^i)$
5: **else if** time $t \geq 2$ **then** ▷ Recursion
6: Sample $\boldsymbol{X}_t^i \sim q(\boldsymbol{q}_t \mid \boldsymbol{X}_{1:t-1}^i)$
7: Compute the weights $W_t^i \propto w_{t-1}(\boldsymbol{X}_{1:t-1}^i) \cdot \alpha(\boldsymbol{X}_{1:t}^i)$
8: **end if**
9: **end for**

where q_{t-1} denotes the previous distribution in the recursion. The associated unnormalized weight function of (2.43) is then recursively computed according to the decomposition

$$\begin{aligned} w_1(\boldsymbol{q}_{1:t}) &= w(\boldsymbol{q}_1) \prod_{k=2}^{t} \alpha(\boldsymbol{q}_{1:k}) \\ &= w_{t-1}(\boldsymbol{q}_{1:t-1}) \cdot \alpha(\boldsymbol{q}_{1:t}) \ , \end{aligned} \tag{2.51}$$

where $\alpha(\boldsymbol{q}_{1:t})$ is the *incremental importance weight* function given by

$$\alpha(\boldsymbol{q}_{1:t}) = \frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})\, q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})} \ . \tag{2.52}$$

It describes how particles are weighted over time and will thus be of major interest for concrete implementations of the so called *sequential importance sampling* (SIS).

The SIS is given in Algorithm 3 in pseudo-algorithmic form describing the recursive procedure for a single time step t. In the initialization from line 2 to 4 particles are sampled according to the a priori distribution $q(\boldsymbol{q}_1)$ and weighted based on (2.51). For all further time steps, the recursive procedure starting from line 5 is carried out for each particle index by $i = 1, \ldots, N$. An estimation for $\tau(\boldsymbol{q}_{1:t})$ is then obtained at any time step using (2.46) and for Z_t using (2.47).

At first glance, SIS provides a good solution to handle the computational complexity associated with MC methods. This is especially true if the time to compute $\alpha(\boldsymbol{q}_{1:t})$ and to sample from the importance $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})$ is independent of t, which happens if q is reasonably chosen [Arn01]. In practice, however, this is often not the case, and SIS suffers from several problems that need to be addressed [Orh12]. The most crucial problem is a phenomenon known as *weight degeneracy*, which is caused by the recursive behavior of (2.50) and (2.51). It occurs when a small number of particles have significantly higher weights, while the remaining particles

have very low weights. In essence, the majority of particles have negligible influence while only a few contribute to the estimate. The consequence of particle degeneracy is that the target distribution is poorly represented and overly dependent on the few particles with high weights. Consequently, this leads to high variance as the degenerate particles fail to explore the full range of the state space, resulting in a loss of diversity in the overall target approximation.

The authors of [DJ09] and [Orh12] demonstrate this using a simple toy example. Lets consider $q \in \mathcal{Q} = \mathbb{R}$ and a target distribution of the form

$$\begin{aligned} \tau(\boldsymbol{q}_{1:t}) &= \prod_{k=1}^{t} \tau(q_k) = \prod_{k=1}^{t} \mathcal{N}(q_k \mid 0, 1) \\ \gamma(\boldsymbol{q}_{1:t}) &= \prod_{k=1}^{t} e^{-\frac{1}{2}q_k^2} \\ Z_t &= (2\pi)^{\frac{t}{2}} \ . \end{aligned} \tag{2.53}$$

For the approximation using SIS, we chose a similar importance distribution of the form

$$q(\boldsymbol{q}_{1:t}) = \prod_{k=1}^{t} q(q_k) = \prod_{k=1}^{t} \mathcal{N}(q_k \mid 0, \sigma^2) \ . \tag{2.54}$$

Lets consider we want to estimate the normalizing constant, we have a variance $\mathrm{Var}(\hat{Z}_t) < \infty$ only for $\sigma^2 > \frac{1}{2}$, where $\hat{Z}_t$ is the approximation of Z_t according to (2.47). By investigating the relative variance

$$\frac{\mathrm{Var}(\hat{Z}_t)}{Z_t^2} = \frac{1}{N}\left(\int \frac{\tau^2(\boldsymbol{q}_{1:t})}{q(\boldsymbol{q}_{1:t})} d\boldsymbol{q}_{1:t} - 1\right) = \frac{1}{N}\left[\left(\frac{\sigma^4}{2\sigma^2 - 1}\right)^{t/2}\right] \tag{2.55}$$

it can be seen that even in such a simple case, the variance grows exponentially with t as $\frac{\sigma^4}{2\sigma^2-1} > 1$ for any $\frac{1}{2} < \sigma^2 \neq 1$. Depending on the size of t, this can quickly become a critical problem. For example, choosing $\sigma^2 = 1.2$ seems like a reasonable choice for the importance distribution, since $q(\boldsymbol{q}_{1:t}) \approx \tau(\boldsymbol{q}_{1:t})$. If we now plug this into (2.55), we get something of the form

$$N\frac{\mathrm{Var}(\hat{Z}_t)}{Z_t^2} \approx (1.103)^{t/2} \ . \tag{2.56}$$

For $t = 1000$ this is about $1.9 \cdot 10^{21}$, which would require the impractical amount of $N \approx 2 \cdot 10^{23}$ particles to get a relative variance of $\frac{\mathrm{Var}(\hat{Z}_t)}{Z_t^2} = 0.01$. This shows that as time increases, the number of particles must also increase to ensure a reasonable variance, otherwise the risk of

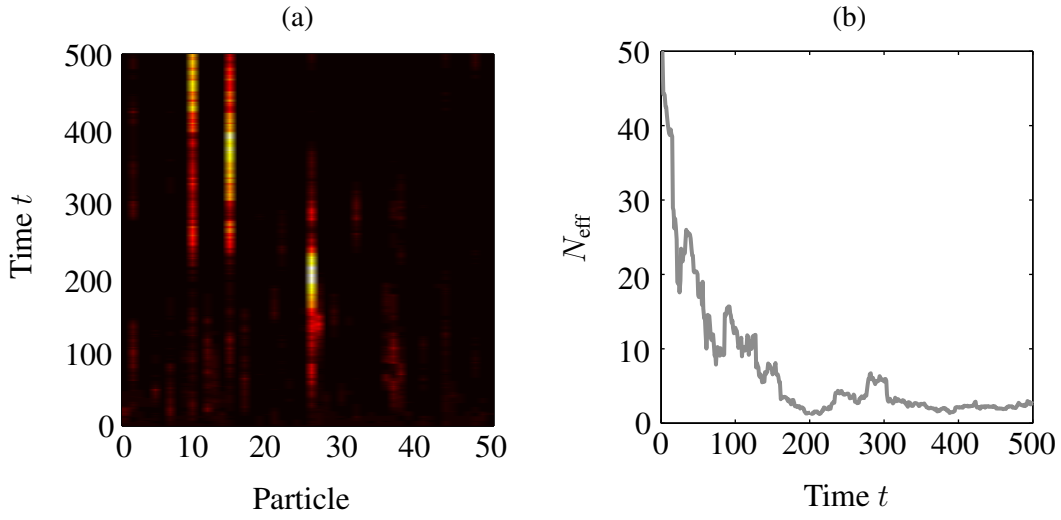


Figure 2.4: Simulation of the example presented in (2.53) and (2.54) using $N = 50$ particles for a duration of 500 time steps. In (a) the weights of all particles at each time step t shown, with brighter colors indicating larger weights. In (b) the effective sample size N_{eff} is plotted as a function of t. The figures are based on [Orh12].

weight degeneracy increases significantly. Conversely, one could say that a high variance is basically an indicator of the occurrence of weight degeneracy.

A way to measure for degeneracy is the effective sample size $N_{\text{eff}} \in \mathbb{R} : 1 \leq N_{\text{eff}} \leq N$ given as

$$N_{\text{eff}} = \frac{N}{1 + \text{Var}(\tau(\boldsymbol{q}_{1:t}))} \approx \frac{1}{\sum_{i=1}^{N} (W_t^i)^2} \ , \tag{2.57}$$

where in practice the true variance $\text{Var}(\tau(\boldsymbol{q}_{1:t}))$ is obviously unknown and thus approximated. A large N_{eff} indicates a more diverse and representative particle set, with less degeneracy. Thus, a value close to N means that the particle set is well distributed and the weights are relatively balanced. On the other hand, if N_{eff} is much smaller than N, it indicates that the majority of the particles have negligible weights and do not contribute significantly to the estimation. Figure 2.4 illustrates the effective sample size for the above toy example with occurring degeneracy using $N = 50$ particles for a duration of 500 time steps. The weights of particles at each time step t are visualized in Figure 2.4a, with brighter colors indicating larger weights. Looking at the effective sample size plotted as a function of t in Figure 2.4b shows that as time goes on N_{eff} quickly drops from 50 to values less than 5. In conclusion, although SIS is a very promising approach, additional methods need to be integrated to stabilize the variance with time, especially in real-time scenarios with unknown duration.

2.3.2 Resampling

The de facto standard for (partially) solving the problem of weight degeneracy and the associated increase in variance are so-called *resampling* techniques. In IS / SIS, the target distribution $\tau(\boldsymbol{q}_{1:t})$ is approximated by weighted samples $\Upsilon_t = \{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$ from the importance distribution $q(\boldsymbol{q}_{1:t})$, since direct sampling is not feasible. The process, as seen in Algorithm 3, is split in two: first sampling from $q(\boldsymbol{q}_{1:t})$ and then weighting, i.e. only after assigning weights, the particle set approximates $\tau(\boldsymbol{q}_{1:t})$. The problem here is that the particles are still distributed based on the importance distribution, and thus only the weights represent the actual approximation $\hat{\tau}(\boldsymbol{q}_{1:t}) \approx \tau(\boldsymbol{q}_{1:t})$ of the target (cf. (2.46)). The idea of resampling is to generate samples that are distributed according to $\hat{\tau}(\boldsymbol{q}_{1:t})$ and thus represent the target more accurately. This reduces the number of extremely low weighted particles and thus the variance of the estimate. To do this, an additional sampling step is added at the end - hence the name resampling - which draws a new set of particles $\dot{\Upsilon}_t$ from the current set of particles Υ_t based on the weights W^i_t. Basically, you could say that the positions of the particles in the state space are updated based on the approximation $\hat{\tau}(\boldsymbol{q}_{1:t})$. In addition, the newly obtained particle set $\dot{\Upsilon}_t$ also improves the exploration of the state space at $t + 1$.

Thus, in most practical applications, resampling is an unavoidable step, which is why a variety of approaches have been proposed in the literature over the years [CCF99; Fox01; HR19; HSG06; Kit96; LCL01; LSS12]. The authors of [LBD15] provide a very good overview of different resampling schemes. At a high level, they distinguish between sequential and parallel resampling, the latter representing two or more sequential implementations running simultaneously. The sequential schemes are further broken down into single-distribution, compound, and special strategies. The four most popular resampling algorithm in literature, also referred to as traditional methods, are illustrated in Figure 2.5d. They are all based on a single-distribution scheme, where we are interested in obtaining N new samples from the approximation $\hat{\tau}(\boldsymbol{q}_{1:t})$ and thus Υ_t. According to [DJ09] this can be interpreted as associating a number of offspring $N^i_t \in \mathbb{N}_0$ with each sample $\boldsymbol{X}^i_{1:t}$ such that $N^{1:N}_t = (N^1_t, \ldots, N^N_t)$ follows some multinomial distribution and assigning a weight of $1/N$ to each offspring. The resampling approaches now mainly differ in the number of times a sample $\boldsymbol{X}^i_{1:t}$ is considered in $N^{1:N}_t$, usually based on its weight W^i_t. In most cases, this means that particles with high weights are considered in $\dot{\Upsilon}_t$ more often than particles with low weights, i.e. the number of offspring N^i_t is greater for a highly weighted particle than for a low weighted one. This ultimately satisfies an unbiased approximation of $\hat{\tau}(\boldsymbol{q}_{1:t})$ as the expectation $\mathrm{E}(N^i_t \mid W^{1:N}_t) = N^i_t W^i_t$, which implies that the weight W^i_t is proportional to the number of times N^i_t that the ith particle is resampled.

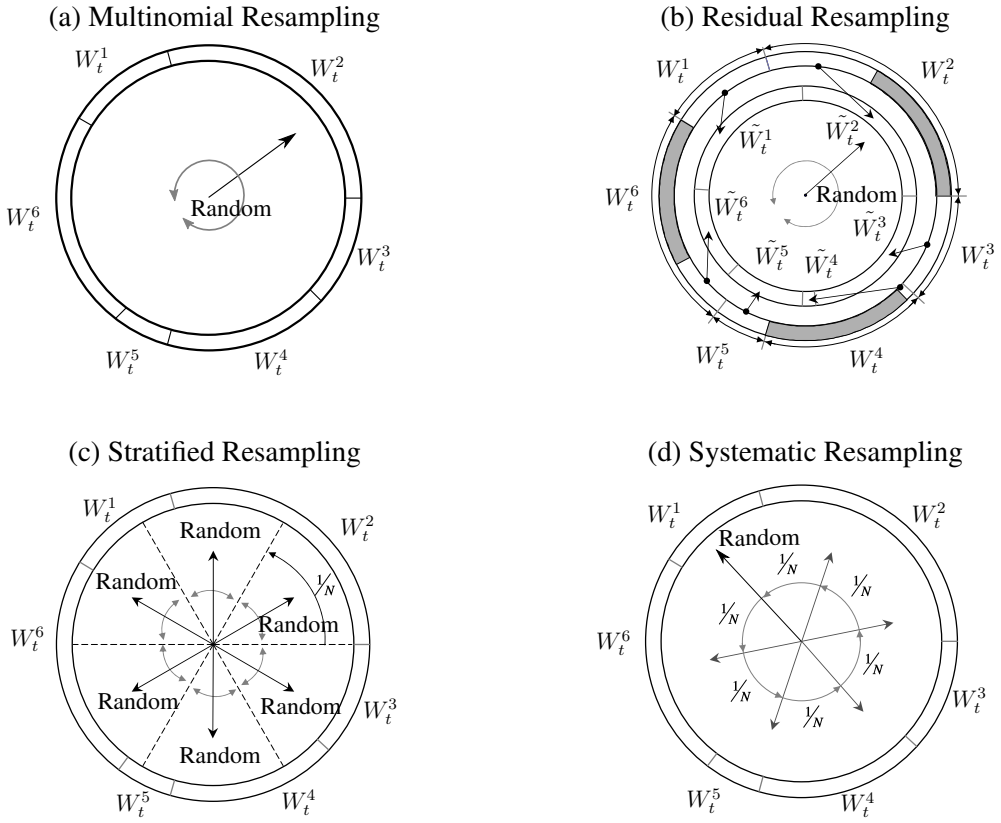


Figure 2.5: These figures demonstrate the traditional resampling schemes. The term "Random" indicates random selection from a uniform distribution, $\mathcal{U}[\cdot]$. The size of the weights W_t^i is directly proportional to their respective values. The figures are inspired by the work of [Bla09].

To most easy and straightforward approach is *multinomial resampling*. The basic principle is shown in Figure 2.5a. Here, a random number U^i is drawn from an uniform distribution $\mathcal{U}(0, 1]$ to pick a particle within a cumulative frequency of the weights from all particles in Υ_t corresponding to U^i. The concept of cumulative frequency, alternatively expressed as the inverse of the cumulative distribution function and thus the cumulative sum of the normalized weights, is represented as a circle. Consequently, the circumference corresponding to each particle is directly proportional to its weight W_t^i. By randomly choosing a direction within the circle, a particle is selected with a probability equal to its weight. This corresponds to a multinomial distribution of the parameters $(N, W_t^{1:N})$ for sampling $\dot{\Upsilon}_t$, hence the name multinomial resampling. The drawback of this approach is its computational complexity of $\mathcal{O}(N \log(M))$, where M is a factor resulting from finding the cumulative bound using, for example, a binary search [CCF99].

The basic idea of *residual resampling* is to determine the number of offspring for each particle based on the integer part of its weight according to $N_t^i = \lfloor NW_t^i \rfloor$, and then distribute the

remaining available slots among the particles that have a non-integer number of offspring, called residuals. This two step approach is shown in Figure 2.5b using two circles. The shaded areas in the outer circle illustrate the weights corresponding to a particle that satisfies $N_t^i \geq 1$, i.e. all weights that are bigger then $1/N$. Thus, the total number of residuals is $R_t = N - \sum_{i=1}^{N} N_t^i$, which are then selected by multinomial resampling according to the residual weight

$$\tilde{W}_t^i = W_t^i - \frac{N_t^i}{N} \quad . \tag{2.58}$$

By using residual resampling, particles with higher weights will have a higher chance of being selected multiple times, while still allowing particles with lower weights to be included. The implementation requires two loops with a complexity in the order of $\mathcal{O}(N - R_t) + \mathcal{O}(R_t)$.

Stratified resampling handles weight degeneracy by ensuring a more uniform representation of particles across the weight range. For this, the particle set Υ_t is divided into disjoint subintervals, called strata, of size $1/N$, i.e. one for each particle. Within each stratum, a single number U^i is randomly chosen using a uniform distribution $\mathcal{U}[i - 1/N, i/N]$. The offspring are then chosen based on the cumulative sum of normalized weights as in multinomial resampling. This selection process as shown in Figure 2.5c guarantees that particles from different weight ranges are included in the resampled set $\dot{\Upsilon}_t$.

The last traditional scheme is *systematic resampling* and by far the most popular and efficient. It can be seen as a simplified version of stratified resampling as the particle set is again divided into strata of size $1/N$, however instead of drawing U^i per strata, the whole interval is considered. For this an initial random number $U^1 = \mathcal{U}(0, 1/N]$ is drawn as starting point and the other numbers according to

$$U^i = U^1 + \frac{i-1}{N} \quad . \tag{2.59}$$

As can be seen in Figure 2.5d, this results in fixed increments of $1/N$ from U^1 and $\dot{\Upsilon}_t$ is again determined using the cumulative sum of normalized weights. As mentioned by [Kün05], systematic resampling violates the assumption of statistical independence between particles, since only U^1 is chosen randomly. The complexity for both systematic and stratified resampling is of the order of $\mathcal{O}(N)$, with systematic resampling being more computationally efficient because fewer random numbers need to be generated.

As mentioned above, there are countless other methods besides the traditional (single-distribution) ones, namely compound and special resampling. While the class of single-distribution methods always satisfies the condition of unbiasedness and equal weight for all resampled particles, the latter do not necessarily attempt to do so. Compound sampling methods group particles into non-overlapping subsets of the particle population based on predetermined

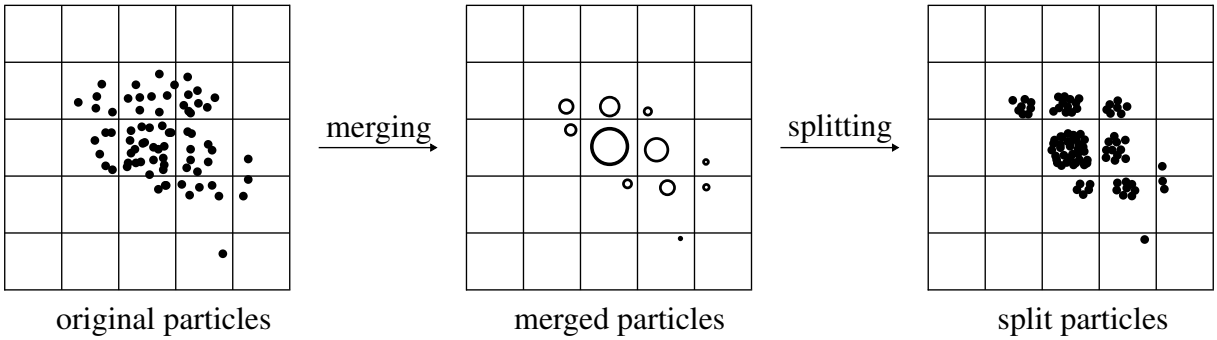


Figure 2.6: Deterministic resampling with particle merging and splitting in a two-dimensional state space. In merging, particles sharing a cell of the grid are merged at the average position, and the size of the circles represents the sum of the weights. After splitting, the particles are distributed according to some distribution, and the sum of the weights is equal to the original weight in that cell. The figure is based on the work of [LBD15].

criteria before resampling. Typically, grouping is determined by particle values and/or weight thresholds that allow particles with similar weights to be placed in the same group. Different resampling methods are then applied to each group. Compound sampling therefore serves purposes such as reducing execution time and preserving particle diversity [LBD15]. It's worth noting that particle grouping is often used in parallel resampling, which is usually achieved by clustering neighboring particles with similar indices.

Common operations for such approaches are particle merging and splitting. For example, *deterministic resampling* replaces the second step of residual resampling by merging particles using their residuals [LSS12]. The basic idea behind the operations can be seen in Figure 2.6. In merging, particles sharing a cell of the grid are merged at the average position, and the size of the circles represents the sum of the weights. This is especially useful when the number of particles is small, as it does not discard particles that occupy a cell, thus preserving diversity. In SLAM for robots this is a popular approach and an essential part of so-called occupancy grids [TBF05]. Splitting, on the other hand, replaces a particle with a weight greater than a predefined threshold with a set of particles distributed according to some distribution, e.g. a Gaussian. The sum of the weights is required to be equal to the original weight. So while merging reduces the number of particles, splitting increases it. This concept is also used to reduce the computational complexity of the SIS filter by performing merging before weighting and using splitting as resampling.

Of course, all this requires that the particles have some kind of geometric representation, such as a position vector. In practice, the state $\boldsymbol{q}$ often consists of a variety of different parameters, so only a subset can be used to satisfy this property. This may cause problems in some dimension of $\boldsymbol{q}$, as we have nothing to prevent degeneracy here. Resampling methods

that take this into account will be presented later in this thesis, as they require several practical constraints and parameterization that deserve further explanation.

An important thing the authors of [Che03] point out is that resampling rarely solves the weight degeneracy problem completely, it just saves further computation time by discarding the particles associated with insignificant weights. However, the time savings come at a price. The replication of mainly high weight particles poses additional problems, as it inevitably introduces a high correlation between particles. The most crucial problem is the phenomenon of *sample impoverishment*. It occurs due to a decreasing diversity of particles, as low weight particles are discarded and only duplicates of high weight particles are preserved. This leads to a high concentration of them at similar positions and therefore a poor representation of the underlying probability density and ultimately worse estimation results. In the context of sensor fusion, the problem can be serious enough that the estimator completely loses track, gets stuck, and never recovers [Fet17].

The most obvious solution for solving impoverishment is by only executing a resampling when absolutely necessary. Most of the time, this is achieved using the effective sample size N_{eff} as shown in (2.57). If N_{eff} is below a certain threshold, then start the resampling process. However, it is quite easy to understand that the effect of impoverishment is only slowed down but not eliminated. Additionally, in scenarios where rare events or outliers are of particular interest, N_{eff} may not accurately reflect their importance. It is also not able to consider the concentration of the weight. In some cases, even with a high N_{eff}, the weights may be highly concentrated on a few particles, indicating a lack of diversity. Thus, detecting impoverishment using N_{eff} is not possible.

Solving the problem of sample impoverishment is of great interest in this thesis, and several different approaches, including modifications of some of the above methods, will be presented in Section 6.

2.3.3 State Estimation

At the beginning of this chapter, we introduced classical point estimators. Then we found that for solving time-sequential problems, especially in the context of sensor fusion, it is often more useful to estimate the complete posterior or, in the general case, the target distribution. The methods of choice were Bayesian filters approximated with SIS and an additional resampling step. The advantage of directly estimating the distribution or its density function has been discussed several times and, in summary, simply brings the advantage of a non-linear and recursive view of the problem over the entire state space. For many practical applications, however, it is still of interest to additionally obtain a state estimate per time step. For example, in the case of

localization, the position of an object can be specified on the basis of the point estimate. The uncertainty/variance then results from the scattering behavior of the estimated density.

As the target is approximated using a set of particles $\Upsilon_t = \{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$, the most common approach is calculating the expected value of the distribution. For a discrete number of weighted samples, this is usually done by calculating the weighted average state $\boldsymbol{q}^{\text{wa}}_t$ using

$$\boldsymbol{q}^{\text{wa}}_t = \mathrm{E}(\Upsilon_t) = \frac{\sum^N_{i=1} \boldsymbol{X}^i_t \cdot W^i_t}{\sum^N_{i=1} W^i_t} \quad . \tag{2.60}$$

Here the denominator must be greater than zero, otherwise the weighted average is undefined. For a unimodal distribution, this is often the desired solution, as it adds some smoothness to the estimation trajectory over time. However, as Bayes filters can model arbitrary distributions, e.g. non-linear and non-Gaussian, multimodalities are possible. These are particularly likely to occur in applications with restrictive model dynamics, such as no walking through walls or other thresholds restricting the state space [Fet16].

The simplest way to deal with multimodalities is to select the particle with the highest weight as a point estimate [Rek04]. However, this requires dealing with situations where several particles share the highest weight or detecting possible outliers that have received the highest weight due to things like an erroneous observation. To solve such scenarios different deterministic approaches are presented in the literature. One is the already mentioned merging operation as illustrated in Figure 2.6. Here, the point estimate is defined by the cell with the highest sum of weights. This adds a higher robustness against outliers and densities with narrow multiple modes, but highly depends on the grid size and resolution. For high dimensional scenarios this additional adds computational complexity to check if a particle is inside some polytope, e.g. a polygon or polychoron. Other approaches would only select an empirically defined percentage of the best-weighted particles or adding a threshold for min weights [Ebn21]. However, deterministic solutions are highly dependent on the application, so it is often more advisable to eliminate the cause of (unwanted) multimodalities with more appropriate sensor models than to introduce some hard-to-verify heuristics.

Under asymptotic conditions, i.e. as $N \to \infty$, the max-particle approach should give the desired result, as it shows the point with the highest probability according to all sensor models. However, having only a limited number of particles, which also suffer from degeneracy and impoverishment effects, can alter a correct representation of the target distribution, preventing samples from being located in areas where the density is highest. In simpler terms, even if a set of particles approximates the target distribution quite well, there may be no sample to choose from at the exact points of the modes of a distribution. The maximum particle as a point estimate therefore does not necessarily represent the highest / dominant mode of the distribution.

One possible solution to this problem are so-called mode-seeking algorithms. The best known is the mean shift method [Che95], as it is extensively used for clustering in unsupervised learning and classification. The basic idea can be understood as a process of iteratively shifting the center, or mean, of a window within the state space towards the regions of high data density (weights). This shifting process is performed for each sample in the particle set until convergence is reached. Given a weighted sample set $\Upsilon_t = \{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$ of size N and a kernel function K, mean-shift utilizes the concept of a *kernel density estimation* (KDE) and *gradient ascent*. Consider a density estimator $\hat{f}_t$ which estimates the probability density function of τ_t at point $\boldsymbol{x} \in \mathbb{R}^d$ using

$$\hat{f}_t(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^N W^i_t} \sum_{i=1}^N \frac{W^i_t}{h} K\left(\frac{\boldsymbol{x} - \boldsymbol{X}^i_t}{h}\right) \quad , \tag{2.61}$$

where $h \in \mathbb{R}^+$ is an arbitrary smoothing parameter called bandwidth [Bul18]. The general formula of gradient ascent from generation k to generation $k+1$ is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{m}(\boldsymbol{x}_k) \quad . \tag{2.62}$$

The mean shift vector $\boldsymbol{m}(\boldsymbol{x}_k)$ for any generation k is then given by

$$\boldsymbol{m}(\boldsymbol{x}_k) = \frac{\sum_{i=1}^N \boldsymbol{X}^i_t \cdot g(\boldsymbol{x}_k - \boldsymbol{X}^i_t) \cdot W^i_t}{\sum_{i=1}^N g(\boldsymbol{x}_k - \boldsymbol{X}^i_t) \cdot W^i_t} - \boldsymbol{x}_k \tag{2.63}$$

where $g(\dots) = -K'(\dots)$ and the weights need to be normalized to 1 [Che95]. In most cases, a Gaussian kernel is used due to its simplicity and fast computation. However, it does not matter if a non-optimal kernel is chosen, as the quality of the KDE component depends primarily on the bandwidth h [Bul18].

The process steps are pretty straightforward and are visually explained in Figure 2.7. As input we need the bandwidth h, a convergence threshold $\hbar$, optimally $\hbar \simeq 0$, and a maximum number of iterations for the gradient ascent if convergence cannot be reached. For each particle in Υ_t we initialize the starting point $\boldsymbol{x}_1$. By calculating $\boldsymbol{m}(\boldsymbol{x}_k)$ using (2.63), we then update $\boldsymbol{x}_1$ by shifting it by the vector $\boldsymbol{m}(\boldsymbol{x}_k)$ according to (2.62). If the shift magnitude $\bar{x} = ||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||$ is greater than the threshold $\hbar$, repeat with $\boldsymbol{x}_{k+1}$ or otherwise select the next particle. So for each generation k, the mean shift takes a step in the direction of the gradient, moving uphill. This step is aimed at reaching a higher density region within the probability density. In a multimodal setting, this automatically generates clusters as particles converge on the mode they belong to by moving uphill [HZ03]. Consequently, the point estimate is then the mode with the highest density value or, depending on the scenario, according to some specific heuristics. Although this approach is very promising and gives good results if the bandwidth is chosen correctly, the

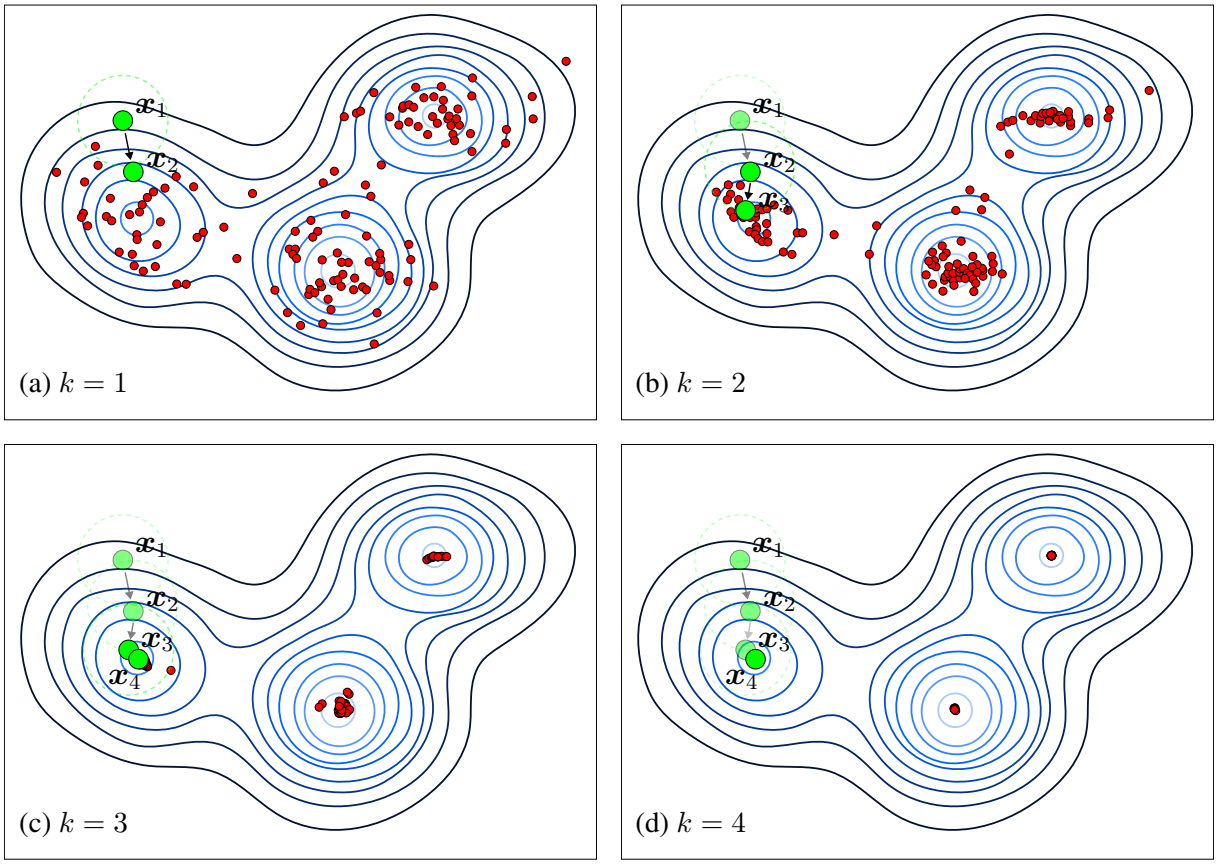


Figure 2.7: Illustration of the mean shift algorithm for mode finding in a multivariate Gaussian. The lighter the blue, the higher the density. From (a) to (d) one iteration step k is shown for all particles running the algorithm in parallel. In (d) they converge into the three modes. The green circles visualizes the mean shift procedure for a single particle according to (2.63).

complexity is quite high at $\mathcal{O}(N^2)$. In addition, the choice of bandwidth depends strongly on the use case and is therefore not trivial.

Finally, the state estimation process completes the presented generic framework for SMC. We have thus introduced an extremely powerful tool for solving non-linear and non-Gaussian problems sequentially in time. Unlike traditional numerical integration methods, SMC is very easy to implement, works under weak assumptions and performs well when applied to complex and high dimensional state spaces. In combination with HMM and thus sequential dynamics, it becomes an excellent solution not only for the filtering problem, but also for advanced filtering (see Section 5) approaches that do not necessarily rely on the basic posterior of the form $\tau(\boldsymbol{q}_{1:t})$. Further discussion of the convergence results and stability of SMC can be found in [DJ09]. An even more general framework, where the SIS algorithm can be viewed as a particle interpretation of a Feynman-Kac model, is presented in great detail in [CP20].

2.4 Particle Filter

At this point, all the concepts introduced above come full circle, resulting in the very popular class of *particle filters*. As discussed in Section 2.2, our goal in sensor fusion is to make inferences about $p(\boldsymbol{q}_{1:T} \mid \boldsymbol{o}_{1:T})$ (cf. (2.24)) sequentially in time. According to the HMM presented in Section 2.2.1, this can be put into a time-discrete dynamic context with recursive behavior, yielding the Bayes filter of the form (2.16). The Bayes filter in non-linear non-Gaussian dynamical models thus relies on the posterior $p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ and, more importantly, its marginals $(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t})$ for each time step t. The (classic) particle filter is a direct application of the SMC methods to this sequence of target distribution $\tau(\boldsymbol{q}_{1:t}) = p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ and its marginals. Various more advanced algorithms are available for more complex sequences of target and importance distribution, e.g. auxiliary particle filter or Rao-Blackwellised particle filtering [DJ09]. In the following we will concentrate on introducing and discussing methods to approximate the Bayes filter as seen in Algorithm 1 according to the generic SIS framework in Algorithm 3.

We start again with a generic framework, often called the *sequential importance resampling* (SIR) particle filter, because it is the direct application of SIS to Bayes filtering with resampling. Let $\gamma(\boldsymbol{q}_{1:t}) = p(\boldsymbol{o}_{1:t} \mid \boldsymbol{q}_{1:t})p(\boldsymbol{q}_{1:t})$ be known point-wise, yields the desired target distribution $\tau(\boldsymbol{q}_{1:t}) = p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ and normalization constant $Z_t = p(\boldsymbol{o}_{1:t})$ according to (2.24) and (2.42). As seen in the toy example of SIS (cf. (2.53)) the choice of an appropriate importance density $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})$ (cf. (2.50)), which samples new particles, is crucial for a good approximation. To achieve this, we want to minimize the conditional variance of the incremental importance weights $\alpha(\boldsymbol{q}_{1:t})$ (cf. (2.52)) calculated with respect to the importance $q(\dots)$. Given the variance

$$\begin{aligned}
&\mathrm{Var}^{q(\dots)}(\alpha(\boldsymbol{q}_{1:t})) = \mathrm{Var}^{q(\dots)}\left[\frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})}\right] \\
&= \mathrm{E}^{q(\dots)}\left[\frac{\gamma^2(\boldsymbol{q}_{1:t})}{\gamma_{t-1}^2(\boldsymbol{q}_{1:t-1})q^2(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})}\right] - \left\{\mathrm{E}^{q(\dots)}\left[\frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})}\right]\right\}^2
\end{aligned} \tag{2.64}$$

only requires to minimize the first term as the second does not depend on $q(\dots)$. A lower bound that is independent of $q(\dots)$ is provided by Jensen's inequality

$$\begin{aligned}
&\mathrm{E}^{q(\dots)}\left[\frac{\gamma^2(\boldsymbol{q}_{1:t})}{\gamma_{t-1}^2(\boldsymbol{q}_{1:t-1})q^2(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})}\right] \geq \left\{\mathrm{E}^{q(\dots)}\left[\frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1})}\right]\right\}^2 \\
&= \left\{\int \frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})} d\boldsymbol{q}_t\right\}^2
\end{aligned} \tag{2.65}$$

and then obtained by choosing:

$$\begin{aligned} q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1}) &= \frac{\gamma(\boldsymbol{q}_{1:t})}{\gamma_{t-1}(\boldsymbol{q}_{1:t-1})} = \frac{\gamma(\boldsymbol{q}_{1:t})}{\int \gamma(\boldsymbol{q}_{1:t}) d\boldsymbol{q}_t} \\ &= \frac{p(\boldsymbol{o}_{1:t} \mid \boldsymbol{q}_{1:t}) p(\boldsymbol{q}_{1:t})}{\int p(\boldsymbol{o}_{1:t} \mid \boldsymbol{q}_{1:t}) p(\boldsymbol{q}_{1:t}) d\boldsymbol{q}_t} = \frac{p(\boldsymbol{o}_{1:t} \mid \boldsymbol{q}_{1:t}) p(\boldsymbol{q}_{1:t})}{p(\boldsymbol{o}_{1:t} \mid \boldsymbol{q}_{1:t-1}) p(\boldsymbol{q}_{1:t-1})} \\ &= p(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1}, \boldsymbol{o}_{1:t}) \overset{\text{Markov}}{=} p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t) \\ &= \frac{\overbrace{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)}^{\text{evaluation}} \overbrace{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})}^{\text{transition}}}{\underbrace{p(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})}_{\text{predictive}}} \propto p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) \end{aligned} \tag{2.66}$$

Here, the last line is achieved by assuming a Markov process for the states and conditional independence of observation, similar to the derivation of the Bayes filter in 2.2. For the SIS in a Bayesian context the above proofs an optimal importance density of the form

$$q(\boldsymbol{q}_t \mid \boldsymbol{q}_{1:t-1}) = q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t) \ . \tag{2.67}$$

With this, the incremental weight function $\alpha(\boldsymbol{q}_{1:t})$ (cf. (2.52)) can be denoted as

$$\alpha(\boldsymbol{q}_{1:t}) \overset{\text{Markov}}{=} \alpha(\boldsymbol{q}_{t-1:t}) = \frac{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})}{q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t)} \tag{2.68}$$

Note, that the two statistical models; evaluation $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ and transition $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ of Bayes filtering are again present.

The resulting SIR particle filter is given in Algorithm 4. The initialization phase takes place in lines 2 - 4 and involves the predefinition of the a priori distribution $\mu(\boldsymbol{X}_1^i)$ and the assignment of weights to particles based on this distribution. For time instances $t \geq 2$, a new particle $\boldsymbol{X}_t^i$ is sampled, followed by weight computation. The evaluation model $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ incorporates the probabilistic sensor models, while the state transition model $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ describes the dynamics between successive states and time steps. Finally, a resampling step is applied when the effective sample size N_{eff} falls below a predefined empirically determined threshold $\hbar$. In practice, the resampling step is often performed before the loop of a new update starts in line 1, as this allows a subsequent state estimation process to use the particle set weighted by line 7 instead of having equal weights after resampling.

As already mentioned, this is a generic framework, as obviously sampling from the optimal importance is only possible in rare cases. Moreover, by utilizing equation (2.67), the incremen-

Algorithm 4 Sequential Importance Resampling (SIR)

Input: A Priori $\mu(\boldsymbol{X}_1^i)$

for $i = 1$ **to** N **do**
  if time $t = 1$ **then** ▷ Initialization
    Sample $\boldsymbol{X}_1^i \sim q(\boldsymbol{q}_1 \mid \boldsymbol{o}_1)$
    Compute the weights $W_1^i \propto \dfrac{\mu(\boldsymbol{X}_1^i)p(\boldsymbol{o}_1 \mid \boldsymbol{X}_1^i)}{q(\boldsymbol{X}_1^i \mid \boldsymbol{o}_1)}$
  else if time $t \geq 2$ **then** ▷ Importance Sampling
    Sample $\boldsymbol{X}_t^i \sim q(\boldsymbol{q}_t \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_t)$
    Compute the weights $W_t^i \propto \alpha(\boldsymbol{X}_{t-1:t}^i) = \dfrac{p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)\, p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i)}{q(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_t)}$
  end if
end for
if $N_{\text{eff}} \leq \hbar$ **then** ▷ Resampling
  Resample $\{W_t^i, \boldsymbol{X}_t^i\}$ to obtain N new equally-weighted particles $\{\frac{1}{N}, \boldsymbol{X}_t^i\}$
end if

tal weights become proportionally to the *predictive likelihood*, which can be given as

$$\alpha(\boldsymbol{q}_{1:t}) \propto p(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1}) = \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) d\boldsymbol{q}_t \ . \tag{2.69}$$

However, it can also be difficult to evaluate this density analytically. Consequently, finding an appropriate importance density is of great interest when using the SIR for sensor fusion applications.

Several authors like [Che03], [DJ09] or [CP20] have discussed how to choose an appropriate importance distribution in the past. Of course, everyone agrees that the variance should be as low as possible and the shape as close as possible to the target distribution. The importance distribution should have a broader distribution, i.e. the support should cover the posterior. Depending on the number of particles used, it is often useful to choose a distribution that has linear complexity. Especially in real-time applications with high update rates, this can be a critical factor. If the transition and evaluation can be described analytically well, this should be taken into account as well. This prevents a possible discrepancy between the drawing of new samples and the weighting, which is mainly responsible for weight degeneracy. Finally, possible outliers should be taken into account, as these can be sharp changes in the estimation, depending on the application. For this purpose, a long-tailed behavior of the importance is usually implemented, i.e. the distribution is asymptotically flattened at the edges and only small, but not negligible, density values are present at the edges.

Algorithm 5 CONDENSATION Particle Filter Algorithm

Input: A Priori $\mu(\boldsymbol{X}_1^i)$

1: **for** $i = 1$ **to** N **do**
2: **if** $t = 1$ **then** ▷ Initialization
3: Sample $\mu(\boldsymbol{X}_1^i)$
4: Compute $W_1^i \propto p(\boldsymbol{o}_1 \mid \boldsymbol{X}_1^i)$
5: **else if** time $t \geq 2$ **then**
6: Sample $\boldsymbol{X}_t^i \sim p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i)$ ▷ Transition
7: Compute $W_t^i \propto p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)$ ▷ Evaluation
8: **end if**
9: **end for**
10: **if** $N_{\text{eff}} \leq \hbar$ **then** ▷ Resampling
11: Resample $\{W_t^i, \boldsymbol{X}_t^i\}$ to obtain N new equally-weighted particles $\{\frac{1}{N}, \boldsymbol{X}_t^i\}$
12: **end if**

2.4.1 CONDENSATION Particle Filter

One of the most elegant and simple approaches is CONDENSATION (CONditional DENSity propgATION) approach, also known as Bootstrap filter [IB98]. Here, the transition $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ is chosen as importance distribution, i.e. $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t) \approx p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$. This greatly simplifies the incremental weight function to

$$\alpha(\boldsymbol{q}_{t-1:t}) = \frac{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})}{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})} = p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) \ , \tag{2.70}$$

where the transitions are reduced and only the evaluation model remains. As can be seen in Algorithm 5, CONDENSATION allows for a clear separation between transition and evaluation in two distinct steps. In practice, the initial weighting in line 4 is often done using either equal weights if nothing is known beforehand or by using a simple distribution like a Gaussian.

This approach is highly practical and is used in a wide range of sensor fusion applications, e.g. [Guo20; HM04; Kha13a; TBF05] and many more. It is so popular that when people talk about particle filters, they almost always refer to this approach. The main benefits are obvious: simplicity, low computational overhead and flexibility due to the clear separation of the probabilistic models involved. Furthermore, sampling from the transition model is in most cases straightforward as it already describes the dynamics of the system. As the transition is not used for weighting, we have no need to obtain densities from it, i.e. a proper probability density function is not required. This allows for using models that simulate the sampling process by updating the parameters of a new particle $\boldsymbol{X}_t^i$ given its predecessor $\boldsymbol{X}_{t-1}^i$ adding some noise for randomness. There are hardly any limits to such models, which range from simple linear-

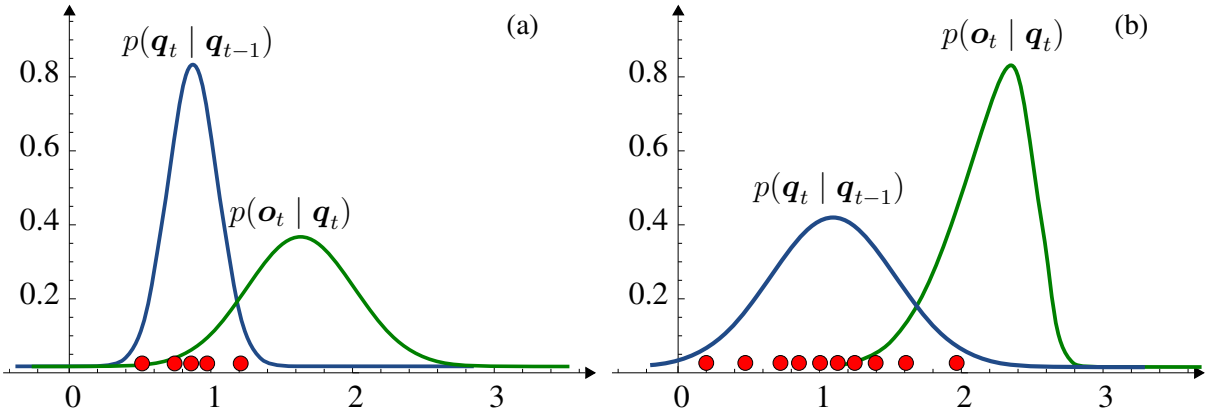


Figure 2.8: Examples of sample impoverishment in CONDENSATION particle filtering. In both cases, the overlapping region of the transition and evaluation support is too small to sample enough particles. This results in a poor overall approximation of the posterior.

dynamic approaches to complex economic processes and the simulation of human movements (cf. Section 4).

Despite its simplicity, CONDENSATION has several drawbacks that need to be considered. A major problem is its high variance because it does not take into account the most recent observation, which becomes apparent when the evaluation $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ has a sharp mode. This situation is illustrated in Figure 2.8b and occurs when the importance density has limited mass near the evaluations mode, so that the overlapping region of both distributions is less sampled. A similar effect is shown in Figure 2.8a, where the transition is not able to sample enough particles to support the evaluation. Therefore, CONDENSATION is prone to sample impoverishment, especially when encountering small measurement noise. Accurate observations also lead to high peaks in the evaluation density, making it difficult for the importance to sample outside these peaks [IB98], i.e. the measurement noise model is sensitive to the outliers. Therefore, as emphasized by [Che03], it is crucial to exercise caution when using this importance model and not to rely solely on its simplicity without a deep understanding of the underlying problem.

2.4.2 Auxiliary Particle Filter

As discussed before, the goal should be the choice of an importance distribution of the form (2.67), i.e. integrating the current observation $\boldsymbol{o}_t$. This further leads to $\alpha(\boldsymbol{q}_{t-1:t})$ being independent of $\boldsymbol{q}_t$ as shown in (2.69), which allows the order of the sampling and resampling steps to be interchanged [DJ09]. As discussed at the beginning of Section 2.3.2, in SIS/SIR particles are distributed according to the importance distribution and the weighting is thus applied according to the importance instead of the target. Resampling is added to represent the target more accurately, leading to a better approximation. This effect can be further increased by such

an interchange between sampling and resampling as it provides a greater number of distinct particles and therefore minimizes the loss of information.

If the importance weights $\alpha(\boldsymbol{q}_{t-1:t})$ are independent of the new state and the importance distribution corresponds the marginal distribution of the proposed states, then the process of weighting, resampling and sampling can be understood as follows: First, reweighting is performed to account for the discrepancy between the old and new marginal distributions of the former states. Second, resampling is performed to generate an unweighted sample. Finally, the new state is generated from its conditional distribution. However, it is obvious that this is only a theoretical construct, since in general $\alpha(\boldsymbol{q}_{t-1:t})$ depends on new states, which makes it impossible to simply swap the order of sampling and resampling. Nevertheless, the idea of using information from the next observation to determine particle survival during resampling leads to the desire to develop methods that can effectively use future information in a broader context. This would allow us to gain the same advantage in situations where the optimal importance distribution cannot be used.

The *auxiliary particle filtering* (APF) is such a method that adds the current observation $\boldsymbol{o}_t$ to the importance, resulting in a predictive method that aims to anticipate particles that are likely to occupy regions of high probability mass at time t, given the current knowledge at time $t-1$. In its original version, APF uses so-called auxiliary variables - hence the name, but several improvements and simplifications have been proposed, of which [CCF99] may be the most widely used. The authors of [JD08] further simplified the formalism and showed that the APF can be derived by utilizing an approximation $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ of the predictive likelihood $p(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ as defined in (2.69). For approximating our target $\tau(\boldsymbol{q}_{1:t}) = p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ this results in an incremental weight function (cf. (2.52) and cf. (2.66)) of the form

$$\begin{aligned} \alpha(\boldsymbol{q}_{t-1:t}) &= \frac{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)}{\int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) p(\boldsymbol{o}_t \mid \boldsymbol{q}_t) d\boldsymbol{q}_t \; q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t)} \\ &\approx \frac{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)}{\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1}) \; q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t)} \;\; , \end{aligned} \tag{2.71}$$

where the importance is again assumed to be optimal at first.

All the above considerations are summarized in Algorithm 6.

For an implementation of Algorithm 6 appropriate choices for $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ and $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t)$ need to be found. With CONDENSATION in mind, the most straightforward choice would be $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t) \approx p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ (transition) and $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1}) \approx p(\boldsymbol{o}_t \mid \mathrm{E}(\Upsilon_{t-1}))$ (state estimate). Choosing the transition as the importance has far fewer disadvantages with the APF, since the particle set has already been reweighted by the predictive on the basis of the current measurements, and the particles are distributed accordingly by the resampling step. Although

Algorithm 6 Auxiliary Particle Filter (APF)

Input: A Priori $\mu(\boldsymbol{X}_1^i)$

for $i = 1$ **to** N **do**
  if $t = 1$ **then** ▷ Initialization
    Sample $\mu(\boldsymbol{X}_1^i)$
    Compute $W_1^i \propto \dfrac{p(\boldsymbol{o}_1 \mid \boldsymbol{X}_1^i)\mu(\boldsymbol{X}_1^i)}{q(\boldsymbol{X}_1^i \mid \boldsymbol{o}_1)}$
  else if time $t \geq 2$ **then**
    if $N_{\text{eff}} \leq \hbar$ **then** ▷ Reweight & Resampling
      Reweight old particles $\tilde{W}_{t-1}^i \approx W_{t-1}^i \times \tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$
      Resample $\{\tilde{W}_{t-1}^i, \boldsymbol{X}_{t-1}^i\}$ to obtain N equally-weighted particles $\{\frac{1}{N}, \boldsymbol{X}_{t-1}^i\}$
    end if
    Sample $\boldsymbol{X}_t^i \sim q(\boldsymbol{q}_t \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_t)$ ▷ Importance
    Compute $W_t^i \propto \dfrac{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)}{\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})\, q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t)}$ ▷ Weighting
  end if
end for

this reduces the variance, it also increases the sensitivity to measurement errors as these can now be better represented by the probability density. This may seem like an advantage, and it is for general use, but it can also be a disadvantage in certain applications. For example, when tracking a target, it is often important to obtain as stable a position estimate as possible. Ignoring individual outliers, as these areas are simply not sampled, can be quite advantageous, especially for sensors with frequent erroneous readings. Because of these properties, multimodal densities are more common in practice with APF, which is not always desirable.

The choice of $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ must be made carefully, as a high variance of this distribution will directly affect the overall approximation. As a result, frequent resampling becomes necessary, leading to an approximation where only a few or even a single unique particle remains, i.e. the phenomenon of sample impoverishment [Fet17]. To compensate for impoverishment, a distribution $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ with thicker tails than $p(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1})$ should be chosen and the predictive likelihood should also encode the same degree of uncertainty as the target [DJ09]. The above suggestion of $\tilde{p}(\boldsymbol{o}_t \mid \boldsymbol{q}_{t-1}) \approx p(\boldsymbol{o}_t \mid \mathrm{E}(\Upsilon_{t-1}))$ is often recommended in the literature, but could lead to large or even infinite variance as (2.71) is not upper bounded. Nevertheless, degeneracy and impoverishment are systematic problems that affect all of the particulate filter methods presented and therefore require more advanced techniques. This is one of the main contributions of this thesis, in which Chapter 5 present concrete approaches for application in sensor fusion, especially localization.

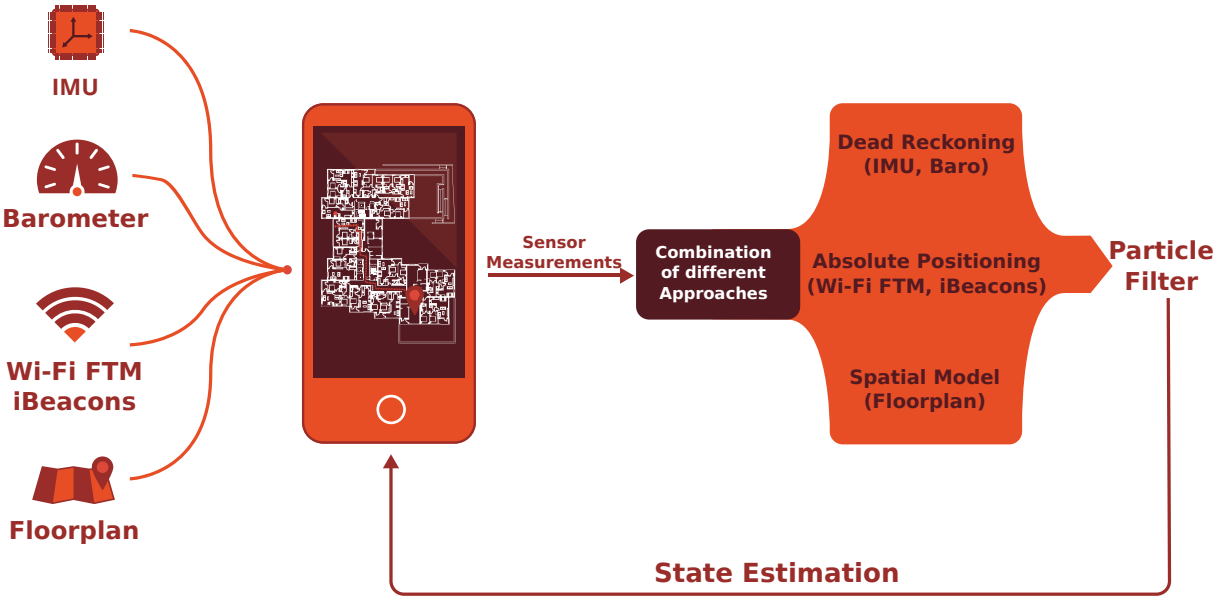


Figure 2.9: Simplified architecture overview of a smartphone-based ILS using particle filters for state estimation. The various sensor measurements are the observations used as input to the models involved. For each filter update, a state estimation is performed to obtain a position within the building.

2.5 Localization and Navigation

As described in the introduction to this thesis, pedestrian localization and navigation in buildings is a very exciting use case for sensor fusion. Hybrid ILS use many different sensor models to provide the most robust and accurate estimate of the pedestrian's position. The combination of physically and technologically different sensors such as radio frequencies, map information or movement models is the challenge for sensor fusion. In addition, there are other challenges already set by the scenario. For example, the need for high accuracy, architectural peculiarities, difficult material properties of buildings, the many degrees of freedom of human movement and many more.

In this section we will present the main structure of our ILS. This will form the basis for all further research results and methods presented in this thesis. A simplified architectural overview is illustrated in Figure 2.9 for a smartphone-based ILS. We consider indoor localization as a time-sequential, non-linear and non-Gaussian state estimation problem. We solve it using an extended version of the CONDENSATION particle filter which recursively updates a state estimate within a fixed time interval. As seen in Section 2.3, this allows an easy modularization of the sensor fusion problem. As discussed earlier, the main advantage of CONDENSATION is the clear separation between transition and evaluation. This makes it possible to model human walking behavior by sampling a new state based on previous knowledge and not just a fixed

density. It further opens up possibilities for complex transition models in contrast to conventional distributions such as a Gaussian. However, the presented system could also be applied to a wider range of Bayesian filters and can therefore be seen as a general approach.

2.5.1 Generic Filter Framework

We have introduced particle filter obtain numerical results for time-sequential, non-linear and non-Gaussian state estimation problems like indoor localization. However, we have chosen a general formalism for our ILS approach that allows for all kinds of Bayesian filters, not just MC methods. This kind of general formulation has already been cultivated in this way in all our previous papers. Meanwhile, many other scientific publications in the field of indoor localization also use this notation (e.g. [Jaw17; SJ18; Sol16]). In this way, the ILS is separated from the concrete implementation and its peculiarities. The result is a clear structure and a concise notation that allows a quick comparison of individual methods. The subsequent implementation also benefits from this, since a modularity of the models results quite naturally.

The general formalism for recursive Bayesian filtering was introduced in Section 2.2.2. Given all observations $\boldsymbol{o}_{1:t}$ up to the current time t, we can compute an estimate of the Markovian hidden state $\boldsymbol{q}_t$ given the posterior distribution:

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) \propto \underbrace{p(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1})}_{\text{evaluation}} \int \underbrace{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})}_{\text{transition}} \underbrace{p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1}) d\boldsymbol{q}_{t-1}}_{\text{recursion}} \quad . \tag{2.72}$$

The attentive reader may notice that the evaluation has been extended by the previous state $\boldsymbol{q}_{t-1}$ and the transition by the previous observation $\boldsymbol{o}_{t-1}$ in contrast to (2.35). A derivation, under the assumption that the posterior we are looking for is $p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}, \boldsymbol{q}_{t-1})$ has been proposed by [KGD14]. The disadvantage of this approach is that the normalization is no longer independent of the state variable. The general idea that the denominator of Bayes' theorem is constant thus becomes false. Such details can often be neglected in a later implementation, but can lead to numerical problems with advanced methods like smoothing [Che03].

Taking advantage of the different assumptions for a Bayes filter, conditioning the posterior with $\boldsymbol{q}_{t-1}$ is not necessary. In equation (2.29), we have applied Bayes' rule to our posterior:

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) \propto p(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{o}_{1:t-1}) p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1}) \quad . \tag{2.73}$$

By the means of conditional independence and the law of total probability, it results in

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t}) \propto p(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{o}_{1:t-1}) \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{1:t-1}) p(\boldsymbol{q}_{t-1} \mid \boldsymbol{o}_{1:t-1}) d\boldsymbol{q}_{t-1} \quad , \tag{2.74}$$

this time, without considering the Markov property at first. In the derivation of (2.35) a *first order* Markov property is assumed to hold. This means that past observations do not add information to the current state or influence the stochastic evaluation of future steps. In probability theory, this is also called *memorylessness*. To include $\boldsymbol{q}_{t-1}$ in the evaluation process, (2.72) is assumed to have a *second order* Markov property, allowing the previous state to be added to *memory*. Similar to [KGD14] or [Thr03], the observation $\boldsymbol{o}_{t-1}$ can be preserved within the transition of (2.72), since it does not precede the state $\boldsymbol{q}_{t-1}$.

The observation $\boldsymbol{o}_{t-1}$ is integrated into the state transition to allow for sensor information such as the pedestrian's orientation change, footsteps, or altitude for the movement models. In the context of the CONDENSATION particle filter, this therefore leads to an importance distribution of the form $q(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_t) \approx p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$, adding the observation to the sampling process. With respect to the derivation of the optimal importance in (2.66), this is a huge plus as it reduces the variance of the approximation. This extension brings the benefits of ADF to the CONDENSATION algorithm. The authors of [Thr03] do something similar, by introducing an additional control variable $\boldsymbol{u}_t$. The transition is then given as $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{u}_t)$. This extension is sometimes used in robotics. Here, a robot controls its motions (e.g., rotation and acceleration) or manipulates objects based on a precise set of commands. The control variable $\boldsymbol{u}_t$ thus contains the robot's movement inputs, which can be added as parameters to the transition. Observations $\boldsymbol{o}_t$ contain measurements from additional sensors like LIDAR or radar that help the robot to get information about the state of its environment.

This notation can also be useful for ILS, especially for complex movement models [Ebn22]. Since there are no movement inputs available for a human user, this information is instead obtained from the smartphone's sensors using PDR, e.g., step detection and orientation change. Thus, $\boldsymbol{u}_t$ contains data to control the behavior of the transition. Especially with a fusion of many sensors, $\boldsymbol{o}_t$ quickly becomes very high dimensional due to the amount of data. The introduction of $\boldsymbol{u}_t$ reduces the dimensionality by separating the data for evaluation and transition. A less relevant advantage in practice is the difference in temporal context. The control variable has all information for the transition at time t, while the transition can formally only be extended by the observation $\boldsymbol{o}_{t-1}$. A major drawback is that the introduction of another variable increases the complexity of the presented approaches without any added value. In addition, many advanced SMC methods have been developed without a second conditional variable in mind. The authors of [Thr03] state that the separation of control and observation is strictly for convenience and has no effect on the underlying statistics. Since this notation is rarely used in the field of indoor localization and has almost no use in the environment of SMC methods and probability theory, we will avoid it.

2.5.2 State and Observation

As we already know, by modeling the system with a HMM, a sequence of hidden states $\boldsymbol{q}_{1:T}$ is indirectly observed by noisy measurements $\boldsymbol{o}_{1:T}$. In the context of self-localization, a state represents the pedestrian's pose, which includes its location and orientation relative to a coordinate frame. For indoor localization and navigation, orientation angles are limited to the walking direction (yaw or absolute heading). This is also called a rigid system in planar environments. Depending on the use case, the state can contain a variety of variables that describe not only the pedestrian's pose, but also his environment. For example, [HB22] and [Har12] add the map representation (e.g. landmarks or occupancy grid) for a pedestrian SLAM as state variables. The number of environment variables then depends on the size of the building and the desired granularity. Thus, such an approach can reach an extremely high dimensionality for $\boldsymbol{q}$.

In this thesis, we study the self-localization of pedestrians in a known environment. The floorplan can be considered as a priori information for the ILS. This is also referred to as *static* state variables. Since they are constant and do not change over time, any representation of $\boldsymbol{q}_{1:T}$ (e.g. a sampled particle) contains the same data about it. In contrast, variables that change constantly over time, as described in the paragraph above, are called *dynamic*. In continuous-time state estimation, static variables can be neglected due to their constant properties, and only the dynamic ones are included in $\boldsymbol{q}$. In the sense of an implementation, one could also speak of a global data structure for static variables, which is accessible throughout the entire filtering process.

Given all this, our hidden state $\boldsymbol{q}$ is described by

$$\boldsymbol{q} = (x, y, z, \Theta), \quad x, y, z, \Theta \in \mathbb{R} \ . \tag{2.75}$$

Here x, y, z represent the position in three-dimensional Euclidean space and Θ the absolute heading of the user. In the context of particle filtering, a particle $\{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$ is thus a weighted representation of a possible state $\boldsymbol{q}$.

Especially in the field of geodesy it is common to use a geographic coordinate system with real longitude, latitude and ortometric height for x, y, z. If a floorplan is used, it must also be described by geographic coordinates. Measuring a building in such a global system requires GPS reference measurements, which is often an additional initial effort since few building plans contain such information. The creation of a floorplan based on services such as OpenStreetMap, Google Maps or Bing Maps is not recommended due to inconsistencies and inaccuracies of the data entered there [Cie10]. We refrain from the direct use of a global geographic coordinate system, as this usually does not bring any advantages in practice. The complexity and also the computational effort increase due to the more complex calculations and the need for more

decimal places. Since usually only the final estimate of the position is relevant, e.g. for visualization in a navigation application, a transformation from the Euclidean to the geographic system is sufficient for this purpose.

Before describing the observation vector $\boldsymbol{o}$, consider again the time-sequential hidden Markov process of Bayes filtering. For each time step t, a filter update consisting of transition, evaluation, and resampling is invoked to compute the posterior density $p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t})$. There are different strategies how often and with which data such an update is performed. If one sticks close to the filtering theory, a full update would have to be performed for each new incoming measurement. As described in [Ebn22], such an approach has practical drawbacks. For example, the dynamics of the system are often modeled using a PDR based on step detection. However, a step is only detected when it has already been taken, resulting in a constant time discrepancy of one step length between the transition density and the observations of the evaluation. To compensate for this effect, a constant velocity model can be assumed [Dic16]. In addition, a higher computational effort is expected due to more updates, which can quickly become a problem, especially for smartphones.

To avoid this, some authors rely on a strategy that performs an update only at each detected step [Hil14; Rac16]. All other measurements, e.g. Wi-Fi or relative heading change, are accumulated until this time. Since the transition is performed before the evaluation, the detected steps can be essentially pre-dated to the detected start time, thus avoiding a constant discrepancy. The disadvantages of this strategy are obvious. If a pedestrian stops to look at an exhibit in a museum, for example, no further updates are performed. Sensor readings continue to accumulate, resulting in outdated observations.

One of the most common strategies is to update the filter after a fixed time interval [Ebn15; Hel13; Pip19]. Similar to the stepwise variant, a detected step can be pre-dated to its start. The size of the interval can be found by considering the requirements of the system. For example, how often a new position estimate should be displayed to the user in a UI, or based on some hardware requirements. As stated by [Ebn22]: Depending on the configured interval, a set of accumulated measurements may contain multiple steps that need to be considered for the transition. In addition, using a particle filter for implementation may exacerbate the problem of sample degeneracy, since the resampling process is not performed until after each accumulation window. This also slows down the ability of the evaluation to correct for inaccuracies introduced by the transition, e.g. correcting for a falsely detected turn or step. Our commercially available system [Ste] has shown that a time interval of $300\,\mathrm{ms}$ to $1000\,\mathrm{ms}$ is a reasonable benchmark for most standard ILS.

The variables of the observation vector $\boldsymbol{o}$ depend on the considered scenario and the sensors used. However, in order to give the reader an overview of all possibilities in this work, we

define a "most valuable" observation vector, including all sensor measurements discussed later, as follows:

$$\boldsymbol{o} = (\boldsymbol{s}_{\text{wifi}}, \boldsymbol{s}_{\text{ble}}, \boldsymbol{s}_{\text{ftm}}, \Delta\theta, n, \Delta\psi, \boldsymbol{\Omega}) \ , \tag{2.76}$$

where $\boldsymbol{s}_{\square}$ contains the measurements of all nearby radio transmitters of the respective technologies Wi-Fi, BLE or Wi-Fi FTM. To provide a PDR model, $\Delta\theta$ and n describe the relative angular change and the number of steps detected for the pedestrian since the last filter step. The relative barometric pressure $\Delta\psi$ is given with respect to a fixed initial reference. Finally, $\boldsymbol{\Omega}$ contains the results of an activity detection specified for the pedestrian. Recognized activities are standing, walking, ascending and descending. All individual measurements consist of a tuple of timestamp and measured value.

The purpose of ILS is to determine one's position within a building and to navigate to a desired destination. Therefore, there is no a priori knowledge of the current location, which is why the system is initially distributed uniformly. However, if there are no absolute sources of information, as in most PDR-only solutions, it is usually assumed that the starting position is known to some degree [Köp14]. If the basic architectural structure of a building is very different, i.e. rooms have different layouts and corridors do not overlap identically over several floors, a PDR-based ILS can still converge even with a uniformly distributed initial density. This can be achieved by integrating a restrictive floorplan that does not allow walking through walls [Ebn22; Fet18; Kna17]. The floorplan acts as a kind of fingerprint for each area. The more different they are, the faster we can rule out false hypotheses (e.g. particles running into a wall).

Although the sensor and movement models discussed below would be applicable to other implementations of the Bayes filter, we will focus explicitly on particle filters. Due to their nonlinear properties and the representation of a probability density by weighted samples $\{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$, special considerations arise. In particular, the separation into context-dependent particle groups, their dynamics, and the individual manipulation of the particle weight yield exciting insights. Finally, this notation facilitates the subsequent formalism and enhances the understanding of the developed methods.

Chapter 3

Sensor Models

Until now, the term sensor measurement has been interpreted very broadly, e.g. activity detection or step detection require pre-processing by appropriate algorithms. Both are based on IMU measurements. At this point, it would be conceivable to introduce a distinction between physical and virtual sensors. The authors of [MKS] suggest the following definition: A physical sensor is a sensor that responds to a physical stimulus (e.g., temperature, light, pressure, magnetism, or motion) and transmits a resulting impulse-typically through electrical signals that can be captured and stored in digital form. In contrast, a so-called virtual sensor is a pure software sensor that autonomously generates signals by combining and aggregating signals it receives (synchronously or asynchronously) from physical or other virtual sensors. Depending on the device, it is not so easy to determine whether a sensor really provides raw data or whether some form of pre-processing, such as smoothing or pre-calibration, is calculated in the device software. For simplicity, we consider anything that provides some form of information that is subject to measurement noise to be a "sensor".

Much of the actual sensor fusion takes place in the evaluation of the Bayes filter. As mentioned above, we now consider (2.72) to be solved with a particle filter, where a possible system state $\boldsymbol{q}$ is represented by $\{W_{1:t}^i, \boldsymbol{X}_{1:t}^i\}_{i=1}^N$. Assuming statistical independence between the respective sensors and their models, the probability density is defined as

$$\begin{aligned} W_t^i \propto p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i) = p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{wifi}} \;\; p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ble}} \;\; p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ftm}} \\ p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{baro}} \;\; p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}} \;\; p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{act}} \;\; . \end{aligned} \tag{3.1}$$

Here, each component refers to a probabilistic sensor model. $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}}$ is a slight exception. As we will see later, the movement model used in the transition should be given the opportunity to manipulate the probability density of the evaluation. In the case of an implementation with the CONDENSATION particle filter, the transition is used as a proposal distribution,

which only allows the sampling of new particles, but no operations on the particle weight at time t. The integration of $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}}$ can thus be seen as a workaround to not violate the probabilistic foundation on the theoretical side.

The other sensor models are straightforward. Absolute position information is given by either $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{wifi}}$ for Wi-Fi RSSI, by $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ble}}$ for BLE beacons and by $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ftm}}$ for Wi-Fi FTM distance measures. Using $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{baro}}$ the barometer evaluates the current floor and $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{act}}$ the activity of the pedestrian. Assuming statistical independence between the models, the respective densities can be easily multiplied and normalized to 1.

Of course, statistical independence between all these models is a strong assumption in this case, since radio technologies in particular may exhibit signal interference at the physical level. However, modeling these effects requires sophisticated physical models that need detailed information about the building, such as material coefficients and wall properties. Even if the benefits would justify the effort, many of the physical effects are only detectable under laboratory conditions and are unsuitable for real-time systems due to the extreme computational complexity as well as the random dynamics of a system, e.g. the presence of other people.

The modularity required for a sensor fusion system is well described by the equation (3.1). For example, if a sensor fails or the user's smartphone does not have a barometer, the probability density of the sensor model is simply set to 1.0. This approach not only increases the robustness of the overall ILS, but also allows for complex scenarios such as switching between buildings that use different technologies. It is important to note that all of these sensor models are of course probabilistic in nature. This means that the occurring sensor noise and uncertainties are modeled by some kind of probability density function, e.g. Gaussian or von Mises-Fisher distribution. In the following, we present several such models that are relevant to this work.

3.1 Wi-Fi and Bluetooth

Depending on the scenario in which the ILS is used, there may be different requirements for the sensors. In the case of self-localization for pedestrians, e.g. finding one's position in a shopping mall, the main goal for the provider is to attract as many users as possible to its system while keeping costs low. The obvious solution is to use existing infrastructure and widely available technologies. Furthermore, the solution offered must meet the user's expectations based on already established outdoor applications such as Google Maps or OpenStreetMap. This results in the need to develop an ILS that works without much configuration and special hardware.

Wireless technologies such as Bluetooth and Wi-Fi are particularly well suited to such a scenario. Both can be used with off-the-shelf receivers (e.g., smartphones), and in the case of Wi-Fi, many buildings today have a well-covered infrastructure of Wi-Fi transmitters (access

points). Thanks to small, inexpensive Bluetooth transmitters, such an infrastructure can be installed quickly and inexpensively [Fet18]. To locate itself, the receiver performs a network scan, identifying all nearby transmitters and measuring the intensity of the received signal. The so-called *received signal strength indication* (RSSI) can be used to derive an estimate of the distance between transmitter and receiver. Its value is usually expressed in dBm (decibels relative to one milliwatt) and quantified in whole numbers. A higher RSSI value generally indicates a stronger signal, with values close to $-30\,\mathrm{dBm}$ indicating an exceptionally strong signal and values closer to $-90\,\mathrm{dBm}$ indicating a weak or very weak signal. So the higher the RSSI value, the stronger the signal and the closer the transmitter is likely to be to the receiver. However, as we will see in the next section, other factors such as interference, obstacles, and the number of clients connected to the access point will affect the overall performance of the wireless network.

3.1.1 Technical Principles

A transmitter is identified by some unique identifier. For Wi-Fi, this is primarily the MAC address, as specified in the broad family of IEEE 802 networking standards [Noab]. In the case of Bluetooth, the MAC address could also be used, but a better option has been available for several years. With the introduction of version 4.0, the standard was split into two main types, each with its own disjoint protocols: Classic Bluetooth (CB) and *Bluetooth Low Energy* (BLE) [FH15]. While the focus of CB is to provide two-way communication between devices, such as headphones or cars, BLE reduces power consumption and allows for one-way communication. These one-way devices are called beacons, and are specified under the BLE proximity subprofile. Their sole purpose is to broadcast advertising packets, consisting of an identifier and a few bytes of arbitrary data, to nearby devices. Beacons are therefore inexpensive to manufacture, consume very little power, allowing temporary battery operation, and can be detected by any standard smartphone. In addition to simple location-based use cases such as proximity marketing, these characteristics make them well suited for ILS. Google and Apple have also recognized this, which is why the former launched the Eddystone beacon protocol [Noaa] in 2018 and the latter pioneered iBeacon [Inc] in early 2013.

BLE operates exclusively in the *radio spectrum reserved internationally for industrial, scientific, and medical* (ISM) use between $2.402\,\mathrm{GHz}$ and $2.480\,\mathrm{GHz}$. It uses 40 $2\,\mathrm{MHz}$ channels which transmit data via Gaussian frequency shift keying modulation at a bit rate of $1\,\mathrm{Mbit/s}$ per channel. Only 3 channels are defined for advertising, while the others act as data channels. These advertising channels have been assigned center frequencies that minimize overlap with Wi-Fi channels 1, 6, and 11, which are commonly used because they are non-overlapping

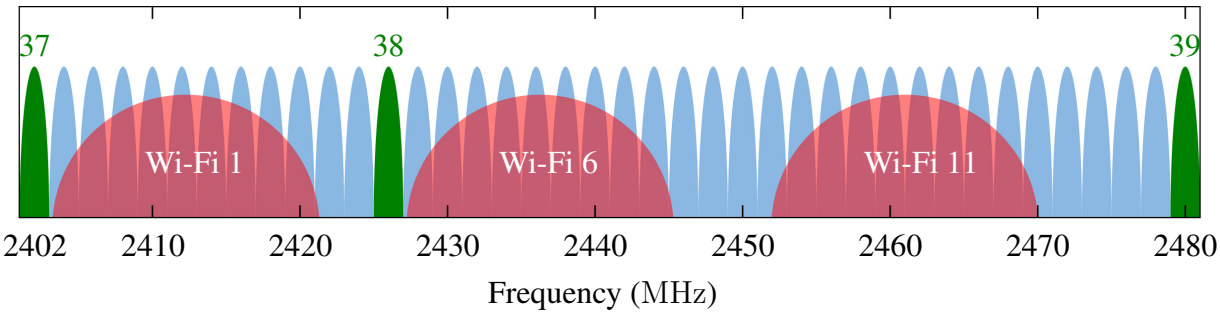


Figure 3.1: The radio band between a spectrum range of $2.402\,\mathrm{GHz}$ to $2.480\,\mathrm{GHz}$ shared by *Bluetooth Low Energy* (BLE) and Wi-Fi channels 1, 6 and 11. The advertising channels are highlighted in green, the data channels are colored blue. It can be seen, that the advertising channel frequencies are chosen to avoid the Wi-Fi channels in red.

[GOP12]. In Figure 3.1 it can be seen that the advertising channel frequencies are chosen to avoid the dominant Wi-Fi channels. To achieve redundancy, each advertisement is broadcast on each of the three channels in quick succession [PGH17].

Wi-Fi is a trademark of the non-profit Wi-Fi Alliance and uses many parts of the IEEE 802 family of protocols. The versions of Wi-Fi that have appeared regularly since 1997 are specified by several IEEE 802.11 protocol standards. These include 8 radio frequency bands, of which $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$ are the most common. Similar to BLE, these bands are divided into channels. From $2.401\,\mathrm{GHz}$ to $2.495\,\mathrm{GHz}$, there are 14 channels with a separation of $5\,\mathrm{MHz}$ and $20\,\mathrm{MHz}$ channel width. Since Wi-Fi is highly regulated worldwide, only channels 1 to 11 are allowed in most countries. To avoid interference from crossed channels, a building's Wi-Fi infrastructure is often configured to avoid them, resulting in the channels shown in Figure 3.1. Under heavy usage, providing only the 3 channel can cause interference, resulting in significant slowdowns.

In contrast, at $5\,\mathrm{GHz}$, the frequency band is not uniformly regulated worldwide. In Europe and the USA, frequencies from $5.15\,\mathrm{GHz}$ to $5.35\,\mathrm{GHz}$ with channels from 36 to 64 and frequencies from $5.5\,\mathrm{GHz}$ to $5.825\,\mathrm{GHz}$ with channels from 100 to 165 are commonly used. By choosing a channel width from $20\,\mathrm{MHz}$ to 24, the channels are free of overlap. With the channel bonding process introduced in 802.11n, widths of 40, 80 and 160 are possible. Of course, a larger channel width increases throughput, but also reduces the number of non-overlapping channels. RSSI values can be significantly affected by the level of interference in an environment, e.g. due to rapidly increasing packet loss [Gar17]. It is advantageous for an ILS to have a higher number of non-overlapping channels.

Wi-Fi scans are passive, as the transmitter waits for a *service set identifier broadcast* (SSID) from each nearby access point. This limits the rate at which RSSI measurements can be taken. With over 50 WiFi channels available in the $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$ bands, and a typical SSID

broadcast interval of $100\,\mathrm{ms}$, a single scan can take several seconds, resulting in a very low update rate. In contrast, most commercially available BLE beacons focus on localization and therefore offer multiple transmission modes starting at very high advertising rates of $50\,\mathrm{Hz}$ [GOP12].

In terms of range, consumer hardware using $5\,\mathrm{GHz}$ can only cover relatively small ranges of up to $50\,\mathrm{m}$, while $2.4\,\mathrm{GHz}$ can reach up to $100\,\mathrm{m}$. The range is reduced even further in rooms with strong attenuation. Reinforced concrete usually shields the $5\,\mathrm{GHz}$ signal completely. Of course, from a more technical perspective, the achievable range to still receive RSSI measurements depends on the respective transmit power. For both Wi-Fi and BLE there are enterprise solutions ranging from a few meters to several hundred meters [Bea; Zen22]. The choice of which range makes sense for subsequent localization purposes depends on the scenario. In open factory halls, long ranges may be useful to keep the number of devices low, while in office buildings a "much helps much" paradigm is more applicable due to the many NLOS scenarios.

In addition to channel interference, Wi-Fi and BLE are affected by path loss and multipath as their signal travels through free space and obstacles. In general, path loss describes the loss of signal strength along the path of the radio waves. This can be due to many effects that cause signal attenuation, such as shadowing, reflection, refraction, diffraction, absorption and scattering [Rac11]. They are more prevalent inside buildings due to the large number of NLOS transmissions. In particular, reinforced concrete and metallized glass have an even greater amplification factor. Some of these phenomena can be simulated quite reliably for simple scenarios using radio propagation models [DMAZC20]. However, large buildings with partly unknown construction materials as well as strongly varying material properties pose great challenges to these models [Ebn17]. In this context, the human factor, which can be considered as a dynamic absorbing object, can hardly be modeled in real time.

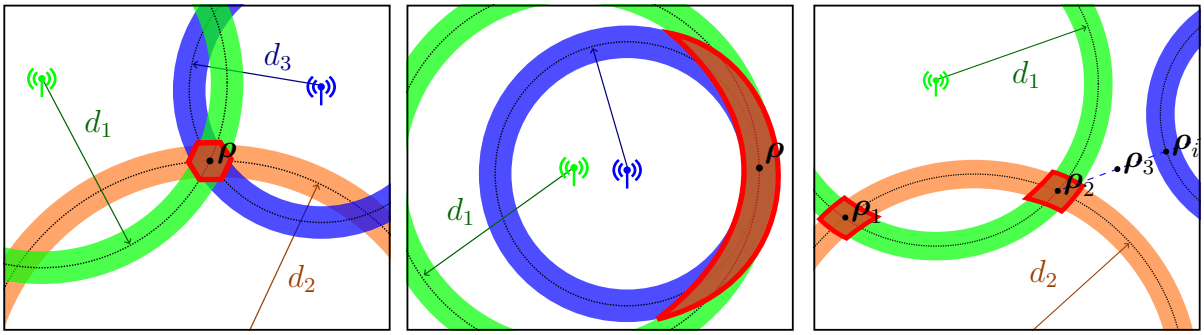
Reflection and scattering can cause multipath. This is when the receiver measures numerous combinations of multiple copies of the radio signal along different paths. Even a very small distance shift of $10\,\mathrm{cm}$ can result in a $20\,\mathrm{dB}$ to $30\,\mathrm{dB}$ decrease in RSSI [FH14]. For example, a signal copy that passes through several walls will reach the receiver first, resulting in a lower RSSI value, while another signal that is reflected several times and not absorbed on its way will result in a higher RSSI value. In the literature, the problem is usually solved by increasing the number of transmitters, adding more antennas per device, or filtering over multiple measurements [Wis19].

3.1.2 Basic Location Estimation

As discussed in Section 1.2.1, many authors use RSSI measurements for ILS using either a (true-range) multilateration or a fingerprinting approach. In the simplest case, to obtain a position in fingerprinting, an initially recorded radio map is compared to current measurements using a lookup table and nearest neighbor classification [BP00]. This method incorporates environmental features into the previously recorded fingerprints. Therefore, the structure of the building (e.g. walls and furniture) and the positions of the access points do not necessarily need to be known. Under optimal conditions, small positioning errors in the low single-digit meter range can be achieved. However, even small changes in the environment, such as a new workstation or a newly installed machine, can massively alter the radio map and cause large positioning errors. In addition, the accuracy depends strongly on the density of the fingerprints and on whether the smartphone used for the recordings is also used for tracking later, since the RSSI characteristics can differ greatly between individual devices.

There are countless adaptations and improvements of this concept, see the survey papers of [Hon09b], [HC16] and [KGA17]. The following basic questions are usually answered by newly proposed approaches: How to capture the fingerprints, which radio map (database) design to choose, and how to perform the final position estimation? Especially for large buildings, fingerprinting is a time-consuming and tedious process that needs to be repeated whenever the environment changes significantly. Crowded-sourced approaches have been proposed in the literature that try to work without any prior knowledge or at least to keep existing radio maps continuously updated [Jia16]. For this purpose, simpler ILS systems are usually used to obtain a position for the fingerprint recordings. Of course, this position is subject to errors that must be taken into account when creating the radio map. Crowdsourcing promises to reduce this error on average due to the very large number of measurements. Another well-established approach is to use robots to capture fingerprints, but these are mostly limited to flat surfaces [Mir14].

To obtain a position, multilateration uses geometric principles based on the distance between the current unknown position and multiple spatially separated known reference points. The mathematical idea is simple: If you only know the distance to one known point, your own location (when viewed in a plane) is on a circle, in 3D space on a spherical shell around that point. If there are three known points, the location is on the intersection of the three spherical shells, i.e. on a circle. This is commonly known as trilateration; if more than three distances are available, it is called multilateration. Due to the presence of noisy and imperfect measurements, the circle considered in the 2D case is more like a donut, as Figure 3.2 shows. The reference points, e.g. BLE beacons, form a common intersection or confidence area. In the most optimal constellation, the angle of intersection is nearly orthogonal, which makes the area small,



(a) (b) (c)

Figure 3.2: Trilateration scenarios as a geometric solution for position estimation. The distance measurements of the transmitters are given by d_1, d_2 and d_3. Drawn as a dotted circle. The noise is represented by the colored outline. Intersections are marked red and positions are given by $\boldsymbol{\rho}$. (a) The transmitter circles intersect at an almost orthogonal angle, giving a small area in which the position $\boldsymbol{\rho}$ can be found. (b) Two transmitters are placed on a common line, resulting in a larger intersection area compared to an orthogonal placement. (c) Two circles intersect and one is isolated. $\boldsymbol{\rho}_3$ is the midpoint between $\boldsymbol{\rho}_2$ and $\boldsymbol{\rho}_i$.

see the red area in Figure 3.2a. The size of the area in which the position to be computed is located depends on the measurement noise, the approximation error of the propagation model, and the geometry of the reference points to each other. The larger this area, the larger the total positioning error can be expected. For a simple two-dimensional position estimation for trilateration, one could simply select the geometric center of the intersection. This is illustrated in Figure 3.2a, finding the position $\boldsymbol{\rho}$. For the complex case of three-dimensional multilateration, such an approach is of course not very suitable, since complex geometric structures are formed for the intersection area [CP17].

There are other multilateration constellations to consider. If two transmitters are on a common line, the intersection area increases, as can be seen in Figure 3.2b. In this scenario, the possible error varies depending on the direction. The narrow shape of the red area in the horizontal direction allows a stable position. In the vertical direction, however, a larger error occurs due to the large height of the area. In Figure 3.2c two circles intersect and one is isolated. This results in two intersection areas with geometric centers $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$. So we are faced with a situation where there is no single point estimate. A geometric solution to find such a point, including the isolated circle (blue), is to calculate the midpoint $\boldsymbol{\rho}_3$ between $\boldsymbol{\rho}_2$ and the intersection $\boldsymbol{\rho}_i$ given by the blue dashed line from the transmitter to $\boldsymbol{\rho}_2$. However, as the number of transmitters increases (more circles), the complexity of finding such a solution increases. Another more common approach is the least squares algorithm, which minimizes the quadratic error between the measured distance and the actual distance at a given point [Bul20b].

3.1.3 Radio Propagation Model

The distance between the receiver and the transmitter can be estimated using a radio propagation model. Such models attempt to predict the path loss a signal will encounter over distance by modeling the behavior of radio waves as a function of frequency, distance, and other conditions. Radio propagation models are empirical by nature, so they can never model exact behavior, only an approximation of behavior under certain conditions and assumptions [Rac11]. As described above, indoor environments are highly affected by multipath and signal attenuation. Therefore, different propagation models have been developed for indoor scenarios.

The well-known *log-distance model* accounts for obstacles between the transmitter and receiver by attenuating the signal with a constant, empirically chosen path loss exponent γ, given by the environment [Rap96]. However, when the radio signal passes through several different objects on its way to the receiver, such as a glass door, drywall, and concrete wall, a single global value is not sufficient to describe the signal. In a localization scenario, the pedestrian moves through the building, which means a constantly changing combination of objects between the transmitter and receiver. Adding a *wall-attenuation-factor* (WAF) solves this problem by considering all objects blocking the line of sight and attenuating the signal for each by a constant factor. It can be described as

$$P_{\text{waf}}(d) = \underbrace{P_0 - 10\gamma \lg \frac{d}{d_0} + \mathcal{X}_\sigma}_{\text{log-distance model}} + \underbrace{\sum_{i \in I} \boldsymbol{\phi}_i}_{\text{WAF}} \ , \tag{3.2}$$

predicting the received signal strength from a transmitter at a given distance d. A transmitter can thus be described by a set of parameters $(\boldsymbol{\varrho}, P_0, \gamma, \boldsymbol{\phi})$, where $\boldsymbol{\varrho} = (x, y, z)^T$ is the position. Given the position $\boldsymbol{\rho} = (x, y, z)^T$ for which the signal strength is to be estimated, the distance can be easily obtained by $d = ||\boldsymbol{\varrho} - \boldsymbol{\rho}||$. The reference signal strength P_0 is measured at a known distance d_0, usually $1\,\text{m}$ [SR92]. To model slow fading and random channel noise, a zero-mean Gaussian $\mathcal{X}_\sigma$ with a predefined standard deviation is also given [KGA17]. The local attenuation factors are described by $\boldsymbol{\phi} = (\phi_{\text{wood}}, \phi_{\text{concrete}}, \phi_{\text{glass}}, ...)$ and are either taken from the literature or have to be measured on site. Thus, any obstacle I intersecting between $\boldsymbol{\varrho}$ and $\boldsymbol{\rho}$ reduces the signal strength based on ϕ. The associated intersection test is very expensive, especially for large and complex buildings [Ebn15]. Although (3.2) does not account for multipath effects, the absorption and shadowing of walls and other intersecting objects can be reasonably modeled given the semantics of the building. Again, this requires knowledge of the building's materials and obstacles, as well as sufficient computing power to perform the intersection tests.

To compensate for this, a compromise between the log-distance model and the wall attenuation factor model was presented in [Ebn17], which only includes ceilings. Of course, ignoring walls reduces the accuracy of the model's prediction, but reduces the performance impact when used on smartphones, while being sufficient for many use cases [Fet17]. The *ceilling-attenuation-factor* (CAF) is added to the log-distance model as follows:

$$P_{\text{CAF}}(d) = \underbrace{P_0 - 10\gamma \lg \frac{d}{d_0} + \mathcal{X}_\sigma}_{\text{log-distance model}} + \underbrace{\Delta f \beta}_{\text{CAF}} \ , \tag{3.3}$$

where Δf is the number of floors between $\boldsymbol{\varrho}$ and $\boldsymbol{\rho}$. The attenuation per floor is given by β. For example, a good choice for steel reinforced concrete floors is $\beta \approx -8\,\text{dB}$ [Ebn17]. So the description for a single transmitter changes to $(\boldsymbol{\varrho}, P_0, \gamma, \beta)$. These parameters must be known in advance for each installed transmitter. For simplicity, it is common to use the same empirically chosen values for all stations. This may provide enough accuracy for some use cases and buildings, but fails in complex scenarios, such as a historic building or an industrial workshop [Fet18; Fet23b].

3.1.4 Probabilistic Location Estimation

Unlike log-distance, (3.2) and (3.3) are not reversible due to the intersection tests required. Thus, they cannot be solved for a distance d, as the geometric considerations of the multilateration approach would require. In addition, solving the problem of position determination with multilateration in three-dimensional spaces with complex geometric structures is a difficult task, as described above. This can be well accounted for by a probabilistic approach that describes the estimated position in terms of probability density functions. Such an approach allows the uncertainty of the measurement error to be modeled by probability. The probability of observing a measurement under the condition of a possible position in the building is expressed. Thus, for each RSSI measurement, a hypothesis is made about how it relates to the RSSI estimate between the known transmitter and the potential position by a propagation model. The combination of all hypotheses results in a joint density function describing the most likely location of the pedestrian. As discussed in [Bul20b], finding a position estimate is not the intersection of spherical shells, but the position with the highest probability.

With all of the above, our sensor models $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{wifi, ble}} = p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{wifi}}$ or $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ble}}$ can be described using a probability density. Given radio measurements $\boldsymbol{s}_{\text{wifi, ble}}$ corresponding

to a known location provided by $\boldsymbol{\rho}$, it can be written as

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{wifi, ble}} = p(\boldsymbol{s}_{\text{wifi, ble}} \mid \boldsymbol{\rho}) = \prod_i \prod_m p(s_{i,m} \mid \boldsymbol{\rho}) \ , \tag{3.4}$$

where $\boldsymbol{\rho} = (x, y, z)^T$ is the known position of the particle and $s_{i,m}$ denotes a RSSI measurement from transmitter i with count m of M_i for a particular evaluation step of the particle filter. We assume statistical independence between each transmitter i, which results in a joint density function by multiplying the respective densities. Since the number of measurements M_i can differ between the transmitters, a normalization should be performed for each iteration step i. As discussed at the beginning of Chapter 2, statistical independence between transmitters with similar technological behavior is a strong assumption, especially when we consider attenuation and multipath effects. However, without appropriate inference models that can also be efficiently computed in a real-time pedestrian ILS, this assumption arises automatically.

To provide a result for $p(s_{i,m} \mid \boldsymbol{\rho})$, we use the Gaussian noise of the propagation model $\mathcal{X}_\sigma \sim \mathcal{N}(...)$ as the probability density function:

$$p(s_{i,m} \mid \boldsymbol{\rho}) = \mathcal{N}(s_{i,m} \mid P_{\text{CAF}}(\dot{d}_i), \sigma^2_{\text{wifi,ble}}) \ . \tag{3.5}$$

Here $\dot{d}_i$ is the distance between the transmitter position $\boldsymbol{\varrho}_i$ and the particle position $\boldsymbol{\rho}$. The noise for Wi-Fi or BLE is then modeled using either σ_{wifi} or σ_{ble} as the standard deviation. In simple terms, the formula (3.5) represents a comparison between the RSSI measurements of the receiver and the estimate of the propagation model (3.3) that should be measurable at the position $\boldsymbol{\rho}$. The better the measurement and the estimate match, the more likely this position, i.e. this particle, is. While multilateration fails if not enough measurements are available, the probabilistic approach can still generate a hypothesis. Surprisingly, the geometric considerations of multilateration still apply to this model, since the highest probability is observed where the individual density functions overlap. Visualized on a two-dimensional plane, each RSSI measurement of a transmitter generates a ring-shaped probability density, which in cross section represents the Gaussian distribution. Consequently, it is not always possible to obtain a single point estimate, as there are situations where many positions could be equally likely [Bul20b].

The assumption of a Gaussian distribution for modeling the sensor noise caused by attenuation and multipath effects is controversially discussed in the literature. While many ILS assume a Gaussian distribution and obtain good results [Dic16; Fet18; Nur14], several other authors confirm that while these assumptions are appropriate for simplicity, RSSI measurements do not follow a Gaussian distribution, especially due to multipath effects [Lad05; RM12; Tan11]. In [KK12], the authors investigated the RSSI error for indoor environments under LOS and NLOS

conditions. They show that more than $70\,\%$ of the distribution is left-skewed, and they conclude that signals with stronger power and line of sight often have strongly left-skewed histograms. However, since both propagation models do not include multipath effects, we also assume a simple Gaussian.

3.1.5 Parameter Optimization

The transmitter parameters $(\boldsymbol{\varrho}, P_0, \gamma, \beta)$ must be known in advance. In most buildings, they vary widely from transmitter to transmitter. Especially in very complex scenarios such as historic buildings, which are often built of massive stone walls and have additions from different historical periods made of different materials, empirical choice of values may fail [Fet18]. A fingerprinting approach has the advantage that no knowledge of the parameters is required. All relevant information is contained in the RSSI of the fingerprint. Conversely, this means that statements about the transmitters can be made based on measurements at known locations. If a set of reference measurements $\boldsymbol{s}_{\text{opt}}$ is recorded throughout the building, an optimization scheme can be applied to find suitable parameters per transmitter. Unlike fingerprinting, where reference measurements are taken on small grids between $1\,\mathrm{m}$ to $2\,\mathrm{m}$, it's sufficient to take measurements every $3\,\mathrm{m}$ to $7\,\mathrm{m}$ in the center of a corridor and between 1 and 4 references per room, depending on its size. This significantly reduces the number of references required and thus the total cost of installation and maintenance [Ebn17].

To optimize the 6 model parameters for a single transmitter, the target function reduces the squared error between reference measurements $s_i \in \boldsymbol{s}_{\text{uuid}}$ with well-known location ρ and corresponding model predictions $P_{\text{waf}}(d)$ (cf. equation (3.2)). It can be written as

$$(\boldsymbol{\varrho}, P_0, \gamma, \beta) = \underset{\boldsymbol{\varrho}, P_0, \gamma, \beta}{\operatorname{argmin}} \sum_{s_i \in \boldsymbol{s}_{\text{uuid}}} \left(s_i - P_{\text{waf}}(\|\boldsymbol{\rho} - \boldsymbol{\varrho}\|)\right)^2 \ , \tag{3.6}$$

where $\boldsymbol{s}_{\text{uuid}}$ is the subset of $\boldsymbol{s}_{\text{opt}}$ for the AP in question, identified by it's UUID, e.g. MAC address. There are several optimization strategies to solve (3.6). However, optimizing all 6 parameters results in a non-convex, discontinuous function, while optimizing only the propagation model parameters P_0, γ and β usually means solving only a convex function. Figure 3.3 illustrates this in a simplified way for the average error (in dB) between references $s_i \in \boldsymbol{s}_{\text{uuid}}$ and corresponding model predictions $P_{\text{waf}}(\|\boldsymbol{\rho} - \boldsymbol{\varrho}\|)$ for a Wi-Fi transmitter [Ebn17]. In Figure 3.3a, the position $\boldsymbol{\varrho}$ and the floor attenuation factor β are fixed, so the path loss exponent γ and the signal strength P_0 are modified using (3.6). Note that the graph is shaped like a cup $\cup$, clearly indicating a convex function. In contrast, Figure 3.3b shows the change of the y- and

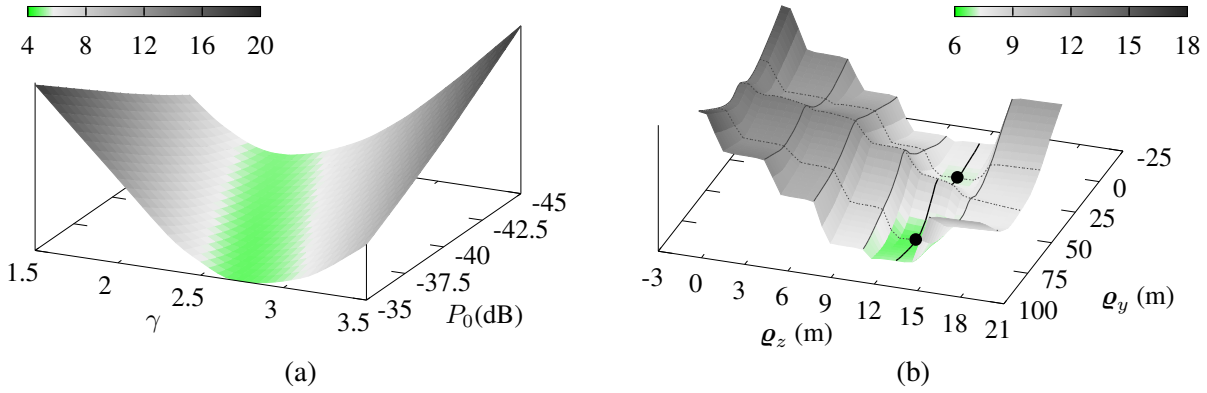


Figure 3.3: The average error (in dB) between reference measurements $s_i \in \boldsymbol{s}_{\text{uuid}}$ and corresponding model predictions $P_{\text{waf}}(\|\boldsymbol{\rho} - \boldsymbol{\varrho}\|)$ for a Wi-Fi transmitter. (a) The transmitter position $\boldsymbol{\varrho}$ is known and β is fixed, while changing the path loss exponent γ and signal strength P_0 denotes a convex function. (b) Changing the y- and z-position (in m) with fixed $\boldsymbol{\varrho}_x$, P_0, γ and β denotes a non-convex function with multiple local minima. Figure taken from [Ebn17].

z-coordinates of the transmitter position $\boldsymbol{\varrho}$. As can be seen from the two local minima, this is sufficient to produce the non-convex, discontinuous function mentioned above.

For convex functions, of course, all local minima are global minima. Thus, an algorithm like gradient descent is well suited and even able to converge in a (near) optimal way for the error function, depending of course on the chosen step size and descent direction [MBG13]. On the other hand, for non-convex functions, a gradient descent approach may be able to find a global minimum, but it cannot guarantee to find a global minimum or even a local minimum. Even worse, theoretical guarantees for non-convex functions are usually weak or sometimes non-existent. A non-convex optimization is at least NP-hard. Besides the parameter optimization discussed here, deep neural networks are another popular example of non-convex problems [JK17].

The authors of [Lia16] optimize a two-dimensional position based on distributed *multiple-input multiple-output* (MIMO) radar sensors. They use the half-quadratic and semidefinite relaxation optimization techniques to alternately determine the target position and the required auxiliary variables. Another promising approach is the use of a genetic algorithm inspired by the process of natural selection [Gol89]. In [Ebn17] and [Fet18] we presented such an approach. The genetic algorithm starts with an initial population that is uniformly sampled within predefined bounds of the parameters to be optimized. γ, P_0, and β are set within a reasonable interval around empirically chosen values. For example, for BLE beacons, $P_0 = -59\,\mathrm{dB}$ at $1\,\mathrm{m}$, $\gamma = 2$ for open space, and $\beta = -8\,\mathrm{dB}$ for reinforced concrete ceilings. The location of the transmitter is assumed to be inside the building and is therefore limited by it's size plus a few meters to the outside. The latter is necessary because the optimization tries to fit the parameters

based on the (unrealistic) signal strength prediction model by minimizing the error between measured and predicted RSSI. Since walls are ignored by the model, optimizing the position of the transmitters can compensate for effects not considered by the model. As a result, the position of the transmitters may be several meters away from the actual position. However, since the same propagation model is used in the live phase, the error is minimal. If a more accurate model were used, one would expect the optimized and real positions to converge.

Optimization is performed as follows: The genetic algorithm starts with the initial population and then retains the best $25\,\%$ for each subsequent iteration. The remaining entries are then recreated by modifying the best entries with uniform random values within $\pm 10\,\%$ of the known bounds. To further stabilize the results, the modification range is narrowed over time, which reduces the probability of accepting worse solutions. This additional step is inspired by cooling known from simulated annealing [KGV83]. After a given number of iterations, the optimization stops. To some extent, the results improve with increasing number of iterations and, more importantly, they stabilize as a result. To prevent premature convergence, it is a suitable choice to also increase the population size, with the disadvantage that this leads to more operations, i.e. slower computation.

For the error function described in (3.6), reference measurements $\boldsymbol{s}_{\text{opt}}$ are recorded throughout the building. Optimizing this results in a global model, i.e. a set of parameters $(\boldsymbol{\varrho}, P_0, \gamma, \beta)$ for each transmitter. However, since the CAF model in (3.3) does not take walls into account, erroneous RSSI measurements are to be expected for areas subject to attenuation and multipath effects, e.g. due to aluminum insulation or heavily shielded walls. Such local effects significantly obscure a global model. This can be addressed by optimizing models from a more local perspective, using only reference measurements within local boundaries. For example, in [Fet18] we used a per-floor approach using $\boldsymbol{s}_{\text{opt}}$ references associated with the corresponding floor. More detailed per-region or even per-room approaches are also conceivable [Ebn17]. For each particle $\{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$ the model then changes according to the position of the particle. In overlapping areas, such as stairs or elevators, interpolation is performed between the respective parameters of the models.

3.1.6 Limitations and Discussion

A constantly changing environment is not only problematic for fingerprinting, but also for multilateration approaches, which have to adapt and update at least the transmitters in the immediate vicinity. The same is true for the presented optimization method. Here the recalibration has to be done within the range of the transmitter. In a commercial application, this is obviously a cost issue, but a solvable problem. What is difficult, however, is the dynamic factor of people

walking around. Roughly speaking, people are walking water bags. The attenuation factor at $2.4\,\mathrm{GHz}$ is about $-15\,\mathrm{dB}$ for the human body when the signal passes right through it [Jan08]. In comparison, values between $-5\,\mathrm{dB}$ and $-8\,\mathrm{dB}$ are reported for a brick wall [Rac11]. A suggestion from the authors of [Hon09a] is to place the transmitters as high as possible so that there is little intersection with the human body on the direct path between the transmitter and the receiver. In this case, the attenuation factor can be reduced to as low as $-3\,\mathrm{dB}$. Another suggestion, especially in large crowds, is to reduce the signal power of the transmitters to a few meters in order to have fewer intersections with people in the LOS. The downside is that the number of transmitters has to increase dramatically.

In general, since the early days of indoor localization research, people have been thinking about the optimal placement and number of transmitters. Optimal in this sense means coverage that is suitable for the purpose of localization and not, as is usually the case, in the sense of providing the best possible network coverage. The authors of [Raj16] describe some practical issues that need to be considered: First, as we saw above, the coverage of a transmitter depends on the attenuation factors of the building materials. Obviously, a wooden cabin has better conditions than a subway station. Second, indoor spaces have extensive semantics that lead to different localization accuracy requirements in different areas. For example, in certain areas, room-level accuracy may be sufficient, while analyzing work processes in an industrial scenario requires accuracy of less than one meter within a large room [Fet22]. Third, physical factors and legal considerations often limit deployment. For example, it may be advantageous to place transmitters near electrical outlets or in locations where they are not exposed to potential theft.

One of the most basic approaches is to extend the concept of *geometric dilution of precision* (GDOP) to indoor environments. GDOP provides an assessment of the expected positioning performance based on the transmitter-receiver geometry [Raj16]. Lower GDOP values indicate better estimation accuracy due to the greater angular separation between the transmitters as seen from the receiver. While it allows us to evaluate the given scenario in terms of position estimation accuracy, it only considers the relative geometry of the transmitter and receiver. Effects such as signal attenuation or multipath propagation are not taken into account.

The authors of [BHE01] propose a grid-based approach that divides the indoor space into equidistant squares. They start with an initial empirical distribution of transmitters. Using a square window function, they compute the cumulative localization error, the distance between estimated and actual position, for all overlapping squares. A new transmitter is added to the center of the square window that reports the highest error. This approach is very general. The localization error can be computed using synthetic data, such as comparing the results of a radio propagation model with Euclidean distance, as well as real-world data. The latter, however, requires repeating the entire process each time a new transmitter is added.

Although these and many other approaches proposed in the literature can give good results, they only satisfy the first point of the practical problems introduced above. In particular, the third point allows for many degrees of freedom. To put this in a more technical context, it has been proved that the placement problem is NP-hard [YLY19]. The authors of [MBH17] therefore recommend two simple paradigms: First, avoid central areas and prefer exterior walls. Second, avoid placing transmitters on a common line in areas where pedestrians can walk.

As mentioned above, Wi-Fi scans are passive and can take several seconds to process depending on the available Wi-Fi channels. This results in a very low update rate for RSSI measurements [FH15]. The use of frequent active Wi-Fi scans, where the receiver sends a query, increases network traffic and thus reduces throughput and privacy. Conversely, heavy use of the Wi-Fi infrastructure by classic network users also reduces throughput and increases channel noise. Perhaps most detrimental to commercial approaches in public settings is the fact that some mobile system vendors do not provide access to the data from a Wi-Fi scan. For example, Apple's iOS only allows RSSI readings from a connected access point. This prevents any approaches using third-party or passive Wi-Fi transmitters.

Consequently, a cross-platform ILS application should rely on BLE beacons instead. However, because BLE advertisements are broadcast on each of the three advertising channels, most consumer devices do not report the channel on which a particular beacon is heard, and thus provide a composite of that beacon's RSSI readings per scan. Since the channels are widely separated in the frequency domain, the authors of [PGH17] confirmed that the RSSI values also differ per channel due to different propagation characteristics. According to them, this leads to artificially high variances of the composite RSSI values, with $80\,\%$ of the composite variances differing from the individual channel values by at least $1\,\mathrm{dBm}$ and $40\,\%$ of the composite variances being at least $4\,\mathrm{dBm}$ greater. Therefore, a per-channel approach may be useful for fingerprinting and parameter optimization.

3.2 Wi-Fi Fine Timing Measurement

As we saw in the last section, there are many steps involved in obtaining a good estimate of position from RSSI measurements. There are several propagation models that can be used to establish a relationship between distance and RSSI. The problem is that to reproduce the extreme spread (noise) of the RSSI, unrealistically complex models are needed. In the extensive study by [KK12], the RSSI of six different Wi-Fi chipsets was recorded over a period of five days. They report standard deviations ranging from $0.59\,\mathrm{dB}$ to $6.29\,\mathrm{dB}$. In non-laboratory scenarios with NLOS signals and people walking around, the variance increases further. The log-distance model can be easily rearranged so that a distance can be estimated from an RSSI

value. Under very simplified free space propagation assumptions ($\gamma = 2$, $P_0 = -45\,\mathrm{dB}$), the standard deviation changes from $0.4\,\mathrm{m}$ to $7.5\,\mathrm{m}$. This simple example shows that localization using RSSI values reaches certain limits that can only be crossed with high complexity. In the literature, sub-meter accuracy is only achieved under laboratory conditions and is not considered feasible for real-world scenarios [Bul20b; Hor20b].

In summary, RSSI was not designed with localization in mind. An alternative is to use time-based methods, sometimes referred to as time-of-flight propagation. As discussed in the introduction to this thesis, they are based on the time it took the signal to travel from the transmitter to the receiver. The distance between the two points is then obtained by multiplying the time by the propagation time of the signal. Thus, the distance is directly proportional to the time of flight. Under LOS conditions in a vacuum, the speed of the signal can be assumed to be the speed of light. Of course, inside buildings, this assumption is not valid because of walls, ceilings, and other objects through which the signal must pass. The speed of propagation of the signal depends on the relative permittivity of the interacting material. For example, a concrete wall has a relative permittivity between 5 and 7, which can cause a $5.0\,\mathrm{GHz}$ signal to take 2 to 4 times as long to travel through a wall as it would to travel the same distance through air [Hor20b]. In the previous section, we introduced the path loss exponent γ, which describes signal attenuation. It is important to understand that even if the signal is significantly attenuated, it does not directly affect the time of arrival [Wil02].

As recent results have shown, time-based distance measurements are capable of providing sub-meter accuracy because they are more robust to external interference and environmental changes compared to RSSI [Bul22; Hor20b; Oti20; Yu19; Zha06]. Several methods have been proposed in the literature to measure the transit time of a signal. The best known for indoor use are TOA, TDOA, and RTT (see 1.2.1). In TOA, the time is measured for a one-way propagation between transmitter and receiver. This requires strict synchronization of all devices involved. Depending on the underlying communication protocol, the transmitter must also add the current timestamp to the signal in order for the receiver to determine the time the signal has traveled. TDOA is a passive method that uses the difference in signal arrival times at multiple spatially separated receivers. Unlike the previous methods, the logic is reversed. There is a transmitter worn by the pedestrian that actively propagates its signal and is received by stationary receivers. Compared to TOA, only synchronization between the receivers is required. The TDOA measurement can then be given by $\mathrm{TDOA}_{i,j} = \mathrm{TOA}_j - \mathrm{TOA}_i$. Calculating a distance value by multiplying the propagation time of the signal by $\mathrm{TDOA}_{i,j}$ defines a hyperboloid of possible target locations with the two receivers as foci. Using triangulation, a system of three hyperbolic equations provides a position estimate at their intersection. In the presence of noise, solving the hyperbolic equations requires estimation methods such as nonlinear least squares

[QZL14] or an iterative algorithm that linearizes the equations based on a Tyler series expansion [Tor84].

Whether for all or a subset of devices, strict synchronization can only be ensured in a controlled environment, often requiring specialized hardware and/or some arcane network protocols. The first problem can be solved by using RTT to obtain a distance measurement. This involves sending a signal from an initial device (e.g., a smartphone) to a responding station (e.g., an access point) and vice versa. The initial device then measures the time it takes for the signal to come back minus the time it took the responding station to process the signal. The result is the total *time-of-flight* (ToF) of the signal making a round trip, and thus the distance estimate is half the ToF multiplied by the propagation speed. This eliminates the need for synchronized clocks between nodes. The second problem can only be solved by standardization. This may be one of the reasons why the FTM protocol was introduced in the IEEE 802.11-2016 specification, also known as IEEE802.11mc, in 2016 [Au16]. FTM uses time-based techniques, namely RTT, for standards-compliant Wi-Fi devices. As noted by [Bul20b] and [Hor20b], many manufacturers have begun to incorporate FTM into their devices, although implementation of FTM is optional. The potential for FTM is huge, since all that is required is a refresh of a building's Wi-Fi access points, which is necessary after some time anyway. In addition, many flagship smartphones running Android are already capable of the protocol [Hor22].

3.2.1 Two Way Ranging

In the IEEE 802.11-2016 specification, RTT is implemented using two-way ranging (TWR). Here, both devices involved act as transmitter and receiver for a single measurement. The initial signal is propagated by a device worn by the pedestrian. This device, such as a smartphone, is called an FTM initiator because it actively requests FTM measurements. An FTM responder, such as an access point, passively waits for the request. The initiator does not need to be connected to the responder, e.g. to the Wi-Fi network. The distance is calculated on the device, which helps maintain privacy.

An overview of TWR for Wi-Fi FTM is shown in Figure 3.4 for a burst consisting of three FTM exchanges. The process begins with the FTM initiator sending an FTM request to all nearby Wi-Fi access points. If an access point supports the FTM standard, it can be considered a responder. A responder now has the choice of accepting the request by sending an acknowledge frame (ACK) or rejecting it by ignoring it. In the case of acceptance, the responder stores the current time $t_{1,j}$, which represents the start of an exchange, and sends an FTM frame to the initiator. Here $t_{i,j}$ describes a timestamp, where i is the local index of the frames and j is the global index of the FTM exchange. As soon as the incoming FTM frame is measured at the

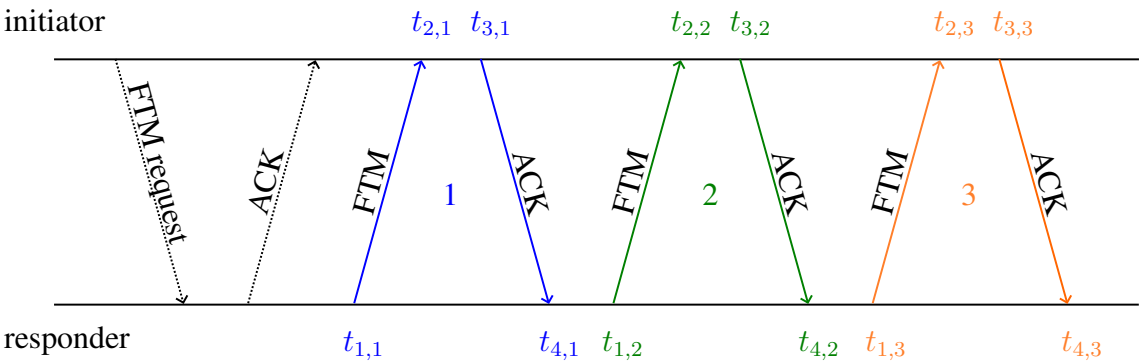


Figure 3.4: Overview of two-way ranging (TWR) used for Wi-Fi *fine timing measurement* (FTM). The burst shown consists of three FTM exchanges (FTM + ACK), distinguished by color. The process starts with the acknowledgement frame (ACK) of the initiator's first FTM request by the responder.

initiator's antenna, it stores the time $t_{2,j}$. In terms of RTT, the initiator responds with an ACK frame sent at $t_{3,j}$ and received by the responder at $t_{4,j}$. The delta between $t_{3,j}$ and $t_{2,j}$ accounts for the signal processing delay of the initiator's hardware.

To obtain an FTM measurement, the timestamps $t_{1,j}$ and $t_{4,j}$ must be sent from the responder to the initiator. In ASAP mode, the timestamps from the last exchange are sent in the next FTM frame. Thus, the last FTM exchange (e.g. orange 3 in Figure 3.4) is used only for timestamp transmission and not for measurement [HYH20]. With all four timestamps available, the ToF for a single FTM exchange can now be calculated using

$$\mathrm{ToF}(j) = (t_{4,j} - t_{1,j}) - (t_{3,j} - t_{2,j}) \tag{3.7}$$

The procedure outlined above is now repeated until the maximum number of FTM exchanges agreed upon in the initial FTM request is reached. The purpose of the repetitions is to reduce the influence of signal noise in the average. As noted in our paper [Bul20b], while RSSI is relatively easy to measure, determining accurate ToF values with a small resolution such as nanoseconds requires much more care. Even small deviations from the real time value result in a large error in the distance estimate, e.g. a measurement error of $10\,\mathrm{ns}$ results in a distance error of $3\,\mathrm{m}$. Therefore, a first step to deal with the error is to average over all N_{ftm} ToF values, given by

$$\hat{\mathrm{ToF}} = \frac{1}{N_{\mathrm{ftm}}} \sum_{j}^{N_{\mathrm{ftm}}} \mathrm{ToF}(j) \tag{3.8}$$

As reported by [Ibr18], increasing N_{ftm} will reduce the standard deviation of the measurements per burst. However, the duration of an FTM burst will also increase. On the one hand, this results in a delay between the start of the measurements and the receipt of the measured value.

On the other hand, the corresponding Wi-Fi channels will be occupied longer, which increases the probability that an FTM request will be rejected by the responder if many FTM initiators try to issue FTM requests. Furthermore, the effect of increasing the number of measurements is finite, since only random stationary additive noise is considered. That is, the error when measuring under LOS conditions, without changing the distance between initiator and responder, over a longer period of time. Measurement errors such as multipath, passing objects and other interferences are not taken into account.

With the calculation of $\hat{\text{ToF}}$ the TWR process is finished and a distance can be estimated with

$$d = \frac{\hat{\text{ToF}}}{2} \cdot c + \mathcal{E}_{\text{ftm}} \tag{3.9}$$

Despite the above discussion of relative permittivity, which reduces signal velocity in the material, we initially assume constant propagation at the speed of light c for the purpose of calculating distance d. We do this because implementations of IEEE 802.11-2016 are usually not documented by hardware vendors. In Android, neither the average $\hat{\text{ToF}}$ nor the individual ToF measurements are returned. Only the distance d and some parameters like the number of successful measurements or the initial timestamp are returned by the [Goo19] API.

To compensate, the total error $\mathcal{E}_{\text{ftm}}$ is introduced, which includes all possible error sources. Of course, this only compensates for erroneous FTM measurements per burst. However, the default number of FTM exchanges is 8 per burst, requiring a total of about $10\,\text{ms}$ to process from the FTM request to the last ACK frame. Although the update rate of individual bursts per initiator is highly dependent on hardware, channel utilization, and other magical radio factors, some authors report values between $3\,\text{Hz}$ and $5\,\text{Hz}$ per newly observed distance measurement [SVW22; Yu19]. This short period should not allow too much error variation, so we consider one error value per burst to be sufficient. Of course, 802.11-2016 allows a much finer configuration to be sent with the initial FTM request. For example, it defines 10 fixed values for burst duration, between $50\,\mu\text{s}$ and $128\,\text{ms}$. Also the number of FTM exchanges (1 to 31) within a burst, the number of consecutive bursts per session (up to $16\,384$), the burst interval (up to $655.35\,\text{s}$), the preferred bandwidth and much more can be configured. Unfortunately, these values are mostly behind a black box and, as described above, probably depend on the current channel assignment, other range requests or various other hardware limitations.

3.2.2 Measurement Error

As already seen with the RSSI, the error must first be described in order to estimate an accurate position. The error $\mathcal{E}_{\text{ftm}}$ is the general difference between the measurement and the real distance from initiator to responder. Although the FTM standard was defined in 2016, it took some time

before the first hardware and thus robust experiments were available to try to make a statement about the error. While the search term "Fine Timing Measurement" on Google Scholar returned only 36 results between 2016 and 2018, the number increased to 91 between 2018 and 2020 and now has almost 200 results. However, there is no clear consensus on which effects primarily influence the error and why.

One effect that is controversial is the relative permittivity of objects through which the signal passes. The authors of [Jio20] and [Mar14] state that the effect of relative permittivity is negligible because the total travel distance in non-air media is usually very short compared to the travel distance in air. Thus, they argue that the signal propagation speed c can be assumed to be constant for all environments. However, the validity of the experiments could be questioned. In [Jio20] they perform FTM measurements at distances from $2\,\mathrm{m}$ to $10\,\mathrm{m}$ at intervals of $2\,\mathrm{m}$, using up to 3 people as an obstacle between the initiator and the responder. The author of [Hor20b; Hor22] claims exactly the opposite. He performed FTM measurements in a three-story wooden house and a large open-plan office building, both under NLOS conditions. The data show that the relative permittivity of the building materials biases the measurements with increasing distance and permittivity. He concludes that materials with high relative permittivity significantly slow down the signal.

Both experiments discussed use a channel with $80\,\mathrm{MHz}$ bandwidth at $5\,\mathrm{GHz}$ band. As shown by [Ibr18] and [Hor20b], a higher bandwidth allows for more accurate distance measurements. As mentioned in Section 3.1, in the $2.4\,\mathrm{GHz}$ range, the channel bandwidth can be set to $20\,\mathrm{MHz}$ or to $40\,\mathrm{MHz}$, which halves the number of channels available. This results in a sampling rate of one sample every $50\,\mathrm{ns}$ or $25\,\mathrm{ns}$. In the $5\,\mathrm{GHz}$ band, a channel bandwidth of $20\,\mathrm{MHz}$, $40\,\mathrm{MHz}$ or $80\,\mathrm{MHz}$ can be set. Again, this reduces the total number of available channels. For $80\,\mathrm{MHz}$ this results in six non-overlapping channels (42, 58, 106, 122, 138, and 155) and a sampling rate of one sample every $12.5\,\mathrm{ns}$. Assuming that the receiver detects the signal on the first sample of the preamble, the smallest possible resolution of the distance estimate is $15\,\mathrm{m}$ at $20\,\mathrm{MHz}$, $7.5\,\mathrm{m}$ at $40\,\mathrm{MHz}$, and $3.74\,\mathrm{m}$ at $80\,\mathrm{MHz}$ [Bul20b].

Especially in multipath scenarios, two or more signals with a separation smaller than the raw resolution must be distinguished. It may be tempting to simply select the first signal because the direct path is supposedly faster. This can be done by improving the raw resolution using super-resolution spectral signal processing techniques such as MUSIC, ESPRIT, or Pencil Matrix [Ge04; HH09; LP04; Not18]. In general, they attempt to interpolate between known samples of the signal by making certain assumptions about the transfer function of the communication channel. Exact implementation details of what algorithms are used in the hardware are of course not available, so again we have to assume a black box. Ascertaining the original signal may work under LOS conditions, but since the signal propagates at the speed of light,

taking into account channel noise, permittivity, etc., it may be subject to interference patterns of multipath reflections. The author of [Hor20b] attributes the "position-dependent error" to this effect. He observed how even the smallest movements of the responder or initiator of a few millimeters caused an error in the distance measurement of up to $2\,\mathrm{m}$. Another observable effect of multipath propagation is that two similarly reflected signals (same length and phase) constructively interfere at the receiver, resulting in a higher received signal power compared to a direct signal. This makes it difficult to distinguish the direct path from the reflected path. In summary, multipath effects are a major source of error in FTM measurements. The accuracy of the measurement therefore depends on whether the direct path or the reflected path can be measured without further interference from multipath effects.

In general, increasing the signal bandwidth improves the accuracy of FTM distance measurements, especially in multipath environments. According to [Hor22], using a bandwidth of $80\,\mathrm{MHz}$ or in the future even higher bandwidth of $160\,\mathrm{MHz}$, the effect of relative permittivity of building materials on the accuracy of distance measurements is higher than that of multipath. Similar to the above, this statement is rather controversial, especially among those authors who assign a rather lower importance to relative permittivity and focus on multipath, such as [Bar22; Gen20; HYH20; Ibr18; Jio20]. In our opinion, this discussion lacks standardized and reproducible experiments with similar hardware components and identical setups, as well as a clear distinction between the frequency spectra and bandwidths used. As already seen with RSSI, a signal propagating at $2.4\,\mathrm{GHz}$ behaves completely different when passing through material than at $5\,\mathrm{GHz}$. The channels used, especially in the wide $5\,\mathrm{GHz}$ band, can also make a significant difference, as their behavior depends on the environment and will be different in different situations. For example, in [Hor22] it was shown that the weighted combination of three and six channels significantly reduced the standard deviation of the measurements.

The above sources of error can be considered more or less due to environmental effects. Another point to consider is a constant offset between measured and actual distance caused by the hardware itself. The offset depends on the type of initiator, the type of responder, the channel used, the bandwidth, the preamble and other hardware specific delays [Hor20b; Ibr18]. Depending on the combination of devices, it can range up to several meters, e.g. we observed an offset of $36\,\mathrm{m}$ using the Espressif ESP32v2 microcontroller. It can even change when operating on different Wi-Fi channels. Think of access points that automatically change the channel depending on the load. Therefore, factory calibration seems to be impossible, at least there are no reports of pre-calibrated hardware in the literature. This requires calibration for each combination of initiator and responder. This can be done either by canceling the offset by the delay of sufficiently long antenna cables or by measuring the offset and taking it into account in the position estimation software [Bul20b; Bul22; Ibr18]. For a freely available ILS appli-

cation, e.g. localization in a museum, this means that there must be some kind of calibration database, including the installed infrastructure of the responder (access points) and all possible commercially available smartphones. Considering the number of different Android devices on the market, this seems to be a huge task.

3.2.3 Data Driven Model

As the above discussion has shown, only very vague statements about the error can be made at this point. An exact analysis or even a physical description via radio models remains open. However, in order to provide a position estimate based on FTM measurements, it is necessary to have a way to account for the error. A straightforward approach, similar to the WAF model (cf. (3.2)), would be to directly manipulate the propagation speed in (3.9) by reducing c based on the relative permittivity at intersecting obstacles and letting $\mathcal{E}_{\text{ftm}}$ represent an empirically chosen constant offset. However, this again requires expensive intersection tests and, more importantly, accurate relative permittivity values of the building materials, since even small measurement errors in the time domain lead to large errors in the distance estimation.

A simpler and more straightforward method is data-driven models. They are based on systematic error analysis of recorded data, preferably in as many different scenarios as possible. For example, recording on an equidistant grid throughout the building, or placing devices in a fixed position and recording over a period of hours or even days. Several authors have performed such analyses [BSA16; Bul20b; Bul22; Gen20; Ibr18; Jio20]. Looking at the resulting data sets and comparing them to the ground truth, the following observations can be made: First, the measurement deviates further from the ground truth as the distance between responder and initiator increases, i.e. the error does not behave monotonically with distance, and thus large outliers that are off by several meters become more likely at larger distances. Second, the error distribution is non-linear, non-Gaussian, and often not even unimodal. Third, as discussed earlier, a device-specific constant offset biases the distance. Fourth and finally, FTM measurements are asymmetrically distributed, as they tend to be larger instead of smaller than the actual distance, i.e. closer to the lower bound.

An example data set recorded in our paper [Bul22] can be seen in Figure 3.5a. To illustrate how the FTM measurements relate to the actual distance, about 73.000 FTM measurements were collected to 15 different access points on a $1\,\mathrm{m}$ equidistant grid using a smartphone as the initiator. The data set consists of $67\,\%$ NLOS measurements at 140 locations. It has been collected under realistic conditions so that as many errors as possible can occur. Most of the above observations regarding the error can be seen in Figure 3.5a. The device-specific offset and the non-monotonic behavior with increasing distance are clearly visible. The asymmetry of

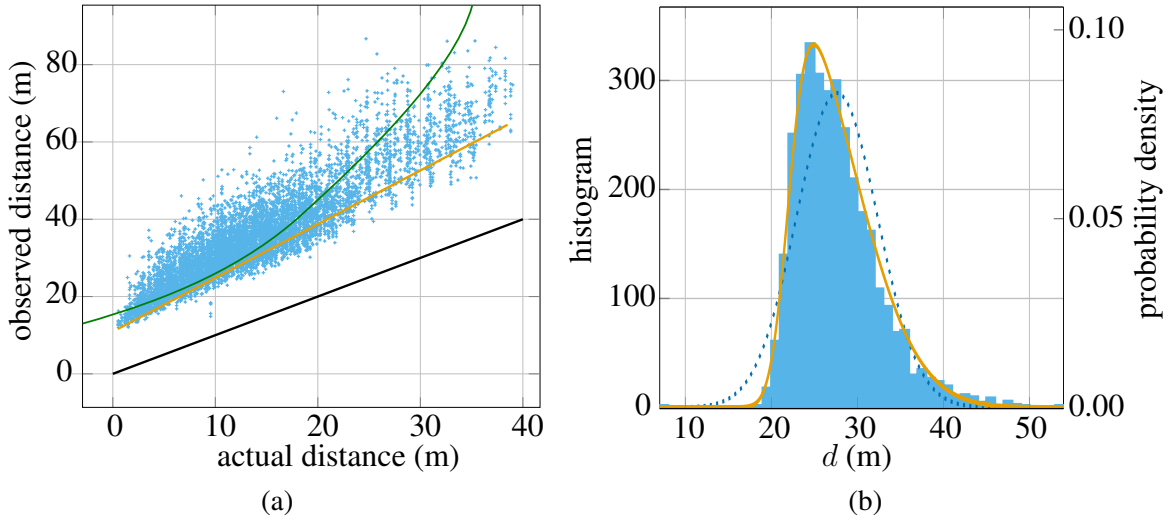


Figure 3.5: Analysis of FTM measurements based on [Bul22]. (a) Observed distances relative to the actual distance (blue), the ground truth (black), the inverse of a fitted linear-log regression model (green), and the fitted estimated location parameter of the skew normal distribution (orange). (b) Distribution of FTM values between the actual distance from $6.5\,\mathrm{m}$ to $7.5\,\mathrm{m}$. The skew normal distribution (orange) is able to represent the distribution where a standard Gaussian (dashed blue) cannot account for the skewness.

the error distribution can also be seen, as the observed distances become more spread out with increasing actual distance (>$20\,\mathrm{m}$). The non-Gaussian properties of the error can be seen in the Figure 3.5b. Here all observed distances with an actual distance between $6.5\,\mathrm{m}$ and $7.5\,\mathrm{m}$ are grouped into a $1\,\mathrm{m}$ wide bin and plotted as a histogram. The resulting distribution does not fit a Gaussian (dashed blue), but can be represented by a skew normal distribution (orange). The latter can have its mode at $24.5\,\mathrm{m}$, where the histogram has the most values, while the Gaussian has its mode at $27.6\,\mathrm{m}$. In addition, as the name implies, the skew normal distribution is able to model the skewness (steep tail to the left and long tail to the right of the mode) present in the data.

Based on these findings, several models have been presented in the literature to account for the error and obtain a position estimate. For example, [Hor20a] uses the Bayesian grid update method for estimation. Similar to particle filters, the possible location of the initiator is described by a conditional probability of observing a reported distance given an actual distance, called the observation model, and a state transition to model the dynamics of the system. The observation model is based on a data-driven approach that calculates the ratio of the observed distance to the actual distance. This reduces the Figure 3.5a relationship from two dimensions to one. On a logarithmic scale, this can be fitted with a piecewise defined exponential function, which the authors call a “double exponential function”. This function is defined for actual

distances from $1\,\mathrm{m}$ to $20\,\mathrm{m}$ in $1\,\mathrm{m}$ increments, resulting in a joint probability function of 20 such a "double exponential function" to provide a probability for an observed distance. While the estimation results are promising, the drawback of such an approach is the large amount of training data required to fit the exponential functions. Also, changes in the building, similar to fingerprinting, require re-training of the model.

Another popular approach is to use an *artificial neural network* (ANN). The authors of [Dvo19] present a network consisting of two identical ANNs for the initiator and the responder, estimating the first-path delay as output. In the first stage of optimization, the input layer receives simulated *channel state information* (CSI) in the frequency domain and is trained until the network converges with the simulated channels. This step speeds up convergence when using real CSI in the second phase, and the use of prior information facilitates the regularization of the network. The special feature of this approach is that it works in the frequency domain and can therefore be used directly on the hardware. However, this is also the biggest disadvantage, as the CSI values are only given at the hardware level and are therefore again in a black box.

A more straightforward ANN-based approach was presented by [Hua20]. They collected FTM distance values as fingerprints at the corners of an equidistant grid throughout the building. These fingerprints, including the directional information of the smartphone's pose, the responding access point positions, and some statistical values, are fed into the input layer of the back-propagation trained network. The output is then the position in the building that best matches the fingerprint pattern. To obtain further position estimates, the network is integrated into a particle filter using a map-aware PDR for state transitions. This approach gives very promising results in terms of positioning accuracy, but has the well-known high overhead of a fingerprinting solution.

However, most of these models are quite complex and require extensive calibration phases. In scenarios where time is a critical factor, simpler options are needed. For this reason, we have developed a very simple data-driven model based on log regression analysis [Fet23a]. Here we try to fit a function of the form

$$\hat{d}_{i,m} = a + b \cdot \ln(d_{i,m}) \ , \tag{3.10}$$

where $d_{i,m}$ is a measured FTM distance for an access point i with count m of M_i. The regression coefficients a and b describe the relationship between $d_{i,m}$ and $\hat{d}_{i,m}$. Therefore, $\hat{d}_{i,m}$ is the resulting error and offset corrected value, an approximation of the actual distance $\dot{d}_i$ between initiator and responder.

A function of the form similar to 3.10 is also known as a linear-log model. The idea behind using a logarithmic representation is, as mentioned earlier, that the error becomes more

scattered with increasing distance. Thus, we assume that at larger distances between initiator and responder, the measurements are more likely to be too large than too small. To find a fitting model, only a single random training walk through the entire building is required. In the best case, the initiator had multiple interchanges with each access point and the path goes through most of the building. Of course, the ground truth of the path must be known and is then compared to the measured distance over time. In this case, $\hat{d}_{i,m}$ is assumed to be the ground truth.

The green line in Figure 3.5a illustrates a linear logarithmic model fitted to the [Bul22] dataset. Due to the reversed axes, the inverse of the model, which is obviously an exponential function, was chosen for visualization. Thus, using the actual distance as input results in an estimate of an observed distance. It can be seen that the function fits quite well between $5\,\mathrm{m}$ and $20\,\mathrm{m}$, but fails to model larger distances. In addition, observing distances smaller than $17\,\mathrm{m}$ can still lead to negative estimates. In summary, this model may work in some limited cases, but will fail in large realistic scenarios. In our opinion, the only advantage of this model is its simplicity, which makes it extremely quick to implement. Nevertheless, this model certainly has its raison d'être and its use cases.

To put all this into a probabilistic context for sensor fusion, i.e. a sensor model representation, $\hat{d}_{i,m}$ is compared to the distance $\dot{d}_i$ between the access point in question and the particle's location $\boldsymbol{\rho}$ using

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ftm}} = p(\boldsymbol{s}_{\text{ftm}} \mid \boldsymbol{\rho}) = \prod_i \prod_m p(\hat{d}_{i,m} \mid \boldsymbol{\rho}), \tag{3.11}$$

with

$$p(\hat{d}_{i,m} \mid \boldsymbol{\rho}) = \mathcal{N}(\hat{d}_{i,m} \mid \dot{d}_i, \sigma^2_{\text{ftm}}) \ . \tag{3.12}$$

Although it has been argued previously that the error of FTM measurements is non-Gaussian, using such a distribution as a probability density does not contradict this. The measurement error is already taken into account by the linear-log model, and the density function is only used to obtain a similarity measure between $\hat{d}_{i,m}$ and $\dot{d}_i$.

In [Bul22] we proposed a more promising approach that can cover the error over the entire measurement range while maintaining simplicity. To this end, we follow the idea of modeling the error using a skew normal distribution on $1\,\mathrm{m}$ wide bins, as shown in Figure 3.5b. The observed distances are grouped according to the actual distance between the ground truth where the FTM distance was recorded and the corresponding access point. For this, we define a bin containing all FTM measurements at distance $k \in \mathbb{N}^+$ as the set

$$\mathbf{BIN}_k = \{(\dot{d}_i, d_{i,m}) \in \mathbf{R} \,|\, k - 0.5 \leq \dot{d}_i < k + 0.5\} \ . \tag{3.13}$$

Again, $\dot{d}_i$ is the actual distance between the access point in question and some known location. Also, the measured FTM distance from an access point i with count m of M_i is again denoted by $d_{i,m}$. The set $\mathbf{R}$ contains pairs $(\dot{d}_i, d_{i,m})$ for all recorded data, associating a measurement with the actual distance to the access point i.

This allows to compute a histogram as shown in Figure 3.5b for $\mathbf{BIN}_7$ at the actual distance from $6.5\,\mathrm{m}$ to $7.5\,\mathrm{m}$. A comparison of different histograms ($\mathbf{BIN}_1$ to $\mathbf{BIN}_{30}$) shows that as the actual distance increases, the distribution becomes less skewed, approaching a standard Gaussian. Conversely, the skewness, i.e. the steep rise to the left of the mode of the distribution and the long flatter tail to the right, is much more pronounced for bins with a small actual distance. In simpler terms, the skewness decreases with increasing actual distance, e.g. the skewness of $\mathbf{BIN}_{25}$ is smaller compared to $\mathbf{BIN}_5$. Therefore, to fit the skew normal distribution as closely as possible to the histogram, it must be parameterized for each k-th bin individually.

With the above, it can be assumed that FTM measurements $d_{i,m}$ can be modeled according to

$$d_{i,m} \sim \mathcal{SN}(\xi_k, \omega_k, \alpha_k) \ , \tag{3.14}$$

where $\mathcal{SN}$ is the skew normal distribution defined by three parameters: *location* $\xi_k \in \mathbb{R}$, *scale* $\omega_k \in \mathbb{R}^+$, and *shape* $\alpha_k \in \mathbb{R}$ [OL76]. The location parameter determines the shift of the distribution along the horizontal axis, and the scale parameter controls the spread, i.e., a larger scale value makes the distribution more spread out; a small ω_k makes it more concentrated. The shape parameter is of particular interest in this scenario, as it controls the skewness of the distribution. If $\alpha_k > 0$, the distribution is right-skewed, so it's tail is left to the mode. Correspondingly, a negative skewness $\alpha_k < 0$ results in a tail to the right of the mode. If $\alpha_k = 0$, the skewness of the distribution is zero, corresponding to a Gaussian distribution.

A sensor model can then be described by the density function of the skew normal distribution given by

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)_{\text{ftm}} = p(\boldsymbol{s}_{\text{ftm}} \mid \boldsymbol{\rho}) = \prod_i \prod_m p(d_{i,m} \mid \boldsymbol{\rho}), \tag{3.15}$$

with

$$p(d_{i,m} \mid \boldsymbol{\rho}) = \mathcal{SN}(d_{i,m} \mid \xi_k, \omega_k, \alpha_k) = \frac{2}{\omega_k} \varphi\left(\frac{d_{i,m} - \xi_k}{\omega_k}\right) \Phi\left(\alpha_k \frac{d_{i,m} - \xi_k}{\omega_k}\right) . \tag{3.16}$$

Here $\boldsymbol{\rho}$ is again the arbitrary position of a particle, from which we can calculate the actual distance $\dot{d}_i = \|\boldsymbol{\rho} - \boldsymbol{\varrho}_i\|$, assuming $\boldsymbol{\rho}$ is the position of the initiator, to the location $\boldsymbol{\varrho}_i$ of the responder. Based on $\dot{d}_i$, we can then look up the correct set of parameters $(\xi_k, \omega_k, \alpha_k)$ according to $k - 0.5 \leq \dot{d}_i < k + 0.5$ that fit the data in $\mathbf{BIN}_k$ from (3.13). $\varphi(x)$ is the Gaussian density

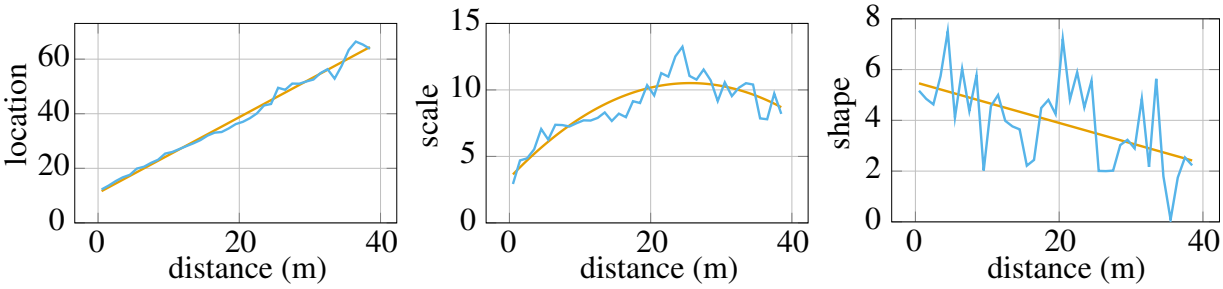


Figure 3.6: Estimation of $\mathcal{SN}(\xi_k, \omega_k, \alpha_k)$ by minimizing the negative log-likelihood function. The parameters location ξ_k, ω_k, and α_k, are optimized per $\mathbf{BIN}_k$ and shown in blue. The corresponding fitted function $\hat{\xi}(\dot{d}_i)$, $\hat{\omega}(\dot{d}_i)$, and $\hat{\alpha}(\dot{d}_i)$ are plotted in orange. Source: [Bul22].

function with zero mean and standard deviation of one, and $\Phi(x)$ is the cumulative distribution function of the Gaussian.

To estimate the parameters of $\mathcal{SN}(\xi_k, \omega_k, \alpha_k)$ using the measurements collected in $\mathbf{BIN}_k$, a MLE method can be used that minimizes the negative log-likelihood function [AC99]. As discussed earlier, the larger the actual distance, the more it approaches the shape of a standard Gaussian. Thus, the value of the shape α_k should decrease as k increases. This effect can be seen in Figure 3.6 for $k = 1$ up to $k = 38$, using the same data set as before. Although α_k increases again at about $20\,\mathrm{m}$, a negative slope is observed over the whole, shown by the orange line. The estimated values for the location ξ_k are quite obvious, as they shift the distribution along the horizontal axes, proportional to the actual distance. Looking at the results for the scale ω_k, the highest spread can be observed at $25\,\mathrm{m}$. If we compare the measurements of Figure 3.5a at the same point, some strong outliers can be observed. The fact that ω_k flattens out after $25\,\mathrm{m}$ could have several reasons, on the one hand the decreasing data volume observed from farther distances, but also effects depending on the placement of the access points (responder), e.g. multipath or absorption [Bul22].

The main drawback of the above approach is the limitation to estimate the probability of an FTM measurement only within a predefined $\mathbf{BIN}_k$. An observed distance that is larger or sometimes even negative must be exactly quantized to a k in order to obtain a probability value. Another problem is that the values of the individual bins sometimes fluctuate strongly, see the shape parameters at $19\,\mathrm{m}$ and $20\,\mathrm{m}$ in Figure 3.6. Between the boundaries of two bins, these rapid changes can lead to unwanted jumps in probability.

For this reason, a continuous model is proposed in [Bul22]. Instead of determining the parameters of the skew normal distribution over the discrete bins, the functions $\hat{\xi}(\dot{d}_i)$, $\hat{\omega}(\dot{d}_i)$, and $\hat{\alpha}(\dot{d}_i)$ are introduced, which provide fitting values for any given distance $\dot{d}_i$ between some initiator and responder indexed by i. The function is then used to continuously determine the parameters of the skew normal distribution. As seen in Figure 3.6, simple low order polynomials

are sufficient to fit the function to the respective parameters ξ_k, ω_k, and α_k. Location and shape can be represented by a first degree polynomial (linear) equation, while scale is described by a second degree polynomial (quadratic) equation. The sensor model in (3.15) thus changes to

$$p(d_{i,m} \mid \boldsymbol{\rho}) = \frac{2}{\hat{\omega}(\dot{d}_i)} \varphi \left(\frac{d_{i,m} - \hat{\xi}(\dot{d}_i)}{\hat{\omega}(\dot{d}_i)} \right) \Phi \left(\hat{\alpha}(\dot{d}_i) \frac{d_{i,m} - \hat{\xi}(\dot{d}_i)}{\hat{\omega}(\dot{d}_i)} \right) , \tag{3.17}$$

where the parameters used in (3.16) are replaced by functions of the actual distance $\dot{d}_i$. The fitting of these functions can be done by standard polynomial regression, e.g. the well-known least squares method [Ald97]. Besides the continuous properties, (3.17) has the advantage that only seven coefficients, two per linear and three for the quadratic polynomial, need to be stored within an ILS application.

Note that this approach is quite similar to the one discussed above in [Hor20a]. Despite the use of a combination of exponential functions, the skew normal distribution was chosen instead. In our opinion, this is not only a much more elegant representation, but it also allows for the continuous properties of the model in the first place. In addition, there is no need to distinguish between the case of two-dimensional and three-dimensional localization. As we have shown in [Bul22], the number of positions at which data are collected to estimate the parameters of the skew normal distribution, and thus the total number of measurements used, can be greatly reduced without degrading the accuracy of the position estimate. For example, the 73.000 FTM measurements used to draw the Figure 3.5a could be reduced to 7.287 at only 13 distinct positions within the building, instead of a $1\,\mathrm{m}$ equidistant grid.

Nevertheless, our approach also shares some disadvantages with the one proposed by [Hor20a]. First and foremost, the model has to be optimized for each building, as it depends very much on the local conditions. Moreover, the optimized model only represents the time, and thus the nature of the error, during the data acquisition process. Even small structural changes or misalignment of the Wi-Fi infrastructure can lead to unpredictable behavior.

3.2.4 Limitations and Discussion

Due to their similarity, Wi-Fi FTM and the classic RSSI-based Wi-Fi have some limitations in common. Both models need to adapt to a changing environment. This requires recalibration and reoptimization. For FTM, this is true even for very small changes such as moving an access point a few centimeters [Hor20b]. Although the effect of signal propagation through the human body is controversial, placing FTM responders at high elevations is preferable. In principle, the number of LOS communications can be increased in this way, which should reduce possible sources of error. Finding the optimal placement and number of access points is also an issue

common to both technologies. The approaches already discussed by [Raj16] (GDOP), [BHE01] (grid-based), or [MBH17] (fingerprinting) can be applied to FTM because of their general nature. However, approaches that perform placement based on signal strength are not compatible. Signal attenuation is a function of permittivity, conductivity, and frequency, and thus behaves differently for signal strength (RSSI) than for time-of-flight (distance) measurements. Again, the two simple paradigms of [MBH17] should be emphasized: First, avoid central areas and prefer exterior walls. Second, avoid placing transmitters on a common line in areas where pedestrians can walk.

On Android, Wi-Fi scans to find nearby access points are limited as of API level 28. No more than four scans per $2\,\mathrm{min}$ are allowed in foreground mode. The author of [Hor20b] dryly points out that at a normal walking speed of $1.4\,\mathrm{m\,s^{-1}}$, a scan is performed approximately every $42\,\mathrm{m}$. There is a workaround to bypass the throttling, but it requires a special setting in the developer options. This is not a good option for an ILS to be used publicly or commercially. Thus, localization using Wi-Fi RSSI measurements is nearly impossible on modern Android devices. Fortunately, Wi-Fi FTM is not directly affected by this, as the scan is only needed to identify nearby access points. Sending FTM requests and receiving range results is not restricted in foreground mode, but strangely ignored in background mode. Once an access point is known, an FTM request can be sent to it at any time, starting from the initiator. If there is a response, the TWR (cf. Figure 3.4) is performed. If there is no response, the access point is not nearby, does not support IEEE 802.11-2016, or is busy with other things. This allows two scenarios. Either an ILS application has already stored a list of all access points with position, MAC address and other relevant parameters of the building, or a Wi-Fi scan first identifies all possible responders in the vicinity. If these support *Location Configuration Information/Location Civic Report* (LCI/LCR data), the position (latitude, longitude and altitude) of the responder can be requested in the FTM request and will be received by the initiator in addition to the FTM distance measurements. This allows localization without prior knowledge, e.g. when entering a new building.

Probably the biggest limitation at the moment is software and hardware support. So far, only Android provides a proper API for sending and receiving FTM packets. With the Linux 5.3.7 kernel, the iwlwifi driver and hostapd have been prepared to support FTM, but still require small manual changes, such as enabling FTM by setting a global boolean flag [Bul20b]. Windows, OSX, and iOS do not provide an API. On the hardware side, things look even worse. According to the Android documentation, there are only four access points that support IEEE 802.11-2016 [Goo19]. There are a few other access points (e.g. Asus Wireless-AC1300 RT-AC58U) and SoCs (e.g. ESP32v2) described in the literature that support the standard [Bul22; Jio20]. According to the Wi-Fi Alliance product finder [Noac], there are at least 20 different chipsets

that are Wi-Fi Location certified. This suggests that many access points would theoretically be able to do this, but either the manufacturers do not communicate this or the support is not advertised in the access point's beacon frame.

According to [Goo19], the Android Wi-Fi RTT API is supported by almost 50 smartphones. Apart from Google's own Pixel line, these are mainly the newer flagship devices of other manufacturers like Samsung, Xiaomi or Zebra. The market share is therefore likely to be very small. So it may be a few years before operators of public buildings upgrade their Wi-Fi infrastructure to support FTM. For non-public use in factories, warehouses or hospitals, as well as for concepts that can offer rental devices, such as museums or trade shows, the technology is already well suited.

An approach that at least eliminates the need to upgrade the Wi-Fi infrastructure is called one-way RTT. This is a simpler protocol introduced in Android 12 in late 2021 that does not perform TWR with the responder, but can only measure the time between sending and receiving a message. Therefore, the turnaround time of the responder is not known and is assumed based on the *short inter frame space* (SIFS). The first works using this protocol report accuracies comparable to Wi-Fi RSSI approaches [Hor22].

As has been shown, FTM is a peer-to-peer protocol that is single-user in nature. With conventional request configuration, a single distance measurement (FTM exchange) can take about $30\,\mathrm{ms}$ per responder. This means that a single responder can serve about 30 initiators per second. This will consume all of its capacity, leaving no bandwidth available for data services. As the number of initiators requesting an FTM session increases, the probability of frame collisions also increases. This reduces the number of successful distance measurements and also leads to an increase in measurement errors [Ban18]. Such scenarios are particularly common in high-traffic locations with large numbers of people, such as football stadiums, airport terminals, or public transportation. The standard does not provide a native solution for this, so either more powerful hardware is required, e.g. more antennas and chipsets per access point, or other protocols. An approach to the latter has been presented by the authors of [Ban18]. They present a protocol that allows clients to passively listen to timing measurements periodically exchanged by an unmanaged network of base stations. They do this by exploiting the capabilities of the more accurate clocks required by IEEE 802.11-2016. They claim that this protocol allows an unlimited number of Wi-Fi users to position themselves. This may be true, but the protocol uses accurate ToA measures for positioning, which requires either synchronized clocks between all devices or methods to compensate for random clock drift. This is, after all, a hardware-oriented protocol. Without standardization by the Wi-Fi Alliance, this will have no application and remain just a good idea.

3.3 Inertial Measurement Unit

An *Inertial Measurement Unit* (IMU) is an electronic device that combines accelerometers, gyroscopes, and sometimes magnetometers to measure an object's own acceleration (specific force), angular rate, and orientation. They were originally developed for navigation and localization of aircraft and other missiles. For example, a standard IMU contains one accelerometer and one gyroscope per axis for each of the three main aircraft axes: pitch, roll and yaw. With respect to a global reference frame, i.e. a known launch position, orientation and velocity, the current position and orientation of a moving object can be determined by integrating the acceleration and angular velocity using dead reckoning methods. For example, the Apollo Guidance Computer of the Apollo program to land the first humans on the moon integrated an IMU for inertial guidance and continuous trajectory confirmation.

While the IMU of the Apollo program was the size of a medicine ball, equivalent systems today are no larger than a fingernail and can be found in almost any wearable device such as smartphones, smartwatches or smartbands. The small size and low manufacturing cost can be achieved by manufacturing IMUs as *Micro-Electro-Mechanical Systems* (MEMS). They are used to rotate the contents of the screen, to count the required 10,000 steps per day or to detect simple sports exercises. The idea of using IMUs for ILS is therefore obvious. Such methods and procedures are called *Pedestrian Dead Reckoning* (PDR). Similar to an aircraft, the goal is to determine the position and orientation of a person relative to the global reference frame. In the simplest case, the IMU is rigidly attached to the body. This has the advantage that the IMU and the person can share a local (relative) coordinate system. In contrast, in the more difficult scenario of smartphone-based localization, the IMU can be considered as a loose object, e.g. it can be moved in the hand or regularly placed in the pocket during long walks. This makes the measured IMU data dependent on the orientation of the smartphone and not on the orientation of the person to be localized. To overcome this problem, a second (more or less) global coordinate system is often used to transform the measured data. This is done by aligning one axis (e.g. the z-axis) of the smartphone's local coordinate system (see Figure 3.7 (a)) with the Earth's gravitational axis, so that the remaining axes are parallel to the (flat) surface, also called the *east-north-up* (ENU) frame. The transformation can be calculated using the gravitational acceleration measured by the three axis accelerometers [Ebn19]. After that, a change in walking direction can be obtained from the measurements of the axis parallel to gravity, regardless of how the smartphone is held by the user.

Despite their low cost and small size, consumer IMUs based on MEMS technology are noisy and often biased. An example of the noise behavior of a Motorola Moto Z smartphone was shown by [Ebn19]. While recording IMU measurements, the device rests on a flat surface with

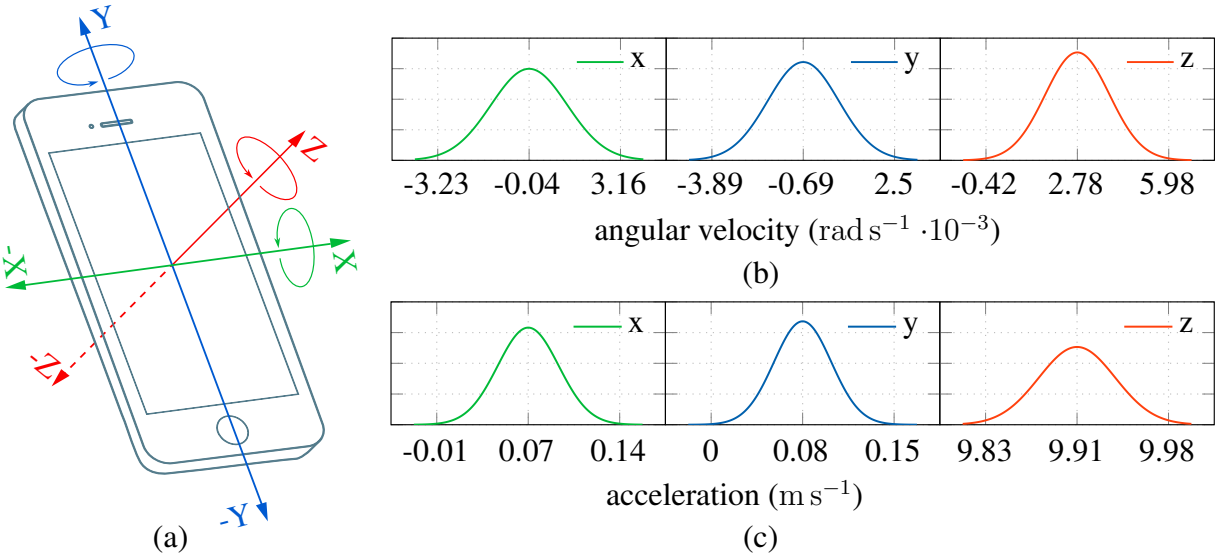


Figure 3.7: (a) shows the local coordinate system of the IMU as defined by Android. According to the right hand rule, the sign of the angular velocity is positive in clockwise direction when looking along the positive direction of the axis. (b) shows the Gaussian distribution of sensor noise for the gyroscope of an IMU and (c) for the corresponding accelerometer. Data was recorded by the author of [Ebn19] using a Motorola Moto Z smartphone.

its local z-axis pointing upward, parallel to the axis of gravity. Figure 3.7 (b) and (c) visualize the data fitted to a Gaussian distribution on each of the three local axes. It is easy to see that the quiescent signals have a non-zero mean and an obvious variance. The expected gravitational acceleration $\boldsymbol{g}$ at the location of the smartphone should be $9.8095\,\mathrm{m\,s^{-2}}$, but the mean over the three axes shown is $9.9062\,\mathrm{m\,s^{-2}}$. According to [Ped], this bias comes from three main sources: imperfect alignment of all three axes on the smartphone motherboard, thermal stress during soldering, and alignment of the smartphone motherboard within the case. In addition, bias is not only a constant problem, but also depends on factors such as the current ambient temperature, atmospheric pressure, and humidity. For this reason, most hardware vendors implement online calibration for the IMU, including temperature compensation, bias calibration, and optional scale calibration [Ebn19]. However, the measurements shown in Figure 3.7 (b) and (c) were already taken with this calibration enabled.

Consider the simplest PDR by integrating the IMU measurements over time, i.e. the angular velocity is integrated once to get the change in walking direction and the acceleration is integrated twice to get the distance walked. However, integration turns noise into drift [MYNK09]. Sensor drift occurs when noise (with a non-zero mean) is continuously added during integration of a signal and thus accumulates in the resulting signal. Modern gyroscopes provide good results for time spans up to several minutes without noticeable drift. This is achieved by a high signal-to-noise ratio (SNR), on-chip online calibration that includes temperature compensation,

bias calibration, and optional scale calibration, accurate time measurements, and small integration time frames. Accelerometers, on the other hand, drift rapidly and unacceptably within seconds. Double integration of the increasing acceleration error results in a position error that grows cubically with time in the short term, leading to a drift of more than a meter in $10\,\mathrm{s}$ [Fox05]. An additional position error is caused by the angular velocity of the gyroscope, which is used to project the acceleration signal into the global coordinate system. Even a small tilt error of $0.05°$ causes a horizontal position error of $7.7\,\mathrm{m}$ after $30\,\mathrm{s}$ [Woo].

This requires alternative approaches for estimating the walked distance and the pedestrian's position. A very early and still popular approach was presented by [Fox05]. Here, the IMU is attached to the foot by tucking it into the shoelace. This static placement allows for *zero velocity updates* (ZUPT) by detecting the stance phase of the person, i.e. a constant position and attitude of the system. The ZUPT then attempts to counteract the drift by performing either a hard or soft update of the IMU. As the name implies, a hard update resets the position, velocity, and yaw to zero and re-initializes them using a newly calculated global transformation matrix. Soft updates, on the other hand, use a model that uses knowledge of how the errors have evolved over time to obtain an estimate of the accumulated errors since the last update. The accumulated error estimate is then used to correct and calibrate the PDR system. The author of [Fox05] also integrated the ZUPT into an *Extended Kalman Filter* (EKF) and reported very small drifts of a few centimeters over a total distance of nearly a hundred meters. These results were confirmed by the authors of the [SNH10].

However, for pedestrian self-localization in recreational places such as shopping malls or museums, adding an additional sensor to the foot does not seem very practical. Therefore, most ILS rely on step detection [BH13; Bre17; Guo20; MPS14]. For each detected step, a step length is assumed or determined and added up. The advantage is that even if the step length estimate is inaccurate, the error grows linearly with time.

3.3.1 Step Evaluation Model

Counting steps has become a popular sport, at least in Germany and Japan [IT07]. Devices that count steps are called pedometers, and people are encouraged to count $10\,000$ steps a day to stay healthy. Smartphones use software pedometers based on the IMU for this purpose. For example, since Android 4.4 (2013) a step detector and counter using acceleration data is integrated into the operating system [Goo]. There are numerous approaches in the literature to achieve robust step detection using IMU data. A particularly good and detailed summary is given in [BH13]. The authors distinguish three main classes based on the data representation that the algorithmic approaches operate on: time domain, frequency domain, and feature clustering. While the latter

two allow for a complex approach and versatile parameterization, the experiments conducted by [BH13] have shown that especially the simpler time-based methods provide more robust results at lower complexity.

In the time domain, a low-pass filter is often applied to slightly reduce noise in a pre-processing step. The step detection algorithms are then mostly based on an analysis of the cyclic behavior of the walking motion. Assuming a symmetrical walking motion, one cycle per stride (two steps) can be read from the data. For each step, a strong short term vertical acceleration occurs from the moment the heel is lifted. A step can now be detected by thresholding, peak detection, or period analysis such as zero crossing counting. To obtain rotation invariance, the magnitude of the signal is often used instead of the individual axes.

In this work, we also use step detectors operating in the time domain. The first approach is simple, but has served well in previous work [Ebn15; Fet16]. First, a simple low-pass biquad filter (e.g., cutoff frequency $3.0\,\mathrm{Hz}$, sampling frequency $100\,\mathrm{Hz}$, quality factor 0.7) is used to filter the magnitude $|\boldsymbol{a}|$ of the three-axis accelerometer $\boldsymbol{a} = (a_x, a_y, a_z)^T$ rotated to the ENU frame. The filtered and transformed magnitude is then normalized by removing the current gravitational acceleration $\boldsymbol{g}$, resulting in $|\hat{a}|$. We now try to detect a zero crossing by comparing the oldest magnitude value $|\hat{a}|_{\text{old}}$ with the current magnitude value $|\hat{a}|_{\text{cur}}$ extracted from a rectangular window function of predefined size, e.g. $100\,\mathrm{ms}$. If a zero crossing with $|\hat{a}|_{\text{old}} > 0$ and $|\hat{a}|_{\text{cur}} < 0$ and $|\hat{a}|_{\text{cur}}$ exceeds a certain threshold (e.g. $0.8\,\mathrm{rad\,s^{-2}}$), a step will be detected. To prevent multiple detections within an unrealistically short interval, we block the entire process by the size of the window function after a zero crossing.

The probabilistic model that incorporates this into a particle filter-based sensor fusion is as simple as the method itself. Again, we consider a Gaussian as probability density function given by

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{step}} = p(n_t \mid \boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}) = \mathcal{N}(\Delta\rho_t \mid n_t \cdot \mu_{\text{step}}, \sigma_{\text{step}}^2) \ . \tag{3.18}$$

Here $\Delta\rho_t = \|\boldsymbol{\rho}_t - \boldsymbol{\rho}_{t-1}\|$ is the distance covered by the particle between time t and $t-1$. The position of the particle $\boldsymbol{\rho}_{t-1} \Rightarrow \boldsymbol{\rho}_t$ changes according to the used movement model (see Section 4). The number of steps n_t detected between $t-1$ and t multiplied by the step length μ_{step} represents the mean and σ_{step}^2 the variance of the Gaussian. The equation (3.18) can thus be interpreted as a similarity measure between $\Delta\rho_t$ and the distance measured based on n_t and μ_{step}. This means that those particles that are closest to the distance traveled by the pedestrian between two successive filter updates will be weighted higher.

Of course, this is only true if both the step detection and the step length are accurate. In particular, estimating the step length of a pedestrian based on freely carried smartphones is a difficult problem. Similar to step detection, there are a large number of different approaches. A

good summary can be found in [Jah10] and [VJ19]. A more recent approach has been proposed by the authors of [KA20]. They train a convolutional neural network based on a set of features (e.g. standard deviation, amplitude, max, min ...) extracted from a moving window of the IMU data to compute a step length or distance change. While the results are promising, the data set used for the experiments seems to have limited realistic conditions. It considers four different states (bag, hand, pocket, body) of the smartphone position during walking sequences of 8 subjects. Looking at the provided video, the changes between the states seem very mechanical and the walking movements are purposeful. Even the authors who provided this data set are self-critical, as the subjects do not perform truly natural movements [YHF19]. Finally, similar to other data sets, only shorter distances around $200\,\mathrm{m}$ are examined. It would be exciting to test this approach on a more realistic and therefore more difficult data set such as [Her22].

The most advanced approaches have one thing in common: a training or calibration phase must first be carried out. As is well known, there are a number of things that need to be taken into account in order to prevent the system from being over-adapted, e.g., tailored to the walking behavior of a single person. On the other hand, it is not easy to train a certain generalization for a multi-purpose system, since the walking behavior of individual people can be very different. To avoid this controversy and to have a ILS that is as easy to install as possible, i.e. free of complex data collection and calibration, the step length μ_{step} is often assumed to be a constant value, e.g. $70\,\mathrm{cm}$ per $500\,\mathrm{ms}$. If knowledge of the building's structure is available, μ_{step} can be taken based on the current position of the particle, e.g. on a staircase a smaller step length like $30\,\mathrm{cm}$ can be assumed. Similar assumptions can be made based on activity recognition (running, walking, standing, strolling).

Our step detection approach presented above provides a strictly binary classification, i.e. either a step is detected or not. In the presence of sensor bias, drift, and other sources of noise, this step detector tends to produce false-positive results. This means that in (3.18) every particle has to obey this decision and a wrongly detected or missed step produces a global error in the posterior. A good way to deal with this problem is to estimate a probability for a step, as suggested by [KGD14]. Again, a biquad filter is used to low-pass filter the three-axis accelerometer $\boldsymbol{a} = (a_x, a_y, a_z)^T$ rotated to the ENU frame. Removing the gravitational acceleration then yields the normalized magnitude $|\hat{a}|$. The local maximum and minimum are then searched for using a data vector $|\hat{\boldsymbol{a}}|$ extracted from a rectangular window function of predefined size, e.g. $500\,\mathrm{ms}$, containing the respective $|\hat{a}|$. The probability γ of detecting a step is then estimated using a Gamma distribution $\Gamma(...)$:

$$\gamma = \eta \cdot \Gamma(\Delta f_{\text{min}}^{\text{max}}(|\hat{\boldsymbol{a}}|) \mid \alpha, \beta) \tag{3.19}$$

with

$$\Delta f_{\text{min}}^{\text{max}}(\boldsymbol{x}) = \| f_{\text{max}}(\boldsymbol{x}) - f_{\text{min}}(\boldsymbol{x}) \| \ ,$$

where $\Delta f_{\text{min}}^{\text{max}}(\boldsymbol{x})$ is a function that calculates the absolute difference between the maximum and minimum values found with $f_{\text{max}}(\boldsymbol{x})$ and $f_{\text{min}}(\boldsymbol{x})$ in a given data vector $\boldsymbol{x}$. The parameters α and β can be chosen either empirically or by fitting a gamma distribution to a labeled training data set where it is known at what timestamp a step was performed. The authors of [KGD14] report $\alpha = 20.12$ and $\beta = 0.19$ by fitting (3.19) to a $30\,\text{m}$ straight walk. η denotes a normalization $[0, 1]$ based on the mode of the distribution, which can be easily computed in closed form.

The gamma distribution was originally chosen for its ability to correct for skewness, i.e., a sharp increase at the beginning and a flattening decrease toward the end of the mode of the distribution. This is because low values of $\Delta f_{\text{min}}^{\text{max}}(\boldsymbol{x})$ very rarely represent a step, while larger values are more likely to represent a step, and very large values are usually more likely to represent some other activity, such as dropping or shaking the smartphone. However, if we look at the gamma distribution resulting from the parameters estimated by [KGD14], a Gaussian could probably suffice.

The above then results in a sensor model of the form

$$\begin{aligned} p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{step}} &= p(\gamma_t \mid \boldsymbol{\rho}_t, \boldsymbol{\rho}_{t-1}) \\ &= \gamma_t \cdot \mathcal{N}(\Delta\rho_t \mid \mu_{\text{step}}, \sigma_{\text{step}}^2) + (1 - \gamma_t) \cdot \mathcal{N}(\Delta\rho_t \mid \mu_{\overline{\text{step}}}, \sigma_{\overline{\text{step}}}^2) \ , \end{aligned} \tag{3.20}$$

where $\Delta\rho_t$ is again the distance traveled by the particle between time t and $t-1$. The mean μ_{step} and the variance σ_{step}^2 are also similar to (3.18). Basically, the formula (3.20) is a mixture model with two parts: On the left side of the plus operator is the probability density for detected steps, weighted by γ_t, and on the right side is an alternative probability density for the absence of steps. The right side can be adjusted depending on the scenario, but the most obvious would be a density with $\mu_{\overline{\text{step}}} = 0.0\,\text{cm}$ and $\sigma_{\overline{\text{step}}} = 0.1\,\text{cm}$, giving a higher value to particles that stop or move only slightly.

The big drawback of this approach: The update rate of the particle filter must be similar to the average step frequency of a person, cf. the size of the data vector $|\hat{\boldsymbol{a}}|$, so that a suitable probability can be assigned per possible step. If the update rate were too slow, multiple steps could be detected between updates of the filter, and thus γ could not cover this due to the small size of $|\hat{\boldsymbol{a}}|$. Of course, the sensor model of the binary model in (3.18) and the probabilistic approach in (3.20) can be combined by making n a tuple, i.e. adding a probability γ to each detected step. Finally, the Gaussians from (3.20) are swapped with those from (3.18).

3.3.2 Turn Evaluation Model

As discussed at the beginning of Section 3.3, the walking direction or relative angular change $\Delta\theta$ can be obtained using the IMU's gyroscope. To be independent of how the smartphone is held by the user, we need to transform the gyroscope readings to the ENU frame. There are two variants of the required transformation $\boldsymbol{T}_{L\to G}$ from a local to a global coordinate system. Either into an unspecified coordinate system or into a specified coordinate system. The former rotates one axis of the local coordinate system parallel to the gravitational axis, while allowing the two perpendicular axes to drift at the same rate as the gyroscope. This is similar to Android's virtual game rotation vector sensor, which uses only the gravitational axis as an absolute source of information. In contrast, the specified coordinate system is calculated using the magnetometer, i.e. an absolute orientation is determined using the Earth's magnetic field.

An excellent approach for estimating the transformation is the Madgwick filter [MHV11]. It offers low computational complexity compared to similar approaches and requires only one free parameter for configuration. We use the filter to perform sensor fusion of gyroscope and accelerometer to compute $\boldsymbol{T}_{L\to G}$ for an unspecified global coordinate system. This is done by using gradient descent to optimize a quaternion that orients the accelerometer data to a known gravity reference. In our use case of receiving a walking direction, the vertical z-axis of the local coordinate system of the smartphone is defined by the direction of the gravity vector as its reference. This quaternion is then weighted and integrated with the gyroscope quaternion and the previous orientation. The gyro quaternion is obtained by rectangular integration, i.e. multiplying the measured angular velocity $\boldsymbol{g} = (g_x, g_y, g_z)^T$ by the time period Δt in which it was recorded, and a simple radian-to-quaternion conversion. The result is the transformation $\boldsymbol{T}_{L\to G}$, which can then be normalized and converted back to radians for easy use. Note that especially for cheap IMUs or scenarios where the orientation of the smartphone is constantly changing, the Madgwick filter must be computed in very short intervals of one or two seconds to maintain good results and to compensate for drift.

After transforming the IMU readings to the ENU frame using $\boldsymbol{T}_{L\to G}$, the angular change θ (yaw), given in radians, of a single gyroscope reading is calculated by integrating over the gyroscope's z-axis:

$$\theta = \boldsymbol{T}_{L\to G} \cdot \boldsymbol{g} \cdot (0, 0, \Delta t) \ , \tag{3.21}$$

where Δt is the time difference between the last gyro reading and the current reading. The integration is approximated using the quadratic rule [DR07] and thus results in a simple multiplication by Δt. The relative change in walking direction $\Delta\theta_t$ since the last update of the particle filter is then obtained by summing all measured θ between $t-1$ and t. It is stored in the observation o_t. The number of summed θ depends on the sampling frequency of the gyroscope.

A probabilistic sensor model can be given by

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{turn}} = p(\Delta\theta_t \mid \Theta_t, \Theta_{t-1}) = f_{\text{mises}}(\Delta\Theta_t \mid \Delta\theta_t, \kappa_t) \tag{3.22}$$

where $f_{\text{mises}}(x \mid \mu, \kappa)$ denotes the von Mises distribution [Ebn15]. It is also known as a circular normal distribution because it is a continuous probability distribution on a circle and is thus defined only for values within $[0; 2\pi]$. In contrast to a Gaussian, this makes the von Mises a suitable similarity measure for angular values. The position center μ is given by $\Delta\theta_t$ and the input $\Delta\Theta_t$ is the angular difference of a single particle between time step $t-1$ and t. κ denotes the concentration measure, which is a reciprocal measure of the dispersion, so $1/\kappa$ is analogous to a Gaussian' variance σ^2. It is thus a measure of the allowed deviation of the particle motion performed by the transition model and the measured change in walking direction $\Delta\theta_t$ provided by the gyroscope. In other words, the larger the error or drift of the gyroscope, the smaller κ must be chosen to represent the corresponding variance.

In (3.22), the estimated angle $\Delta\theta_t$ is used directly, but this leads to the problem that any change in the orientation of the smartphone is also considered a change in heading. For example, rotating the phone from portrait to landscape will result in an estimated heading change of $90°$. During such a rotation, the acceleration relative to the smartphone also changes. Simply put, the accelerometer readings are also rotated, i.e. the acceleration generated by the walking motion can now be observed on different axes of the accelerometer than before, since the local coordinate system of the smartphone is also rotated. In the global frame, the walking motion does not change, only the smartphone rotates. The vector of the walking motion, or motion vector $\boldsymbol{m}$, therefore always points in the direction the user is moving. It can be estimated from the observation that when walking, most of the variations in the horizontal plane of the acceleration signal are parallel to the direction of motion. First, the linear acceleration (acceleration without gravity) is projected onto the gravitational vector, i.e. transformed into the ENU frame. A *principal component analysis* (PCA) is then applied to the projected measurements to determine the direction in which the acceleration variations are greatest. The motion vector $\boldsymbol{m}$ is thus given by the first principal component, i.e. the eigenvector with the highest eigenvalue and thus the highest variance. Note that finding the direction of $\boldsymbol{m}$ is not possible without further assumptions, such as that the user is always walking forward. We can omit this knowledge anyway, since we are not interested in $\boldsymbol{m}$ itself [Ebn15].

To solve the above problem that any change of the smartphone's orientation is also considered as a change of heading, we are only interested in the change of $\boldsymbol{m}$ between two consecutive time steps of the particle filter. If the orientation of the smartphone changes significantly, i.e. a rotation from portrait to landscape mode, the motion vector also rotates, since it always points

in the direction of the user's movement. The angle between the motion vector $\boldsymbol{m}$ and $\boldsymbol{m}_{t-1}$ can be calculated with

$$\cos \phi_t = \frac{\boldsymbol{m}_t \cdot \boldsymbol{m}_{t-1}}{\|\boldsymbol{m}_t\| \cdot \|\boldsymbol{m}_{t-1}\|} \tag{3.23}$$

and since the direction of the vector is irrelevant to us, the angle difference is calculated using

$$\Delta\phi_t = f_{\text{min}}(\pi - \phi_t, \phi_t) \ . \tag{3.24}$$

This can now be incorporated into (3.22) by setting the concentration measure κ of the von Mises distribution to depend on $\Delta\phi_t$ using the following heuristic:

$$\kappa_t = \frac{\kappa_{\text{turn}}}{e^{(2\Delta\phi_t)}} = \frac{1}{\sigma^2_{\text{turn}} \cdot e^{(2\Delta\phi_t)}} \ . \tag{3.25}$$

Here, κ_{turn} is found empirically, e.g. a value of 5 is identical to a standard deviation of $0.44\,\text{rad}$ ($\approx 25°$). For ease of understanding and use, we have rearranged the equation so that we can specify a variance σ^2_{turn} directly. Reversed as to the variance of a Gaussian, high values for κ_t result in a probability density with a sharp peak, while at very low values such as 0.1 it becomes almost uniform. The latter leads to the effect that for high $\Delta\phi_t$, i.e. strong changes of the motion vector, the change of the walking direction $\Delta\theta_t$ measured by the gyroscope is in fact ignored for the current particle filter update. Therefore, the particles keep their last known heading to a large extent, which prevents a falsely detected heading change as we could show in [Ebn15].

3.3.3 Limitations and Discussion

The previously mentioned step and turn detection methods are always based on a non-specific coordinate system. This means that only the relative orientation of the pedestrian can be determined. For an ILS based on PDR alone, the starting position and orientation must be known. A hybrid ILS is one that also integrates an absolute sensor source, such as BLE, into the fusion to eliminate the need for initial position knowledge. The absolute orientation can then be derived from the mean angle over all particles or the motion vector of the state estimate. However, these estimation measures are not based on a corrective sensor measurement that serves as a similarity measure. This is only possible if the magnetometer is used in addition to the accelerometer and gyroscope.

The magnetometer determines absolute orientation using the Earth's magnetic field. Unlike the outdoor environment, the indoor magnetic field is strongly distorted by structural elements of the building, electrical systems, industrial and electronic equipment [BYB07]. Various approaches to deal with this can be found in the literature. Most of them are based on the fact

that magnetic disturbances are highly localized in space. Therefore, the geomagnetic field has conspicuous signatures for different areas, i.e. a high degree of inhomogeneity throughout the indoor environment. This allows the use of location fingerprinting methods, often referred to as magnetic matching in this specific case. Similar to WiFi fingerprinting (see Section 3.1), a two-step process is performed. First, a fingerprint database is built in an offline or training phase. In the online step, the user's location is estimated by comparing the current magnetic measurements with the database entries using some kind of matching method such as a similarity measure like Dynamic Time Warping (DTW) or a classification approach like k-Nearest Neighbor (kNN). A good summary of different approaches and procedures can be found in [OAM22].

An accurate magnetic match does not automatically provide the absolute orientation. Using the measurement or fingerprint directly is not very promising because the material properties that caused the distortion in the magnetic field are unknown in most cases. Thus, calculating the orientation based on magnetometer readings still contains the error from the distortion. The authors of [Aba14] use a two-step approach to still be able to do this. First, they take advantage of the fact that the magnetic field is not perturbed everywhere in the building and use machine learning to filter out very bad orientation estimates. Then, in a second step, they use a consensus algorithm to merge the orientation estimates of multiple people moving in the same direction in a distributed fashion. While the results of the experiments are very promising, with an average error of nearly 2°, the obvious question of practicality arises here. There are many other approaches such as [Yua15], [RC14], and [Hos14], but many recent research papers describing the magnetometer as too error-prone choose not to use it, preferring to solve the orientation problem via hybrid ILS approaches [Lai19; Leh20].

The most limiting factor is, of course, the human being. As already mentioned, the range of human movement patterns is very diverse and generalization is difficult. The lowest common denominator in our eyes is footstep detection and relative but stable orientation estimation as presented here. What is missed, however, are people with limitations, such as wheelchair users or the visually impaired. While the latter could at least use the approaches by adapting to special movement patterns, IMU data recorded from a wheelchair do not contain a clearly recognizable pattern [Fu14]. For example, the IMU's accelerometer does not produce significant signals at the constant linear speed of an electric wheelchair. PDR approaches therefore use additional sensor technology directly on the wheelchair [Gup15; Mas20].

The combination of the sensor models presented above with a simple transition model, e.g. a normally distributed variance over the step length μ_{step}, in a particle filter results in a PDR based ILS. While this already yields reasonable results [KGD14], we will see in the course of this work that incorporating step and turn detection into the transition step with additional map

information brings some major practical advantages. Nevertheless, from a theoretical point of view, these approaches can be considered identical. At this point, it should be noted that there are much more advanced PDR methods in the literature than those presented above. One of the most promising approaches in our opinion is RoNIN [YHF19] and NILoc [Her22]. Both train neural networks to regress a velocity vector against an IMU sensor history. The results are promising, but require extensive training. Nevertheless, these methods still need to be made usable in untrained/unknown environments, perhaps using transfer learning, to achieve real practicality.

3.4 Barometer

Multi-story buildings require an ILS to be able to detect the current floor as well as to detect a change of floor. In the case of the latter, pedestrians are known to use a staircase or an elevator for this purpose. It should be noted that floors can be skipped if, for example, a staircase connects the first level and third level. The change to three dimensions increases the overall complexity of the system and its probabilistic methods. If any 3D position within a building is a realistic hypothesis for state estimation, the particle filter must also be able to sample to each of these possible positions. This greatly increases the number of particles needed to approximate the posterior, and thus the computational complexity. For most applications, it is therefore assumed that particles can only move in the plane of the floor. A floor change is then either a discrete or a continuous decision. In the first case, a particle is exactly on one floor. In the second case, a particle can also be between two floors during the change. To avoid the complexity of three-dimensionality described above, a spatial model, such as a graph or a mesh, is usually used to model stairs or elevators, greatly reducing the area in which a 3D position is possible.

Independent of the spatial models, which we will discuss in Section 4.1, a discrete floor change or a movement of a particle into the third dimension has to be validated. In other words, based on measurements from suitable sensors, the particle is assigned an appropriate weight in the evaluation step of the particle filter. As seen in Section 3.1 and Section 3.2, Wi-Fi, BLE and Wi-Fi-FTM are able to estimate a 3D position, but in practice their accuracy is often only sufficient to determine whether the user is on a staircase or in a stairwell [Ebn13; KGA17]. Due to their electromagnetic and metallic properties, elevators generate such strong errors that the probability density often leads to multimodalities, i.e. multiple hypotheses for state estimation. The accelerometer is able to measure the acceleration for height changes and thus provides a good basis for detecting a floor change in the first place. However, as we will see in Section 3.5, it needs pattern recognition techniques to achieve the necessary accuracy. With

simpler methods, even a quick removal of the smartphone from the pocket could be detected as a floor change [Ebn20].

For a few years now, especially the more expensive smartphones have been equipped with a barometric sensor. It measures atmospheric pressure in hectopascals (hPa), and since this pressure varies primarily with altitude, the smartphone's barometer is well suited for detecting floor changes. As the name suggests, atmospheric pressure is the pressure within the Earth's atmosphere. Roughly speaking, this pressure is generated by the weight of the column of air standing on the Earth's surface or on a body. The higher we are on the Earth's surface, the lower this pressure becomes. Therefore, lower barometric readings are correlated with higher altitudes. For example, the standard mean sea level pressure (MSLP) is $1013.25\,\mathrm{hPa} \approx 1\,\mathrm{bar}$, while at the top of Mount Everest it is only $325.4\,\mathrm{hPa}$. For weather forecasting or at weather stations, air pressure is often normalized to sea level in order to get comparability between all regions of the Earth. Given environmental information, this is done using the barometric formula [CO67]:

$$f_\psi(h_1) = \psi_{h_0} \left(1 - \frac{a(h_1 - h_0)}{T_{h_0}}\right)^{\frac{Mg}{Ra}} , \tag{3.26}$$

where ψ_{h_0} is a pressure and T_{h_0} is a temperature at reference altitude h_0. The lapse rate a is the decrease of an atmospheric temperature with altitude. The universal gas constant R is well known, and the gravitational acceleration g can be easily derived from publicly available data sets for the current location or by using the IMU's accelerometer. The molar mass M depends strongly on the environment, i.e. whether there is a clear line of sight between h_0 and h_1 or not. Consequently, (3.26) can also be used to calculate the pressure of different floors of a building given its elevation h_0 as well as the floor height and ceiling thickness.

The most naive approach to floor detection would be to compare the barometer reading with the pressures of the floors derived from (3.26). However, determining all the parameters of the barometric formula is not easy. They must be determined accurately in both space and time. Spatially, because different positions within a floor behave differently to the pressure, for example due to increased room temperature or humidity. Temporally, because the absolute pressure changes with the seasons and even the time of day. All this would require that the parameters of the (3.26) be continuously and accurately determined. This can be done using reference measurements from stationary barometers placed on each floor [LHG13] or by calibrating the wearable smartphone using a single reference device on the same floor and meteorological information [Wan06]. Both approaches obviously have to make stronger assumptions about the spatial context.

In addition to the temporal and spatial factors described above, the MEMS barometers built into smartphones are noisy and sensitive to rapid changes. For example, opening a window or

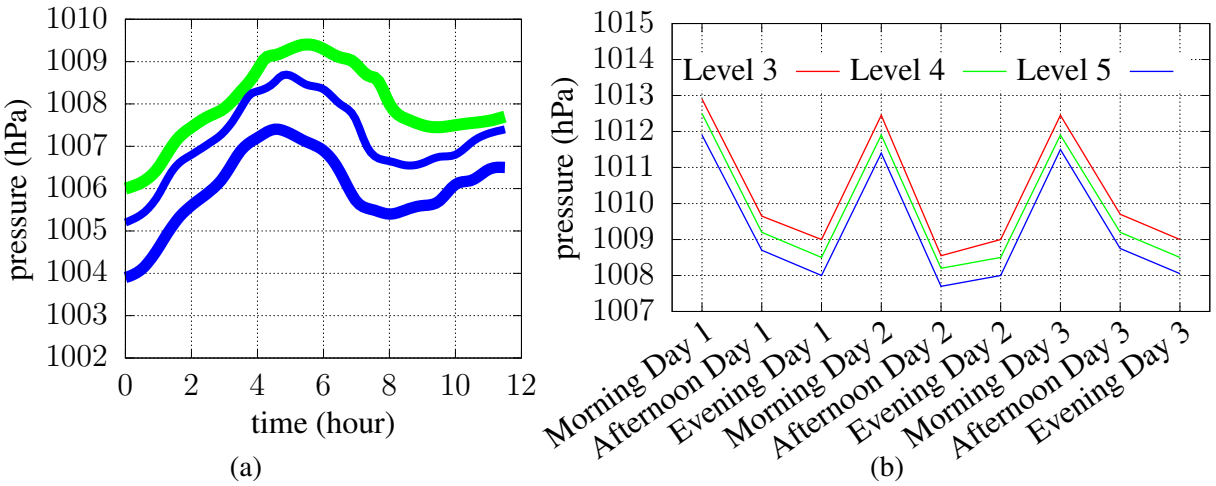


Figure 3.8: Recordings of absolute pressure readings for two scenarios. (a) 12-hour pressure trend on a single floor using one Galaxy S3 (green) and two Nexus 4s (blue). The width of the lines indicates the uncertainty of the measurements. (b) Recordings of three pressure readings on three different floors at three different times on three different days. The figures are based on the data sets provided by [Mur14].

door can dramatically change the measured pressure [PKC08]. Figure 3.8a shows a 12-hour pressure trend on a single floor using a Galaxy S3 (green) and two Nexus 4s (blue). Even though all devices were placed flat on the table in a closed room, the measurements are very different. The width of the lines describes the uncertainty of the measurements, indicating that the barometer has a noise that should not be underestimated, even at short intervals. It can also be seen that not only the time of day and thus things like temperature variations affect the pressure readings, but also the different devices used. For the two NX devices, this results in an average pressure difference of $1.2\,\mathrm{hPa}$, which corresponds to an altitude difference of $\sim 10\,\mathrm{m}$.

Figure 3.8b illustrates how the pressure measurements on different floors behave on different days. These results show that the (absolute) pressure measurements show significant diurnal variations, while the pressure difference between different pairs of floors is remarkably consistent and uniform for any given building. These observations show that absolute pressure measurements, even with reference measurements, are difficult to use for floor identification when two devices of identical construction give different readings. For this reason, we focus on the relative difference between the barometric pressure readings. All measurements are relative to a reference value, e.g. the average of the first $3\,\mathrm{s}$ of a localization run.

It is clear that this does not allow us to determine the exact floor by fingerprinting, but it does allow us to detect the change of a floor. This also has the advantage that we are independent of external information sources, i.e. we only need the smartphone as an information source.

The exact floor is then determined by sensor fusion with absolute information sources such as Wi-Fi, BLE or Wi-Fi FTM.

3.4.1 Atmospheric Pressure Model

Regardless of whether discrete or continuous floor changes are considered, the probabilistic evaluation model can again be described by a Gaussian distribution of the form

$$\begin{aligned} p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{baro}} &= \mathcal{N}(f_\psi(\Delta\rho_{z,t}^i) \mid \Delta\psi_t, \sigma_{\text{baro}}^2) \\ \text{with } \Delta\rho_{z,t}^i &= \rho_{z,t}^i - \rho_{\text{ref}}^i \text{ and } \Delta\psi_t = \bar{\psi}_t - \psi_{\text{ref}} \ . \end{aligned} \tag{3.27}$$

Here $\Delta\rho_{z,t}^i$ denotes the change in z-direction (height) between time t and a reference ρ_{ref}^i taken at the same time as the barometer reference reading ψ_{ref}. The mean of the Gaussian is set to the relative pressure change $\Delta\psi_t$ and the uncertainty (noise) is modeled with σ_{baro}^2. The average pressure $\bar{\psi}_t$ is calculated for all values measured between to last and the current timestamp. To increase stability, a low pass filter can be applied to the raw pressure reading for smoothing purposes [Fet23b]. To get $\Delta\psi_t$ a reference value ψ_{ref} is needed. There are several ways to determine a value for ψ_{ref}. Considering a pedestrian localization of less than an hour, it is sufficient to use the average pressure of the measurements taken in the first few seconds (e.g. $3\,\text{s}$) of this run. As shown before (see Figure 3.8a), this will certainly fail for longer runs. For this, ψ_{ref} can be periodically determined by using a circular buffer of fixed size containing all barometer measurements, where ψ_{ref} denotes the oldest entry. Consequently, a history of all particle sets must be stored in the circular buffer to get the corresponding altitude reference ρ_{ref}^i with the same timestamp as ψ_{ref}.

The conversion of the altitude change $\Delta\rho_{z,t}^i$ to atmospheric pressure is done with $f_\psi(\Delta\rho_{z,t}^i)$ (see (3.26)), which again requires knowledge of the environmental parameters. At least for air-conditioned buildings, the lapse rate a and the temperature T_{ref} can be assumed to be constant. These are usually $0.283\,\text{Kperm}$ ($9.8\,°\text{perkm}$) and $294.15\,\text{K}$ ($21°$). Otherwise, both values may be available from a nearby weather station using a web API. Often it can also be assumed that both values do not change so quickly over time, so a single initial determination per run is sufficient. The universal gas constant R is $8.314\,459\,8\,\text{J}\,\text{mol}^{-1}\,\text{K}^{-1}$, the gravitational acceleration g is somewhere around $9.81\,\text{m}\,\text{s}^{-2}$, and the molar mass M of Earth's air is $0.028\,964\,4\,\text{kg}\,\text{mol}^{-1}$. Thus, for many applications, (3.26) can be simplified to

$$f_\psi(\Delta\rho_{z,t}^i) = \psi_{\text{ref}} \left(1 - \Delta\rho_{z,t}^i \cdot 9.6 \cdot 10^{-4}\right)^{0.1208} \ . \tag{3.28}$$

As (3.27) is a relative model, the height h_0 (cf. (3.26)) of the reference is set to $0\,\mathrm{m}$, as it is unknown in which floor of the building a pedestrians initially resides.

In a scenario where the setup time of the ILS is critical and each floor of the building is almost the same height, we can further simplify the conversion by using

$$f_\psi(\Delta\rho^i_{z,t}) = \Delta\rho^i_{z,t} \cdot c^\psi_{\text{ref}} \tag{3.29}$$

This assumes a constant conversion factor c^ψ_{ref}, the same for each floor, which gives the pressure change per meter. It can be determined, for example, from a few initial floor change records [Fet23b].

3.4.2 Limitations and Discussion

As mentioned above, barometers are currently only available on high-priced smartphones. Applications that rely on the bring-your-own-device principle must expect that a large portion of their users will not have this sensor installed. The use of a barometer is also not recommended for some industrial scenarios where the air pressure in the building varies greatly for structural reasons, e.g. due to air intake systems, cold rooms or humidification units.

The method presented above behaves differently depending on the movement model that is being used. In the case of discrete floor changes due to particles, there will inevitably be a delay as the particles make "jumps" between floors while the barometer measurements remain continuous. This is also called a mixed discrete/continuous model [DDN01]. Especially in long staircases over several floors, large errors can occur which can only be compensated over a longer period of time and, if necessary, by other sensors [Ebn15].

Elevators present a special challenge. Their behavior varies greatly depending on the design, building or country. In addition, they often contain a separate fresh air supply and air conditioning. When entering the elevator, the pressure is equalized with the floor; during the ride, this pressure usually remains constant and only changes when arriving at the destination floor. As a result, particles may not be sampled to the desired floor, especially during long elevator rides, and the system could only make a correct estimate using other sensor information. This behavior has been observed and investigated in several papers on the topic [Ich15; Mur14; VN20; Ye16].

To solve this problem, the authors of [Ye16] use a crowd-based fingerprinting approach. Their goal is to identify the floor level the user is on without absolute knowledge. For this purpose, a fingerprint map is continuously updated by all users of the system. First, a calibration is performed to normalize the barometric readings between different devices carried by users.

Based on this, the pressure readings are then clustered to enable the fingerprinting approach. Consequently, the number of resulting clusters should be identical to the number of floors. While this approach provides good results, updating the fingerprint map requires people to be continuously present in the building and have the appropriate application installed. For many practical applications, such an approach is not a viable option. The high error potential of the barometer and the risk that a smartphone does not have the sensor integrated at all, often makes fusion with other sensors necessary. The next section presents an alternative approach.

3.5 Activity Recognition

Detecting a person's physical activity has many use cases, making it an exciting research topic. One prominent example is the recognition of fitness exercises, such as push-ups or squats, using sensor data from a smartwatch [Ish21]. Another example is the recognition of everyday activities such as brushing teeth, taking medication, or sleeping, which has become the focus of medical applications, enabling entirely new treatment strategies [SKG16]. The ubiquity of wearable devices like smartphones and smartwatches as well as their ever-increasing computing power enable highly attractive use cases for a broad mass of potential users. A good introduction and a very detailed overview of the field of activity recognition with wearable sensors is provided by [Jor19] or [LY13].

Activity recognition can also be used as an additional source of information in the field of indoor localization. It can be applied to distinguish between types of locomotion, for instance if the pedestrian climbs a staircase, is in an elevator, or is not moving. Furthermore, if knowledge about the building map is available, conclusions can be made about the location, e.g. if a stair ascent is detected, there is a very high probability that the pedestrian resides on a stair. The authors of [Zho15] have adopted such an approach. They distinguish between location and non-location related activities. The latter is used to update a PDR approach with the activities of walking and making turns. Location related activities are for example taking the stairs or the elevator. With this knowledge the nodes of an underlying graph-based road network are assigned a weight according to the pedestrians possible location based on the detected activity within the building. In the experiments, this method was able to obtain an error on average of less than $3\,\mathrm{m}$ for an online localization and $1.3\,\mathrm{m}$ for the offline calculation. The limiting factor in this approach is the assumption that all activities take place on one node of the graph so that a weight can be assigned to it. An interpolation between an estimated (free) position of the activity and the node would be a conceivable improvement here. Although the above results are very promising, there are very few ILS in the literature that take activity detection into account.

Imagine an application scenario in a museum, where visitors typically move between exhibits at a slow pace. They therefore spend most of their visit standing in front of them. As discussed in Section 3.3, the sensor models for step detection evaluate based on a constant value for the step length μ_{step}. When standing, μ_{step} needs to be reduced to near zero so that particles that did not move in the movement model are given a high weight. The same is true for climbing stairs. Here μ_{step} has to be adjusted to about half of what it is for walking on a flat surface. Activity recognition of typical locomotion activities such as walking, ascending stairs, descending stairs and standing can provide appropriate values for μ_{step} as well as other parameters used in PDR. Another use is as a standalone (virtual) sensor. If the ILS includes the knowledge of the floorplan of a building, the current activity of the pedestrian can also be used as input for a probabilistic evaluation model. Again, the movement of the particle between time t and $t-1$ is analyzed and compared to the current activity [Fet18]. For example, if a particle is moving up a staircase and the currently detected activity is ascending stairs, it can be evaluated accordingly.

The system proposed here imposes some conditions on the integration of activity recognition. First, it must be independent of the position of the smartphone. In particular, we distinguish between carrying the smartphone in the hand and in the pocket. Furthermore, there should be only a very short delay between the execution of an activity and its recognition. Since we assume a live system, the later an activity is detected, the later this information is integrated into the observation. Temporal backdating may be possible, as we will see later in this thesis, but it requires special methods and restrictive assumptions. Attention must also be paid to the performance of activity recognition, since it is only one part of the overall system and we have the claim that everything should be computable on a smartphone. Finally, high stability and a low false positive rate are required. On the one hand, the activity recognition has to be trained in a general context, i.e. the recognition has to give good results for all possible people and is not trained for a specific person. Second, falsely detected activities have a greater potential to throw the localization system off track than an unrecognized activity [Ebn20; Fet18].

3.5.1 Locomotion Activities

Activity recognition can be seen as a classical problem of pattern recognition in the field of machine learning. Thus, the first important step is to analyze the given data and to find salient features, e.g. cyclic characteristics or local extrema. These features should not only describe the activities to be captured, but also allow to distinguish between them.

In Figure 3.9, the raw data from the IMU's accelerometer and gyroscope are plotted for the typical locomotion activities of standing, walking, and ascending and descending stairs. These

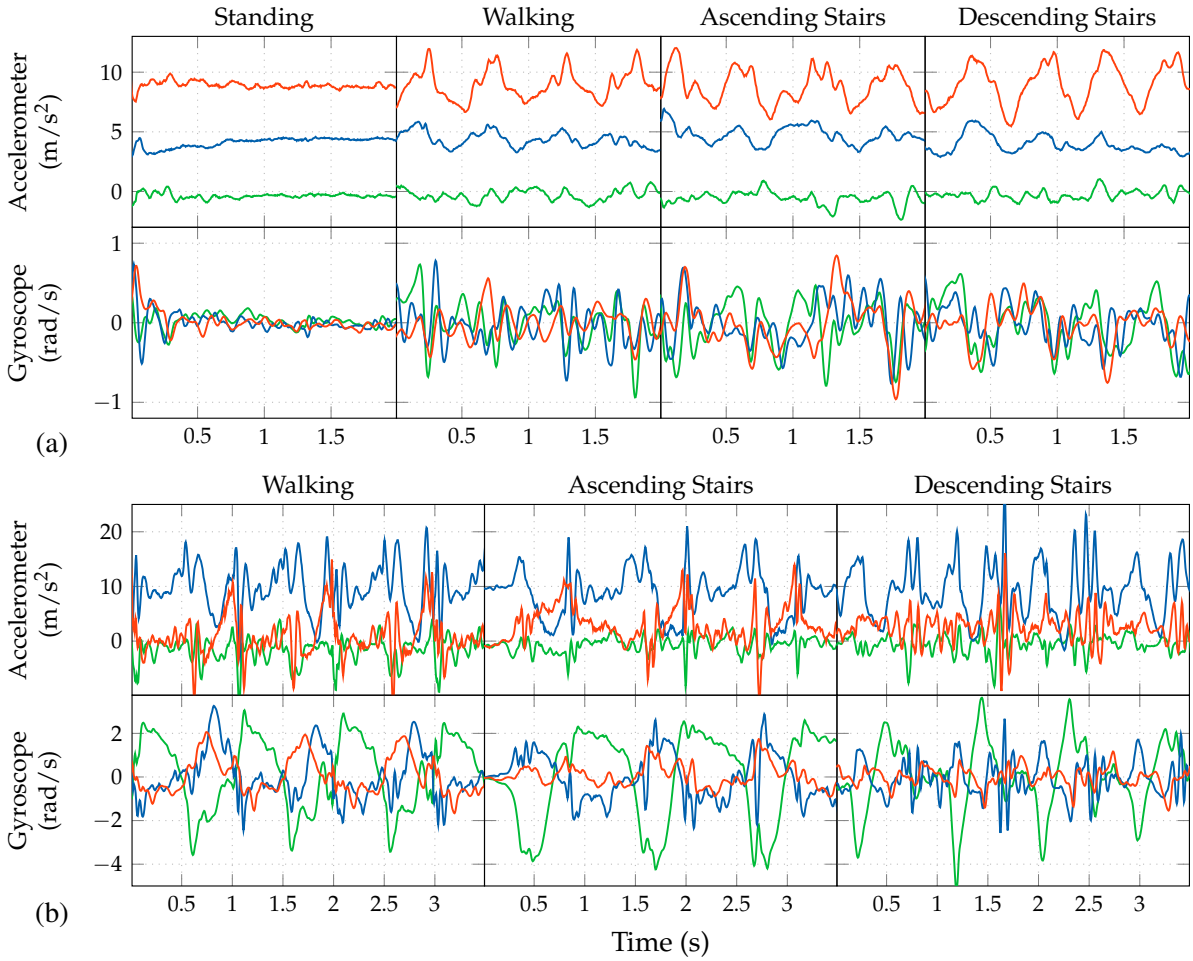


Figure 3.9: Raw data from the IMU's accelerometer and gyroscope for different activities in the handheld case (a) and the pocket case (b). Each sequence shows data for the x (green), y (blue) and z (red) axes in the local coordinate system of the recording smartphone (cf. Figure 3.7). The graph and data are taken from our paper [Ebn20].

are the activities we are investigating for use in an ILS as part of this work. It can be seen that standing does not seem to have any remarkable features. Its main characteristic is rather the lack of any distinctive measurement sequences, cf. no significant movement action of the user. Thus, we did not include standing in the Figure 3.9b, since the data is almost identical between the hand and the pocket case. An example could be the scenario of a museum visit mentioned above, where there are long periods of standing that additionally include the act of looking around.

The walking activity can be recognized by its characteristic periodic behavior. In the case of the hand, it is mainly caused by the up and down movement of the upper body during walking. This is mainly measured by the accelerometer and results in a peak for each step taken. In contrast, for the pocket case, the periodic behavior is found in the gyroscope measurements

(cf. x-axis in Figure 3.9b), since a rotation of the thigh parallel to the walking direction can be observed relative to the pocket. It is interesting to note that the frequency of the detected period is about half that of the hand case measured by the accelerometer. This is because in the pocket case only the walking motion of one leg is measurable, while in the hand case each step produces a significant acceleration. The concrete value of the frequency of a period depends not only on the position of the smartphone but also on the gait of the pedestrian and can therefore vary greatly from person to person [Ebn20].

Using the barometer to detect walking on stairs results in an approach similar to the one presented in Section 3.4. For some fixed window, one would observe the delta between the oldest and current pressure readings and compare it to some threshold. The sign of this delta then gives the direction, i.e. whether the pedestrian is going up or down the stairs. If no barometric sensor is available, the gait characteristics must be examined again. Contrary to the normal gait, the maximum inclination of the thighs increases during stair climbing because they have to be lifted to perform a climbing movement. This can be observed particularly well in the gyroscope data of the pocket case (see Figure 3.9b). Here the shape of the angular velocity differs between ascending and descending. The latter results in sharper and shorter peaks, because the maximum inclination of the thighs is much lower, since most of the rotation is absorbed by the bending of the toes on the previous step [Ebn20]. For the hand case, it is difficult to distinguish between walking on a level surface and walking on stairs. As can be seen in Figure 3.9a, there is no obvious constant visible difference in either the gyroscope or accelerometer measurements. The only thing that might be visible is a slightly faster frequency as well as a variation in the shape of the acceleration. However, as mentioned above, the frequency shown is not necessarily a reliable characteristic, as it can vary from person to person.

However, what the example data stream from Figure 3.9 shows is that due to the cyclic nature of locomotion, continuous detection of activities is possible within a live system. The task of activity recognition is therefore to classify a continuous data stream. For this purpose, the sliding window approach is used again, which considers only a temporally limited segment of the data. For each window, a decision is made as to which activity is reflected by this data. Thus, the detection rate depends directly on the window size. A window that is too short may not contain enough information to allow reliable classification, and a window that is too long may ignore short sequences of activity, such as a very short staircase. Finally, the size of the window depends on the person and their speed of movement. The faster a person moves, the smaller the window size can be, and vice versa. In the literature, window lengths between $2\,\mathrm{s}$ and $3\,\mathrm{s}$ are suggested as a fair compromise and could give reliable results in corresponding experiments [Ebn20; Pre09; Wu12].

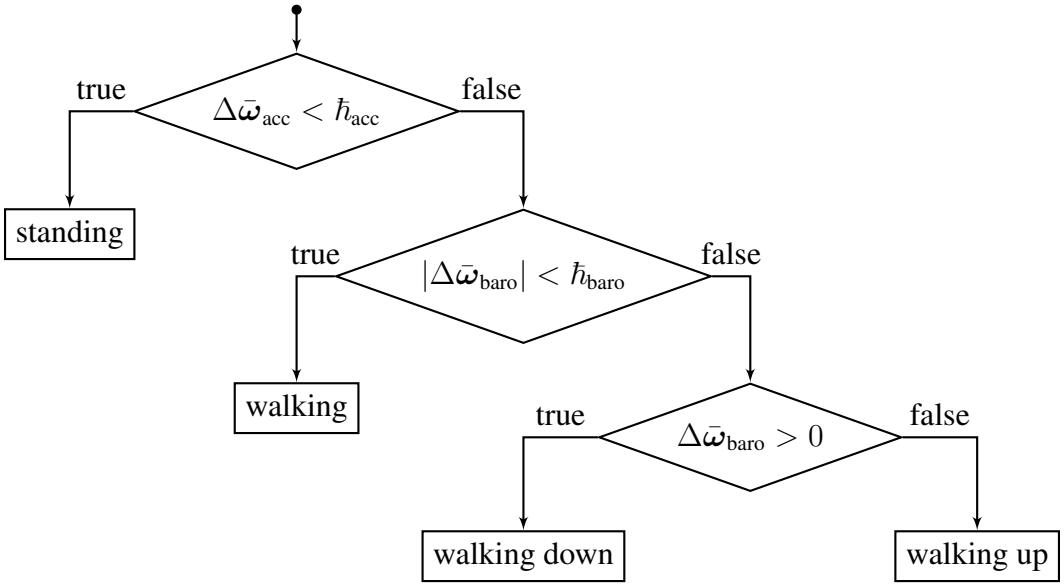


Figure 3.10: Activity recognition by means of a decision tree based on acceleration and barometric pressure values. $\Delta\bar{\omega}_{\text{acc}}$ and $\Delta\bar{\omega}_{\text{acc}}$ are given by the arithmetic mean of two different fixed size windows, one short and one long, containing a set of the most recent sensor readings. An activity is given for each new measurement of the smartphone's barometer.

3.5.2 Threshold Decision Tree

The first approach presented is a simple threshold-based activity recognition using the barometer and the accelerometer. Similar to Section 3.3, we use a rectangular window function, which can easily be implemented as a circular buffer of fixed size. Two windows are defined for each sensor, a short window $\boldsymbol{\omega}_s$ and a longer window $\boldsymbol{\omega}_l$. The window size of $\boldsymbol{\omega}_s$ is optimally chosen between $0.3\,\text{s}$ to $0.6\,\text{s}$ and that of $\boldsymbol{\omega}_l$ between $2\,\text{s}$ to $4\,\text{s}$, depending on the application, the sensor frequency and other information like the pedestrian's walking speed, if available. The barometer data is slightly smoothed with a low-pass filter and then added to the windows. To get a one-dimensional structure, the accelerometer is first rotated to the ENU frame and then the normalized magnitude $|\hat{a}|$ is calculated (see Section 3.3). The respective thresholds $\hbar_{\text{acc}}$ and $\hbar_{\text{baro}}$ are then compared to the moving average $\Delta\bar{\omega} = \bar{\omega}_l - \bar{\omega}_s$, where $\bar{\omega}$ is the arithmetic mean of a window. Past experiments have shown that the values $\hbar_{\text{acc}} = 0.015\,\text{m}\,\text{s}^{-2}$ and $\hbar_{\text{baro}} = 0.042\,\text{hPa}$ provide a good compromise between robustness and flexibility [Fet18]. For each new incoming barometer reading, an activity is then given using the decision tree structure illustrated in Figure 3.10.

Despite its simplicity, this approach reports surprisingly good recognition rates of $81.5\,\%$ for walking, $84.3\,\%$ for ascending stairs, and $82.1\,\%$ for descending stairs. Only standing is very poorly recognized, with a recognition rate of only $51.4\,\%$. [Fet18]. This is mainly due

to the way standing is defined. The data set from which the recognition rates were derived represents a realistic scenario in a museum. The subjects were always standing in front of an exhibit and were asked to behave as naturally as possible. This meant that the smartphones were sometimes placed in their pockets and the subjects had some freedom to move around in front of the exhibit. This slight movement, i.e. small steps from left to right or turning around their own axis, was still labeled as standing. Since the threshold-based approach, as can be seen in Figure 3.10, uses only the acceleration of the smartphone to detect standing, it can only be used to detect a real standstill of the device. This again raises the question of how to define standing in the first place. The unsatisfactory answer: it depends on the scenario. Thus, in the case of a museum, this approach is not optimal and other methods should be considered [Fet18].

3.5.3 Analytical Transformation

A more sophisticated approach that also takes into account a higher versatility of standing has been presented in our work [Ebn20]. In contrast to the above, it does not use the barometer, but only the accelerometer and the gyroscope. The main reason for this is the aforementioned limitation that barometers are primarily installed in the higher-priced smartphones, and thus a large number of devices would not even be considered for activity recognition using the threshold-based decision tree. The approach in [Ebn20] also introduces outlier detection. It detects if the user starts to mess around with the smartphone, e.g. by flipping the device several times, or if an actual activity sequence is taking place. This is achieved by an initial one-class classifier. It decides whether a feature vector is an outlier or contains an activity sequence. This is done by training a one-class *support vector machine* (SVM) with a *radial basis function* (RBF) kernel [Bis06]. The advantage of this kernel is that it has only one free hyperparameter, the kernel scaling parameter γ. The optimal configuration for γ, i.e. the highest classification accuracy, can then be optimized by a gradient descent over negative and positive samples of a labeled data set. As shown in the experiments of [Ebn20], a classification accuracy for outlier detection of $95.5\,\%$ could be achieved with $\gamma = 0.005$.

After filtering outliers, the actual activity detection approach is performed. In [Ebn20] three state-of-the-art approaches, namely analytical transformation, codebook and statistical features, were introduced and tested. While all of them achieve an accuracy above $90\,\%$ for the pocket case, the results for the hand case are very different, with the analytical transformation achieving the highest overall accuracy of $84.1\,\%$. The codebook approach achieves $76.7\,\%$ and the statistical features only $69.9\,\%$ for the hand case. As one can already guess from the previous discussions, especially the activities of ascending and descending stairs are difficult to recognize in the hand case, because the raw data does not differ much from walking on a flat surface (cf.

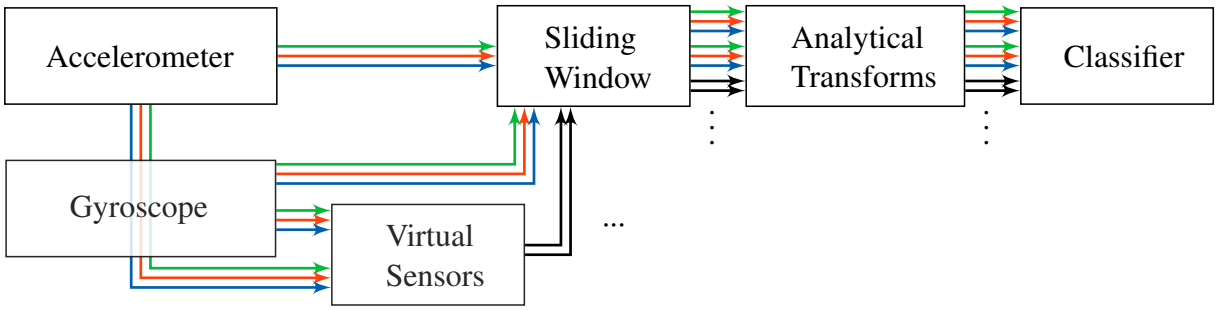


Figure 3.11: Pattern recognition pipeline for the analytical transformation approach shown as a flowchart. Raw sensor measurements are provided by the IMU's accelerometer and gyroscope on three channels each, given by the respective axes x (green), y (blue), and z (red). In addition, channels with artificial data computed by virtual sensors (black) are considered. An analytical transformation is then performed on each channel for dimensionality reduction. The resulting feature vector is then used for classification. Figure taken from [Ebn20].

Figure 3.9a). Only the analytical transformation approach is able to achieve satisfying results of $76.5\,\%$ for ascending stairs and $90.9\,\%$ for descending stairs, while the other approaches show results below $50\,\%$. If we refer to the threshold-based approach presented earlier, the results for the analytical transformation are quite similar. However, they differ greatly for the detection of standing activity. While the threshold-based approach reports an accuracy of $51.4\,\%$, the analytical transformation achieves $93.7\,\%$ for both the hand and pocket cases.

The analytical transformation approach, as the name suggests, learns a linear transformation suitable for reducing the dimensionality of a feature vector. Besides raw sensor data, a feature vector can also be processed data from virtual sensors. By reducing the dimension, one hopes for a better separation between the classes (activities), i.e. similar data points will coincide and thus form a common class in the best case. With such a reduction, at first glance, information is lost as the feature vector becomes smaller. However, if we look again at Figure 3.9, we can assume that the available measurement data of the IMU show a strong correlation. A suitable analytical transformation is able to reduce the correlation without significant loss of information, i.e. the activity described by the data. This is achieved by transforming the feature vectors into a low-dimensional space so that the variance of the data is maximized. It is now assumed that the majority of the information necessary for classification, i.e. the appropriate separation of the respective activities, is not only preserved, but that the projection creates data clusters that allow a good separation. Classical methods for dimension reduction are e.g. the *principal component analysis* (PCA) or the *linear discriminant analysis* (LDA).

The pattern recognition pipeline for the analytical transformation approach is shown as a flow diagram in Figure 3.11. As noted above, the approach uses only the IMU's accelerometer and gyroscope as sensor inputs, already rotated into the ENU frame. Each results in three raw

data channels given by the respective axes x (green), y (blue) and z (red). Again, a sliding window approach is used to extract sequences per channel. For example, with a window size of $2\,\mathrm{s}$ and a sampling rate of $50\,\mathrm{Hz}$, the sequence and thus the feature vector before transformation yields 100 samples, which can also be viewed as a vector with the same number of dimensions.

In addition, virtual sensors are added to further increase separability. They compute metrics based on the raw sensor data and thus generate additional data channels for the pipeline. For the accelerometer data, the magnitude (+1 channel), standard deviation per axis (+3 channel), root mean square per axis (+3 channel), and inclination (+2 channel) are calculated, and for the gyroscope data, the magnitude (+1 channel) and standard deviation per axis (+3 channel) are calculated. As discussed in Section 3.3, the magnitude is invariant to rotations and thus independent of the actual orientation of the smartphone. The standard deviation and the root mean square are chosen because, as shown in [Fig10] and [Sun10], they allow a good separation between resting and non-resting activities. It is important to note that magnitude and inclination can be computed on a per-sample basis, i.e. for each new incoming sensor measurement, so there is no delay between raw and virtual data results. In contrast, standard deviation and root mean square require multiple samples and therefore use a small sliding window $\boldsymbol{\omega}_{\text{virt}}$ with a size of $|\boldsymbol{\omega}_{\text{virt}}| \approx 0.3\,\mathrm{s}$ that moves over the raw input. This causes these virtual sensors to have a small delay of $\Delta t = \frac{\boldsymbol{\omega}_{\text{virt}}-1}{2}$. Otherwise, they behave the same as the real sensors and are treated the same in the pipeline.

For each data channel, an individual analytical transformation is trained to reduce the dimensions, i.e. six for the raw data and seven for the virtual data. A per-channel transformation has the advantage that the axes and sensor-specific information is preserved, while things with low variance, such as sensor noise and redundancies, are efficiently removed by the reduction. The transformation method used is PCA, as it has been shown that PCA often outperforms LDA in multi-class classification problems [MK01]. In particular, cyclic data with simple recurring structures can be described very well by eigenvectors. Figure 3.12 shows the five eigenvectors with the largest eigenvalues for the accelerometer, which supports the previous statement. The authors of [Kha10] suggest that the activity problem should be solved using non-linear approaches such as *kernel discriminant analysis* (KDA). They argue that, compared to linear methods such as LDA, a KDA is able to maximize the between-class variance and minimize the within-class variance by deriving non-linear discriminative features. However, such approaches can be impractical for large datasets or suffer from high memory or computational complexity as they apply the kernel trick [Mik99]. The introduction of virtual sensors does not change the linearity of LDA, but gives the overall system non-linear separation properties [Ebn20].

Training the analytic transformation, i.e. finding the eigenvectors, can be explained conceptually by learning pattern sequences, as shown for the accelerometer in Figure 3.12. When

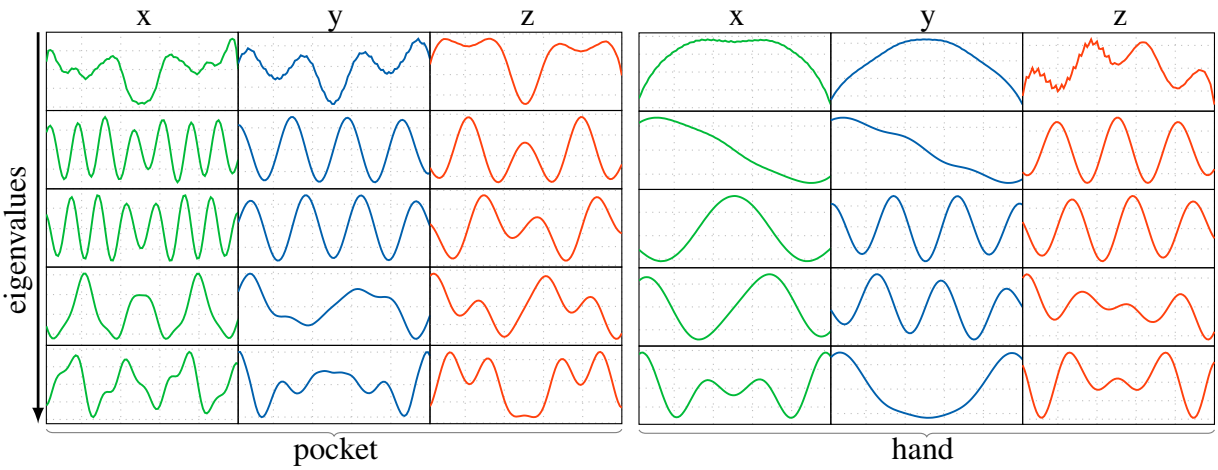


Figure 3.12: The five eigenvectors with the largest eigenvalues, decreasing from top to bottom, for the accelerometer in the hand and pocket cases. The figure is taken from the experiments of [Ebn20] using a window size of 2 s.

applied to live sequences, the result is a histogram-like feature vector that describes the prevalence of each of the learned pattern sequences in the linear combination. For training, a labeled data set is used that assigns the user's activity to each time frame of the sliding window. A corresponding PCA is then trained per channel, while a single feature vector is given by the samples of that channel included in the window. This results in 13 (6 raw channel + 7 virtual channel) individual linear transformations. The number of learned pattern sequences, i.e., the number of dimensions to be reduced, depends on the size of the eigenvalues describing the variance along the corresponding eigenvector axes. A scree test performed in the experiments of [Ebn20] indicates that a reduction of at most 5 dimensions is reasonable. More dimensions add little statistically significant information, indicating redundancy caused by sensor noise and other undefined behavior. This results in a feature vector of size 5 per sensor channel. For input to the classification, all these feature vectors are concatenated, resulting in a single large feature vector of size 13 channel × 5 dimensions = 65 per window, labeled with an activity in the training phase.

Finally, an SVM with RBF kernel is chosen as the classifier. It is well known that SVMs are a binary (two classes) classification method and therefore a combination of multiple SVM classifiers must be used to solve multi-class problems. Several strategies and approaches of such combinations can be found in the literature [DK05; WX14]. For the presented activity recognition, the so-called *one-vs-one* strategy was chosen [HL02]. A binary classifier is constructed for each combination of two activities. Calculating the binomial coefficient over the set of four recognizable activities then results in six different SVMs. Multiclass classification is then performed using max-wins voting (discrete pairwise comparison), i.e., the class to which

the current feature vector has been assigned most often is selected. While this provides the best possible class and thus the most appropriate activity for the current observation, we are not able to make profound statements about the other activities. Especially in the localization scenario presented here, it would be desirable to obtain a probability estimate for each of the classes instead of a discrete decision. This would mean that for each observation, a probability for each activity would be the result of the classification. This allows more versatile evaluation models, e.g. particles can be evaluated not only whether they did the most probable activity or not, but exactly with the applicable class probability of each activity. Especially in the case of the hand, climbing stairs and walking are sometimes only slightly different, leading to more false positives. A continuous classification avoids a discrete break and allows at least a part of the particles to model the supposedly correct activity for the approximation of the posterior. A popular extension of the *one-vs-one* strategy to achieve this is *pairwise coupling* [HT97]. The basic idea is to interpret the output of each binary SVM as the probability of the positive class and then estimate the posterior $P_i = p(\Omega_i \mid \boldsymbol{\omega})$, where Ω_i is the class label and $\boldsymbol{\omega}$ is the observation window.

3.5.4 Activity Evaluation Model

Considering the recognition approaches above, each observation o_t contains a parameter $\boldsymbol{\Omega}_t$, which is either a discrete decision for a single activity or a vector containing probabilities for each activity. Which type it is depends, on the one hand, on the decision which approach to use and, as always, on the scenario at hand. At the same time, depending on whether $\boldsymbol{\Omega}$ is discrete or continuous, we need a corresponding probabilistic sensor model $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}}$. Similar to the models presented in Section 3.3, we add the knowledge of the previous state $\boldsymbol{X}_{t-1}^i$ to be able to make decisions based on the particle's movement modeled by the transition step.

For the discrete case a simple distribution of the form

$$p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}} = \begin{cases} \xi & \boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{\Delta\rho_t}^i \\ 1-\xi & \text{otherwise} \end{cases} \tag{3.30}$$

can be chosen to provide a probability given by ξ. Here $\boldsymbol{\Omega}_t$ is again the current activity given by the recognition method, and $\boldsymbol{\Omega}_{\Delta\rho_t}^i$ is the activity that best corresponds to the change in position $\boldsymbol{\rho}$ of a particle indexed by i between time $t-1$ and t. The probability ξ depends mainly on the accuracy of the activity recognition, but also on other effects like timing errors or the current quality of the posterior approximation. The choice of ξ is thus a trade-off between how much confidence should be placed in activity recognition and how much other possible system states

should continue to be represented. In many cases, $\xi = 0.75$ is a good choice, since it preserves enough margin for misclassification, but sufficiently rewards particles that behave according to activity.

The particle's activity $\boldsymbol{\Omega}^i_{\Delta\rho_t}$ can be defined according to

$$\boldsymbol{\Omega}^i_{\Delta\rho_t} = \begin{cases} \text{standing} & \Delta\rho_t \leq d_{\text{standing}} \\ \text{ascending} & \Delta\rho^i_{z,t} \geq h_{\text{stairs}} \\ \text{descending} & \Delta\rho^i_{z,t} \leq h_{\text{stairs}} \\ \text{walking} & \text{otherwise} \end{cases}, \tag{3.31}$$

where $\Delta\rho^i_{z,t}$ denotes the change in z-direction (height) and $\Delta\rho_t$ the absolute distance covered between time t and $t-1$. The height threshold h_{stairs} can be set to a small value such as $0.05\,\text{m}$, since its main purpose is to avoid numerical rounding errors. The sign of $\Delta\rho^i_{z,t}$ therefore determines whether the particle is ascending or descending. The distance threshold d_{standing} depends on the movement model and scenario used. As already discussed, it is difficult to define which actions are generally considered standing. In the past, a movement of less than $d_{\text{standing}} = 0.1\,\text{m}$ per update was appropriate, otherwise walking is considered [Fet18]. If floorplan information is available, e.g. using a spatial model that identifies floors, stairs, and other elements, the sensor model in (3.30) could be freed from the previous state $\boldsymbol{X}^i_{t-1}$ by choosing $\boldsymbol{\Omega}^i_{\Delta\rho_t}$ according to the particle's location on the spatial model. For example, if the spatial model reports that a particle is on a staircase, and $\boldsymbol{\Omega}_t$ is ascending a staircase, the particle will be assigned a weight of ξ. Of course, this further simplifies (3.31) by removing the ability to distinguish between standing and walking, and ascending and descending stairs. However, if some other activity recognition is used, such as distinguishing between walking, climbing stairs, riding an elevator, and taking an escalator, this approach can add real value to the ILS.

Finding a sensor model for the continuous case is straightforward, since $\boldsymbol{\Omega}$ is a vector containing a (normalized) probability for each activity. Thus, each particle is assigned a weight according to its current own activity $\boldsymbol{\Omega}^i_{\Delta\rho_t}$ as defined by (3.31). As we will see in the upcoming section, the motion model on a spatial model can also benefit from a continuous activity recognition. For example, it can be decided how many particles can and cannot be sampled on a staircase, making some areas of the building more likely and others less likely.

3.5.5 Limitations and Discussion

The time between the execution and the actual recognition of an activity depends on the sensitivity of the thresholds and the chosen window size. For the experiments conducted by [Fet18]

to evaluate the threshold decision tree, an average difference (lag) of $2.96\,\mathrm{s}$ with a standard deviation of $1.09\,\mathrm{s}$ over all walks could be reported. For use in a sensor fusion, this means not only delayed detection, but also a temporal discrepancy with other sensor measurements such as Wi-Fi or step detection. As a consequence, the overall uncertainty of the system increases, i.e. the probability density becomes broader. For example, if a movement model is used that only allows floor changes at staircases, the particle set could have passed the staircase before these activities are detected due to a delay in detecting the activities of ascending and descending. This would mean that not enough particles of the total set would be sampled in the area of the stairs and thus no floor change could be modeled. This lack of sufficient particles for a correct approximation of the system-describing density function is called sample impoverishment and is addressed later in this thesis.

As discussed in the known limitations of Section 3.3, human movement patterns are diverse and generalization is difficult. For a classifier approach, the versatility and the quality of the training data set are particularly important. For the quality there are known factors like number of data, distribution of classes, reliability, noise, feature representation, skewness and many more. Versatility is often determined by the number of subjects. In the case of activity recognition, it is necessary to have an evenly distributed cross section across different age groups and modes of locomotion. Unfortunately, there are hardly any publicly available data sets that sufficiently cover all factors. For the pocket case, MobiAct [Vav16] is a dataset that contains the activities of 66 subjects on a total of 3200 recordings. However, there is a particular lack of recordings from people with gait or visual impairments. In the case of the hand, the situation is even worse. To the best of our knowledge, there are no data sets available for this case at all, which is why a separate data set with 8 subjects between 20 and 30 years of age was recorded in [Ebn20]. Of course, this is not very representative and a limiting factor for the classification.

Another factor that limits use within a localization system is the inability to detect elevator or escalator travel. This leads to undefined behavior at these locations. Interestingly, thanks to the barometer, the threshold-based approach is able to detect an elevator ride quite stably, since its threshold has no upper or lower limit. Other sensors, such as gyroscopes or accelerometers, are not as good at detecting this activity in a stable way, which is a major drawback of the analytical transformation method. Nevertheless, there are approaches that try to perform detection using only these two sensors. For example, the authors of [YKK13] use a very simple approach that only filters for continuous positive or negative values within a $1.2\,\mathrm{s}$ sliding window. The acceleration is normalized, and if a window contains only positive values, it detects an accelerating elevator, and if it contains only negative values, it detects a decelerating elevator. The sequence in between is the entire elevator trip. Amazingly, the authors report a detection rate of $98\,\%$ with this very simple approach. However, it is only able to detect elevators. Therefore, it

does not take other activities into account, so it is not clear how the raw data of e.g. ascending stairs and riding an elevator differ and whether they might have similar characteristics. Theoretically, and as shown by [YKK13], the accelerometer should detect the vertical motion during the acceleration and deceleration of the elevator quite well. Therefore, it is often more promising to use additional sensors, such as the magnetometer or the barometer mentioned above. A magnetometer does not necessarily allow detection of vertical motion, but the magnetic properties of an elevator can be used to separate the stairs from the elevator [KL16], which in turn allows different methods to be applied to the accelerometer data.

Chapter 4

Movement Models

Choosing the transition density $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$ as the importance distribution, which yields the CONDENSATION particle filter, allows the use of a variety of movement models. To recap, the transition predicts a new state $\boldsymbol{q}_t$ given the last state $\boldsymbol{q}_{t-1}$ and as described in (2.72) also the last observation $\boldsymbol{o}_{t-1}$. So this prediction can be interpreted as a statement about the future of the system using only the knowledge of the predecessor. The observation $\boldsymbol{o}_{t-1}$ is integrated to provide useful sensor information like the last known activity or number of steps. A direct use of them naturally leads to a delay, since the execution of an action and its recognition are staggered. Using a suitable implementation strategy can compensate for this, as already discussed in Section 2.5.1 and Section 3.3.

The goal of a movement model in self-localization is to reproduce the walking behavior of the pedestrian. In the context of particle filtering, this is done by updating the parameters, i.e. position $\boldsymbol{\rho}$, heading Θ, etc., of a particle $\boldsymbol{X}_t^i$, hereby simulating the process of sampling from the transition density yielding $p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_{t-1})$. The strict adherence to this assumption would additionally intensify the problem of sample degeneracy, since no new particles are created with a certain randomness, but only already existing ones are continued. This also means that during the transition the size of the sample set, i.e. the number of particles, does not change. For this reason, movement models usually contain additional strategies to remove particles that have, for example, made a completely unrealistic motion, such as walking through a wall, and sample a new one instead. In the simplest case this can be implemented by assigning this particle a zero weight and performing one resampling step per filter update to remove it, e.g. cumulatively over the total amount. Towards the end of this chapter we will see that such strategies reveal fundamental problems of the particle filter for the use in an ILS.

Nevertheless, the advantages of using movement models that prevent unrealistic behavior outweigh the disadvantages. They ensure that new particles are only sampled in areas that could

realistically be reached by a human being between two filter updates. By reducing particles at non-walkable locations, the accuracy is significantly increased [Bre17; Guo20]. Especially when using sensors that have a high uncertainty / error, a better position estimation results automatically by quasi-exclusion of these non-walkable locations. In addition to preventing walking through walls, movement models can also take into account other factors such as realistic movement speed, floor changes only at stairs, or can even make predictions based on known navigation paths [Ebn16].

For most of this, knowledge about the building is necessary. If available, it can be taken from one or more floorplans, showing the architectural features of a building as technical drawing. Our experience shows that in reality, building plans are often out of date within a very short time, since it is common to carry out construction work on a regular basis, but not to document it thoroughly. Historical buildings in particular, which have had several previous owners and countless construction phases, sometimes have faulty and very imprecise plans [Fet18]. However, surveying new buildings is complex and very expensive, which is why the movement model must be able to deal with such problems. From a probabilistic point of view, this uncertainty must thus be modeled accordingly.

In contrast, approaches like SLAM aim at a relative localization with simultaneous generation of the floorplan. Such approaches are of particular interest in robotics, e.g. they allow vacuum cleaner robots to move around in an unknown apartment and to perform further tasks such as cleaning zones or following planned paths with the help of the environmental information [TUI17]. SLAM can also be relevant for pedestrians, especially in time-critical scenarios such as firefighting operations. Firefighters can be located in initially unknown buildings, which is particularly useful in cases of heavy smoke or collapses [HB22]. Regardless of whether the floorplan is given or generated with SLAM, both approaches require a spatial representation to perform operations, such as determining whether a particle is on a staircase.

4.1 Spatial Models for Floorplans

At a high level, a spatial model is an attempt to give a generic spatial representation of a partially definable space or problem interpretable in this sense using topological, geometric, or geographic properties. This can include obvious tasks like cartographic visualization, e.g. the data structures underlying map-providers like OpenStreetMap, more abstract ones like statistical representation of supply chains or even studies of the placement of galaxies in the cosmos. For the context of indoor localization this can be narrowed down to a model that is capable of representing locations of arbitrary objects within indoor spaces. A geometric description of the

system entities, e.g. observations given by sensor models, consequently requires that the model expresses the location information using a coordinate system.

In addition to the pure representation of the building, spatial models are subject to further requirements within the use of an ILS. These can be user-centered context, such as possibilities for interaction with the environment, e.g. a door that can be opened via the smartphone or the integration of popular walking routes into the model to support realistic route planning for navigation [Ebn16]. The authors of [ARC12b] distinguish between two categories of requirements: *service-oriented* and *efficiency-related* requirements. The first includes the already mentioned representation of the indoor space. It further addresses:

- Navigation tasks to guide pedestrians. First, a path planning process is performed to find a path between the current location and the destination. This may require context-sensitive information such as user preferences, realistic walking paths, or other cost functions to avoid hazards. After that, dynamic processes monitor the current position and also keep track of certain events, such as emergencies, that would require adapting the predetermined path [Del09].

- Location-aware communication between entities such as sensors and users in smart spaces. Depending on the location, desirable information may be requested from the user, or a provider may invoke remote callback methods from potential participants. This imposes requirements on the resolution of the space, i.e. fine-grained models allow a more accurate representation of the space, e.g. coarse models.

- Activity oriented interaction with physical (e.g. stairs, door, elevator) or virtual objects (e.g. pictures, information screens) in the user's environment. Based on human activity recognition, the aim is to achieve goals such as matching spatial information with the current activity, e.g. climbing stairs and the spatial information whether the user is actually on a staircase.

- Behavioral analysis and activity monitoring of users of a ILS has many applications in a variety of scenarios. This often requires the ability to represent the spatial model in different resolutions or topological descriptions. For example, while in a conference room booking system it is only important whether the room is occupied or not, creating a heat map of the most popular walkways in a shopping mall requires a very fine-grained representation.

The efficiency-oriented requirements cover the classic technical and business aspects of designing, implementing, and operating a spatial model. They are

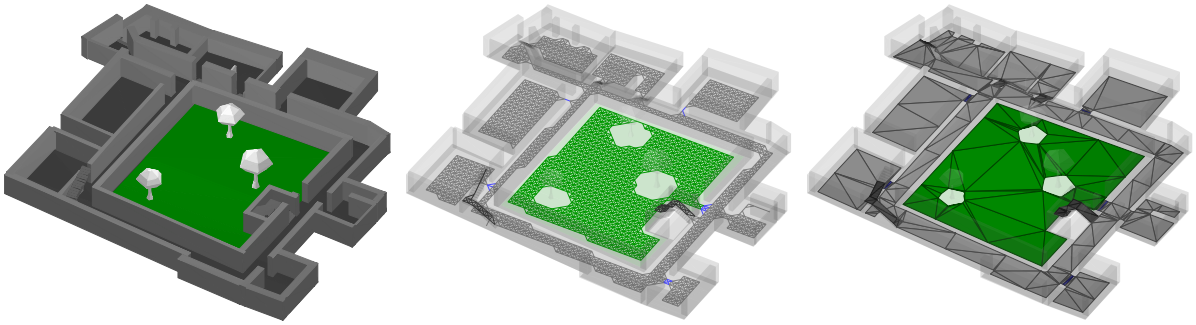
- The effort to design, create and update the spatial model. Modeling the indoor space in as much detail as possible generates large efforts, while a simple topological description does not require any survey work or map editors. The former usually also generates much more care and maintenance effort.

- The flexibility that the spatial model can be used for multiple purposes, different services and abstraction layers. This could be a common localization environment for robots, pedestrians and objects simultaneously. The arbitrary storage of location-dependent meta-information, such as previously visited locations or a work schedule, in the data structure may also be of particular interest.

- Especially ILS for smartphone-based pedestrian self-localization, performance is crucial. Operations such as path finding, metadata lookups, or other queries must be efficient and should not affect the user experience. For real-time systems that frequently compute new position estimates, the spatial model must not be a bottleneck.

- Good scalability places demands on memory requirements in particular, but also on overall resource consumption in general. It is therefore closely related to performance requirements. However, not only the ability to efficiently model the largest possible buildings, but also the dynamic loading and unloading of substructures can ensure efficient scalability.

In summary, this requires a versatile spatial model that is able to represent the indoor space in different levels of detail, store meta-information in an efficient way, perform complex calculations in a reasonable amount of time and still remain scalable. The authors of [ARC12b] again distinguish between two main classes: symbolic and geometric spatial models. Symbolic approaches model the environment using topological-based structures. This is often a simplification, so that only the most important information such as connectivity and accessibility between spatial units is retained. An example of a classic topological map is an underground railway network map as known from large cities such as Berlin or New York. The great advantage of such approaches is the ability to derive the relationships and hierarchy between individual entities (e.g. rooms or machines) of the indoor space. The location of an object is semantically represented by human-readable descriptions (e.g. a room name) and placed in a topological relationship. In principle, symbolic spatial models are therefore a more or less strong abstraction, in which the accuracy of a position is limited to the level of abstraction. A topological description of spaces can thus only provide positional information at this structural level. An arbitrary position in the building, which must be based on some coordinate system, cannot be represented in this way. For the purpose of self-localization in buildings, which is the subject of



(a) 3D floorplan (b) regular tessellated grid (c) irregular tessellated mesh

Figure 4.1: Floorplan and automatically generated spatial models for the ground floor of a museum building ($71\,\mathrm{m}$ x $53\,\mathrm{m}$). In (b) a coarse cell-size of $90\,\mathrm{cm}$ was chosen, resulting in 1700 squares. The grid is thus barely able to represent all doors and stairs, what requires for a finer resolution. In contrast, the mesh in (c) requires only 320 triangles and is able to accurately represent the complexity of the environment. However, the size of the triangles varies greatly, especially in tight areas or with complex obstacles. Figure is taken from our work [Fet18].

this thesis, these approaches are therefore less suitable. The interested reader can nevertheless get an overview of symbolic approaches in [Kan20] or [Gu19].

In contrast, geometric spatial models consider a geometric representation of space, such as grids or meshes, and are based on a coordinate system, usually euclidean or geographic. These approaches decompose the physical space into a finite number of non-overlapping cells, resulting in tessellation as illustrated in Figure 4.1 for the first floor of a museum. If cells of the same size and shape are used, e.g. a grid of squares, this is called regular tessellation. Irregular tessellation occurs when size and shape are not identical, e.g. arbitrary polygons. Although both types of tessellation are able to model only accessible areas of a building and exclude obstacles, irregular cells have the advantage of allowing an adaptive decomposition of space that is able to accurately represent the complexity of the environment under study (e.g. arbitrarily shaped obstacles) without greatly increasing the number of cells. A comparison of Figure 4.1b and Figure 4.1c shows that this is particularly the case for the trees in the center. With a regular approach, the accuracy of the representation depends on the resolution of the cells and thus grows quadratically in two dimensions, which imposes particular memory requirements on the smartphone. In the example of Figure 4.1b, only a coarse resolution of $90\,\mathrm{cm}$ per cell was chosen, what barley able to represent all doors and stairs. On the other hand, irregular models are very complex and require additional methods and data structures, for example to store meta-information in arbitrary areas or to determine realistic navigation paths.

In the upcoming sections, we will introduce two different movement model approaches, still answering $p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_{t-1})$. The first is based on a regular geometric spatial model for the representation of the building map using a graph of equidistant nodes, which equals a grid. The

other approach uses a navigation mesh based on irregular tessellated triangles. Both methods are interesting because of their mathematical properties and their widespread use in other research areas such as computer graphics. However, they place completely different demands on the underlying movement model, making the two different approaches necessary in the first place.

4.2 Random Walks on Grids

As seen above, a grid describes a regular tessellation of the surface of a floorplan using a series of contiguous cells. In this work, the grid is defined as consisting of square cells in the Euclidean plane, extended to include staircases. It can be described using a graph $G = (V, E)$, where each cell is delineated by its center, represented by a vertex $\boldsymbol{v}_i = \{\boldsymbol{\rho}, ...\} \in V$, where $\boldsymbol{\rho} = (x, y, z)^T$ is again a 3D location. Undirected edges $\boldsymbol{e}_{i,j} = \{...\} \in E$ connect $\boldsymbol{v}_i$ to its eight neighbors in horizontal, vertical and diagonal direction [LCR10]. Both, $\boldsymbol{v}_i$ and $\boldsymbol{e}_{i,j}$ are able to store additional semantic or meta information like a room number or if they belong to a stair. The size of the cells is defined by the grids granularity g_s, thus providing the maximum resolution. A very coarse granularity of $g_s = 1\,\mathrm{m}$ is often unable to represent narrow areas such as doors, while a fine one like $g_s = 0.1\,\mathrm{m}$ is often able to model every nook and cranny of the building, but results in a large $G = (V, E)$. The movement model is now based on the idea that the particles located in the cells move along the edges from node to node.

To obtain the result shown in Figure 4.1b, the grid is first superimposed on the floorplan, both being in the same coordinate system. Walls, doors and other obstacles in the floorplan, are called occupied space. The rest of the interior is free space. The cells of the grid are now grouped into these two categories. If a cell intersects an obstacle like a wall, it is considered occupied. On the other hand, a cell in the middle of a corridor is a free cell. This representation is called an occupancy grid and is used in particular in robotics with SLAM [Thr03]. In order to obtain only the walkable areas that we are interested in, the occupied cells are ignored and G is constructed only from the vertices of the free cells. Edges that would point to an occupied cell are also removed.

4.2.1 Origin of the Floorplan

So for creating the spatial model G a suitable floorplan is required first. As explained at the beginning of Section 4, there are different approaches for obtaining and describing the building's architecture. In the best case, digitised plans in some CAD format already exist, but as reality shows, outdated paper plans, escape plans or simple drawings often have to suffice. Camera-based approaches using point clouds or crowed-sourced SLAM methods, automatically gener-

```
<map w="70" h="50">
  ... <!-- Meta information like geo-coordinates, address, etc. -->
  <floors>
    <floor atHeight="0" height="3" name="first floor" group="0">
      <outline>
      ... <!-- Polygons describing the floor ground-->
      </outline>
      <obstacles>
        <wall material="3" x1="1" y1="25" x2="20" y2="25" w="1.2">
          <door material="2" x="0.3" w="0.9" h="2"/>
          <window material="2" x="0.3" y="1" w="1" h="1"/>
        </wall>
        ... <!-- Add more walls or other obstacles with polygons -->
      </obstacles>
      <stairs>
        <stair type="0">
          <part x1="25" y1="10" z1="0" x2="25" y2="13" z2="1" w="1.5"/>
          ... <!-- A staircase consists of several parts -->
        </stair>
        ... <!-- Multiple stairs per floor -->
      </stairs>
      ... <!-- More floor entities like beacons, POIs etc. -->
    </floor>
    ... <!-- Multiple floors per building -->
  </floors>
</map>
```

Listing 4.1: Topological floorplan description in simpleLoc XML format.

ating the floorplan, are also frequently discussed in the literature [Kan20]. Whatever the basis, the floorplan needs to be converted into a suitable format. In addition to proprietary solutions such as the HERE Indoor Map Data Specification [Her] or ArcGIS Indoors Maps [Env], there is also an open standard for describing maps of buildings with Simple Indoor Tagging from OSM [Ope]. Basically, all these data specifications differ only in detail. Typically, the components of the building, e.g. floors, walls and doors, are described hierarchically using a JSON or XML markup language. An example of this is shown in Listing 4.1 for our own simpleLoc file format [sim]. Here the topmost element `map` describes the boundaries of the building and contains several other elements describing the location in geo-coordinates or other meta information like the address. A building is then divided into individual `floors`, each of which contains `obstacles`, `stairs` and other entities such as access points or regions of interest. An obstacle can be a `wall` with `windows` and `doors` or other objects that can be described by polygons. `Stairs` are made up of several `parts` in order to represent a more complex architecture. It is interesting to note that this is clearly a symbolic approach, describing the building in a topological way. So where the graph-based grid G describes the walkable area and is the basis of the transition process, the

floorplan is given as a human-readable representation that can be used, for example, to visualize the building in a graphical user interface.

Creating our own format was a necessary step, as the existing approaches are describing each floor as a single two-dimensional entity. The staircase is then only described by two contiguous areas, i.e. a polygon on each of the two floors surface it connects. Obviously, this lacks any information about the type and construction of the staircase. Without additional knowledge or further assumptions, such as a standard staircase shape, this means that a staircase can only be described by an additional vertical edge between two vertices of successive floors. This is often referred to as discrete stair changes [Ebn15]. The problem is clear: if the movement model based on a PDR represents the motion of the pedestrian, the rotation and acceleration motion on the stairs cannot be represented by the spatial model. Specifically, this means that although a particle moves correctly according to the movement model, the spatial model cannot represent these movements because the staircase is not sufficiently modeled in the graph G, but exists only as vertical edges. Therefore, the above requires for an accurate representation of stairs within G to allow realistic and continues floor changes. As can be seen in Listing 4.1 we describe a `stair` with several connected `parts`. Each `part` is defined by two points, the starting point and the endpoint, as well as a width.

4.2.2 Creating the Graph

The creation of the graph G is illustrated in Figure 4.2. First, the polygons of the `outline` are evenly gridded with vertices $\boldsymbol{v}_i$ depending on the granularity g_s (cf. Figure 4.2a). To create the stairs, the z-value of all $\boldsymbol{v}_i$ are at first ignored per `floor`, which reduces it to a 2D problem. For the respective `parts` of a `stair`, which can be trivially represented as polygons, the vertices can now be gridded in the (x, y)-plane of the ground exactly as before for the `outline`. The z-coordinate can now be recalculated by bilinear interpolation between the start and end points of the `part` for the respective vertices [SGM13]. Vertices with the same (x, y, z)-position are detected as duplicates and removed, they mostly occur at the connection points between the stairs and the floor (cf. Figure 4.2b). Note that the index i of $\boldsymbol{v}_i$ remains unique regardless of which element of the `map` it was created on. It is often useful to remove vertices that are directly above or below the stairs, as these often represent areas that are not accessible or where the pedestrian can only move by crawling (cf. Figure 4.2c). For this purpose a z-threshold model can be used, which examines vertices with the same (x, y)-coordinate and removes those that are above or below it [Ebn16]. In addition to stairs, escalators and lifts must also be considered. While the same procedure is used for escalators as for `stairs`, lifts must again be modeled using a simple vertical connection. However, by interpolating in the z-direction, the grid can be

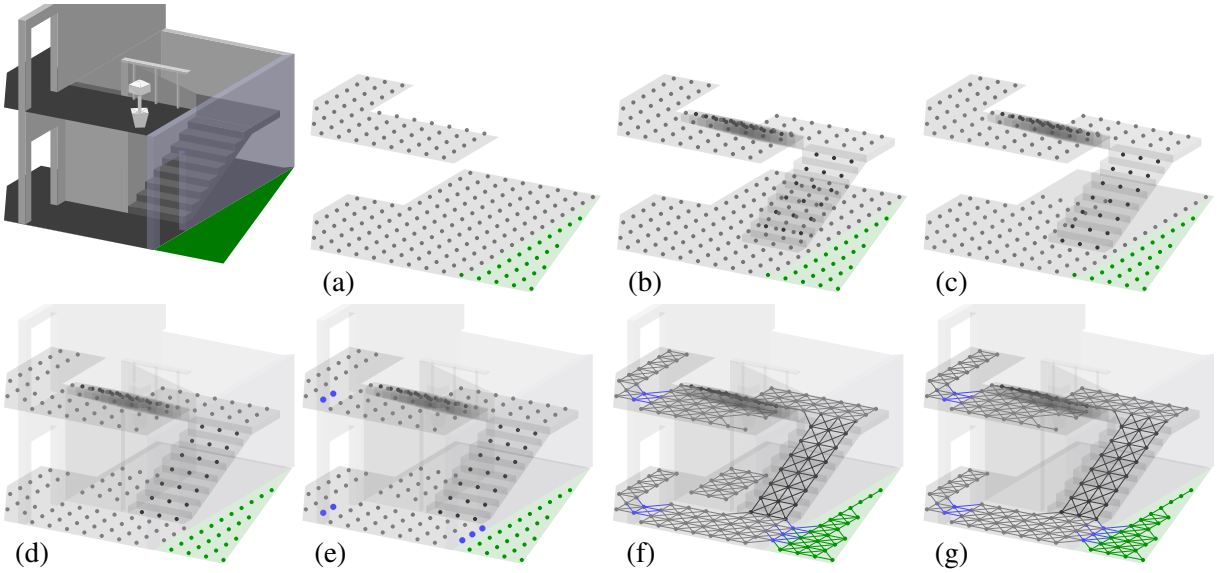


Figure 4.2: To create a graph from a given floorplan, the following steps can be taken: (a) Place vertices along each floor using a granularity $g_s = 50\,\mathrm{cm}$. b) Add vertices along any stairs, escalators, and elevators. c) Remove impassable vertices that are located above and below the stairs. d) Remove any vertices that are blocked by obstacles. e) Add semantic information such as the location of doors. f) Connect adjacent vertices in 3D using equation (3.19) if they are physically connected. g) Eliminate any small isolated areas. The figure was taken from [Ebn21].

maintained and two floors do not have to be directly connected by a single edge. While iterating over all elements of `map`, semantic information such as the location of doors or whether the vertex is inside or outside the building can be stored in $\boldsymbol{v}_i$ (cf. Figure 4.2e). In this step, vertices that are blocked by obstacles can be removed (cf. Figure 4.2d), what reduces the number of possible neighbors. Finally, all vertices can now be connected with edges $\boldsymbol{e}_{i,j}$ given as:

$$\forall \boldsymbol{e}_{i,j} \in E : \boldsymbol{v}_i, \boldsymbol{v}_j \in V \;\text{ and }\; (i \neq j) \tag{4.1}$$

where a neighbor $\boldsymbol{v}_j$ is identified by its position $\boldsymbol{\rho}_{\boldsymbol{v}_j}$, which is well defined by the granularity g_s using

$$\boldsymbol{\rho}_{\boldsymbol{v}_j} = \boldsymbol{\rho}_{\boldsymbol{v}_i} + \boldsymbol{d} \;\text{ with }\; \boldsymbol{d} \in \{-g_s, 0, g_s\}^3 \;\; . \tag{4.2}$$

A edge $\boldsymbol{e}_{i,j}$ that would intersect with an obstacle is removed, automatically describing the occupancy grid discussed above (cf. Figure 4.2f). Although (4.2) allows for a maximum of 26 connections per vertex, which includes nine above, nine below, and eight surrounding it, in practical scenarios, ten neighbors are sufficient. For example, in elevator setups, all eight neighbors on the (x, y)-plane are present, plus two extra neighbors for moving up and down.

The above described process may create isolated regions of adjacent vertices that are disconnected from most others, based on the building's architecture. To reduce the required memory, such regions can be safely removed by finding and deleting all vertices and edges that do not belong to the largest connected set (cf. Figure 4.2g). However, for multiple buildings described by a single floorplan that do not have a connection, this will fail. The greatest complexity lies in the stair construction described. Although the proposed method gives a reasonable approximation for most conventional buildings, it is still limited. For example, individual steps cannot be modeled, so if you look at Figure 4.2f, the staircase is more like a ramp. Especially with powerful activity recognition approaches, for example with an IMU attached to the foot, climbing individual stairs can be clearly distinguished from walking up a ramp, which inevitably requires a higher level of detail in the modeling of the building by the spatial model [AHZ21].

4.2.3 Transition Model

The transition model $p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_{t-1})$ is now derived using the above created graph G. As explained in the introduction to this section, the movement models presented here update the parameters, i.e. position $\boldsymbol{\rho}$, heading Θ, etc., of a particle by simulating the process of sampling from the transition density, rather than deriving a calculable representation as seen in the various sensor models. Providing a calculable representation for a 3D scenario is a challenging task, as it requires significant computation time to perform the intersection tests that are critical for preventing walking through obstacles. Furthermore, such a representation may not include a navigation framework or support for storing relevant meta-information, despite its importance [Köp14]. In contrast, we adapt a simulation method known as random walks on graphs to the problem of indoor localization. This fully covers the requirements above and remains efficiently computable [Ebn21; Ebn15].

Particles can be at any position within the Euclidean space of the system. However, to ensure that the particle does not pass through walls, the transition occurs directly at the edges of the graph. In addition, meta-information can be used to adapt the simulation to the current situation. To simulate the sampling of a new particle $\boldsymbol{X}_t$ via a random walk starting at the current $\boldsymbol{X}_{t-1}$, the following general steps are taken [Fet16] (cf. Figure 4.3):

1. Locate the vertex $\boldsymbol{v}_i$ to which the particle belongs, e.g. by using a hash-map that uses a combination of (x, y, z)-coordinates describing a grid cell as a hash value.

2. Determine a distance d_{walk} that mimics the walking behavior and speed of a pedestrian, which depends on the time between successive transitions.

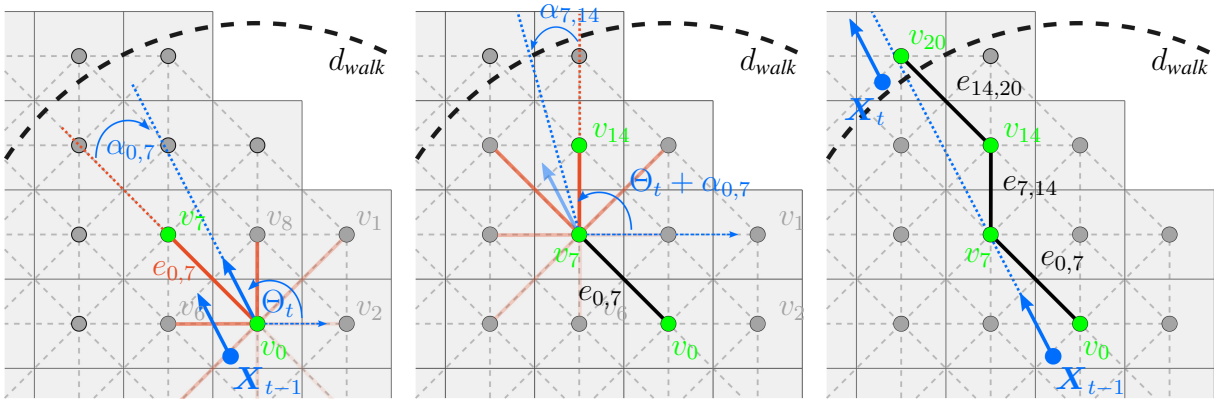


Figure 4.3: Random walks on a graph. (a) The vertex $\boldsymbol{v}_i$ describes the cell the particle $\boldsymbol{X}_{t-1}$ belongs to. The maximum distance of the random walk is given by d_{walk}. The angular difference between some edge $e_{i,j}$ and the particle's heading Θ_t is given by $\alpha_{i,j}$. The smaller this angular difference is the more likely is it for the random walk to choose the respective edge. The opacity of the edges (orange) provides an indicator of that likelihood given by $p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})$. (b) $\alpha_{i,j}$ changes per random walk based on its predecessor. (c) The final simulation for one transition update of the filter. The particle $\boldsymbol{X}_t$ is set randomly within the destination cell.

3. Walk randomly along one of the edges connected to vertex $\boldsymbol{v}_i$ to reach another vertex $\boldsymbol{v}_j$, taking into account the probability $p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})$ assigned to each edge $\boldsymbol{e}_{i,j}$ based on spatial information and recent observations o_{t-1}.

4. Subtract the length of the selected edge from the distance d_{walk}. Repeat steps 2 and 3 until d_{walk} is less than or equal to zero ($d_{\text{walk}} \leq 0$).

5. Determine the particle's position $\boldsymbol{\rho}$ within the destination vertex's cell and update the heading based on the most recent observation o_{t-1}.

The distance d_{walk} can result from sensor observations (e.g. acceleration) or some heuristic (e.g. fixed length based on filter update rate). If step detection is available as described in Section 3.3, this can be used to determine the distance based on the number and length of steps. However, it should be noted that not only the assumed step length is subject to error, but also the step counter itself. A pedestrian may be standing even though a step has been detected and vice versa. If one has given the probability for a step γ_t as defined in (3.19), then similar to the sensor model for step detection in (3.20), a mixed model results from which the distance is obtained:

$$d_{\text{walk}} \sim \begin{cases} \mathcal{N}(n_t^{>0} \cdot \mu_{\text{step}}, \sigma^2_{\text{step}}) & u < \gamma_t \\ \mathcal{N}(\mu_{\overline{\text{step}}}, \sigma^2_{\overline{\text{step}}}) & u \geq \gamma_t \end{cases}, \quad u \sim \mathcal{U}(0,1) \ . \tag{4.3}$$

Here, γ_t is used as a threshold to decide whether the random walk should simulate a standing or walking behavior. The step length is again denoted by the mean of the Gaussian modeling the uncertainty. To account for the number of steps n_t, without reducing the μ_{step} to zero we define

$$n_t^{>0} = \begin{cases} n_t & n_t > 0 \\ 1 & \text{otherwise} \end{cases} . \tag{4.4}$$

If, in addition to step detection, there is also continuous activity detection as in Section 3.5, (4.3) can be extended to

$$d_{\text{walk}} \sim \begin{cases} \mathcal{N}(n_t^{>0} \cdot \mu_{\text{step}}, \sigma^2_{\text{step}}) & \Omega = \text{walking} \\ \mathcal{N}(\mu_{\overline{\text{step}}}, \sigma^2_{\overline{\text{step}}}) & \Omega = \text{standing} \\ \mathcal{N}(n_t^{>0} \cdot \mu_{\text{asc}}, \sigma^2_{\text{asc}}) & \Omega = \text{ascending} \\ \mathcal{N}(n_t^{>0} \cdot \mu_{\text{des}}, \sigma^2_{\text{des}}) & \Omega = \text{descending} \end{cases} , \quad \Omega \sim \boldsymbol{\Omega}_t \ . \tag{4.5}$$

By representing $\boldsymbol{\Omega}_t$ as a discrete distribution, we are able to sample an activity Ω based on the respective probability of each detected activity stored in $\boldsymbol{\Omega}_t$. Sampling from a discrete distribution can be done using a method such as inverse transform sampling or binary sampling [Mas17]. The parameters of the Gaussian distributions must be determined empirically without the use of a step length estimator. In order not to exclude very slow or very fast walking speed, a correspondingly large variance should be chosen. However, it should be remembered that as the variance increases, so does the overall uncertainty, i.e. the posterior scatters more widely. Of course, a good estimation of step length would be beneficial, but as we discussed in Section 3.3, this is an extremely hard problem for wearable devices.

Once d_{walk} has been determined, the actual random walk process begins as illustrated in Figure 4.3. Each edge leading from $\boldsymbol{v}_i$ to one of the neighbors $\boldsymbol{v}_j$ is assigned a probability via $p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})$. The set of edges $E_{i,j}$ is now assumed to be a discrete distribution from which the edge "to walk" can be sampled. This means that edges with a higher probability are selected more often than those with a lower probability. The above process is then repeated until the total length of the adjacent edges is equal or greater then the distance d_{walk}. The way the movement model simulates human walking behavior is significantly influenced by the selection of an edge. Thus, depending on how $p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})$ is chosen and what assumptions about the movement are made, different strategies result.

A common assumption is that, at least for short filter updates (e.g. $2\,\text{s}$), the pedestrian walks straight most of the time and turns only occasionally. The particle therefore moves mainly in a straight line between time step $t-1$ and t. This means that while moving over the graph an

attempt is made to maintain the particle's updated heading

$$\Theta_t \sim f_{\text{mises}}(\Theta_{t-1} + \Delta\theta_{t-1}, \kappa_t) \tag{4.6}$$

as much as possible. Similar to (3.22) we use the von Mises distribution due to its circular properties and increment the particle's last heading Θ_{t-1} with the most recent observation $\boldsymbol{o}_{t-1}$ including the relative angular change $\Delta\theta_{t-1}$. However, unlike the edges, Θ and $\Delta\theta$ are not limited to a multiple of $45°$. So the actual direction of movement may be between two potential edges for the random walk. With respect to the x-axis (i.e. $0°$), the absolute angle of an edge in the (x, y)-plane is given by

$$\angle_{xy}(\boldsymbol{e}_{i,j}) = \operatorname{atan2}(y_{\boldsymbol{v}_j} - y_{\boldsymbol{v}_i}, x_{\boldsymbol{v}_j} - x_{\boldsymbol{v}_i}) \ . \tag{4.7}$$

The z-direction can be ignored, as we are only interest in the direction the pedestrian is moving and not were she is looking at.

Given the above, a strategy considering the heading is then denoted by assigning a probability to each neighboring edge according to

$$\begin{aligned} p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})_{\text{head}} = p(\boldsymbol{e}_{i,j} \mid \Theta_{t-1}, \Delta\theta_{t-1}) = f_{\text{mises}}(\alpha_{i,j} \mid 0, \kappa_{\text{turn}}) \\ \text{with } \ \alpha_{i,j} = \alpha_{k,i} + \angle_{\Delta}(\Theta_t, \angle_{xy}(\boldsymbol{e}_{i,j})), \ \ (i \neq j \neq k) \ . \end{aligned} \tag{4.8}$$

Here, $\alpha_{i,j}$ denotes the angular difference between an edge $\boldsymbol{e}_{i,j}$ and the particle's heading Θ_t (cf. (4.6)). To compensate for the $45°$ constraint, the predecessor $\alpha_{k,i}$ is retained and accumulated for each random walk. This is illustrated in Figure 4.3a and Figure 4.3b for two consecutive random walks. The first walk is either initialized with zero, or to further improve the results, especially for larger granularity g_s and slower filter updates, α can be stored throughout the entire filtering process by adding it as an additional parameter to the state vector $\boldsymbol{q}$ (cf. (2.75)). The von Mises again serves as a similarity measure in the equation above. The smaller the difference $\alpha_{i,j}$ between edge and heading, the higher the probability of selecting the respective edge $\boldsymbol{e}_{i,j}$. As already used in (3.22), the measure of concentration κ_{turn} denotes the allowed deviation from the walking straight assumption. It is a reciprocal measure of dispersion and as already shown in (3.25) the Gaussian' variance can be used analogous with $\sigma^2_{\text{turn}} = 1/\kappa_{\text{turn}}$. As is often the case, the choice of σ^2_{turn} depends on the scenario, with a higher value allowing more deviations from the assumption that a pedestrian walks mostly straight ahead. This can be particularly useful for slow filter updates in very winding buildings. A generally suitable value for the standard deviation may be $\sigma_{\text{turn}} = 0.1\,\text{rad}$ ($\approx 6°$).

Other strategies for selecting an edge can be based on the meta-information stored on the grid, the current activity of a pedestrian or other sensor observations. This allows for a variety of very simple to complex approaches. One approach is to store topological information about individual edges, e.g. whether the edge is on a staircase, lift, etc., and then associate it with an activity, e.g. stairs with ascending and descending [Fet16]. A probability $p(\boldsymbol{e}_{i,j} \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})_{\text{act}}$ for the edge is then obtained by comparing the above with the last known activity $\boldsymbol{\Omega}_{t-1}$. This can be done in a discrete way similar to (3.30) or continuously by just assigning the probabilities given in $\boldsymbol{\Omega}_{t-1}$.

A classical mathematical approach is the assignment of edge weights based on prior knowledge, resulting in a weighted graph. An example of such knowledge can be a navigation path, i.e. the routing from one's current position to a destination [Ebn16]. Classically, this can be calculated using shortest-path algorithms such as Dijkstra or A^*. As commonly known, they search for the shortest and with regard to edge weights, the most economical path. However, this often leads to paths that run directly along walls and thus do not correspond to the natural walking behavior of humans. The aim here is to achieve two things at once: On the one hand, the walking paths should be as realistic as possible and on the other hand, similar to the example above, this path should then be preferred in such a way that the movement model simulates random walks in its favor. The first can be achieved by assigning higher weights to edges that are near obstacles, except for narrow regions like doors or stairs. Therefore, selecting an edge near a wall is less likely then one that is in the middle of a hallway. To then secondly favor walking towards the destination and thus following the path, edges are preferred that reduce the distance to the destination. The random walk then starts by calculating the shortest-path between the starting vertex and the destination of the navigation based on the weighted graph. The probability function $p(\boldsymbol{e}_{i,j})_{\text{nav}}$ is then used to favor edges that are reducing the distance to the destination. Obviously, this would require to calculate a shortest path for every particle, whats computational complex. A simple solution is to run the Dijkstra algorithm inverted in an initial step. It starts from the target and calculates until each node has been evaluated, i.e. has been assigned a path and a distance.

By assuming a statistical independence all the probability densities of the above strategies can be combined into one movement model by a simple multiplication. This is analog to the sensor fusion approach presented in (3.1) of Section 3. The last thing missing, is to determine the particle's final position after the random walks exceeded d_{walk}. As illustrated in 4.3c, the particles is assigned the heading Θ_t according to (4.6) and the new position $\boldsymbol{\rho}_t$ is set randomly within the destination cell using a random function over the cells dimensions and coordinates.

4.2.4 Limitations and Discussion

One of the main limitations is that the movement is discontinuous, as the particles are only allowed to move along edges that are restricted to multiples of $45°$. For short filter updates this can be solved by calculating the direct destination $\boldsymbol{\rho}_t$ of the particle by adding the distance d_{walk} and the heading Θ_{t-1} to the particle start point $\boldsymbol{\rho}_{t-1}$. If the destination is inside a cell of the graph, it is valid and not inside an obstacle such as a wall [Ebn21]. However, this approach has several disadvantages. It requires an extra step of following adjacent edges in the same direction, similar to a shortest path algorithm like A^*, to find out if an obstacle is blocking. This may be faster than classical intersection tests, but it still adds computational weight. Another drawback is that meta-information stored on edges or vertices along the way cannot be used, as no random walks are performed. Finally, in this form it only allows straight connections and will therefore fail for large filter updates (e.g. $10\,\text{s}$) that involve walking around corners. A possible way to deal with this would be to store the current relative heading at each detected step in a vector and then create a trajectory to the destination instead of a straight line [Ebn22].

Another limiting factor, as mentioned several times above, is the granularity g_s of the underlying graph G. For example, round staircases can only be modeled correctly with a low g_s, resulting in a huge number of vertices. For a large building such as an airport or hospital, the number of vertices can easily exceed a few million. For each vertex, at least the index and neighbors must be stored, resulting in a $36\,\text{B}$ data structure [Ebn21]. With additional meta-information per vertex, the memory requirement can increase further, to several hundred megabytes per map. This may make it unsuitable for use on a smartphone. For comparison, the complete topological description of Germany in OpenStreetMap takes up just $3.8\,\text{GB}$ [Geo].

4.3 Navigation Mesh

Navigation meshes based on irregular tessellation are a well established and researched method for path-finding and positioning purposes. Especially in the field of computer games, they have established themselves as the de facto standard for AI opponents in first-person shooters such as Quake 3 or also for unit command in real-time strategy games such as Starcraft 2 [Bot13]. In the latter, it is particularly interesting that the mesh behaves dynamically, i.e. if, for example, a new building is erected in the base, the mesh is recalculated at this point.

In this work, we will only consider triangular meshes as illustrated in 4.1c. They are often preferred because of their compatibility with hardware designed to render graphics. Such hardware is optimized to perform operations on triangles efficiently, such as rasterization and shading. Other polygon shapes, such as quadrilaterals or pentagons, require additional pro-

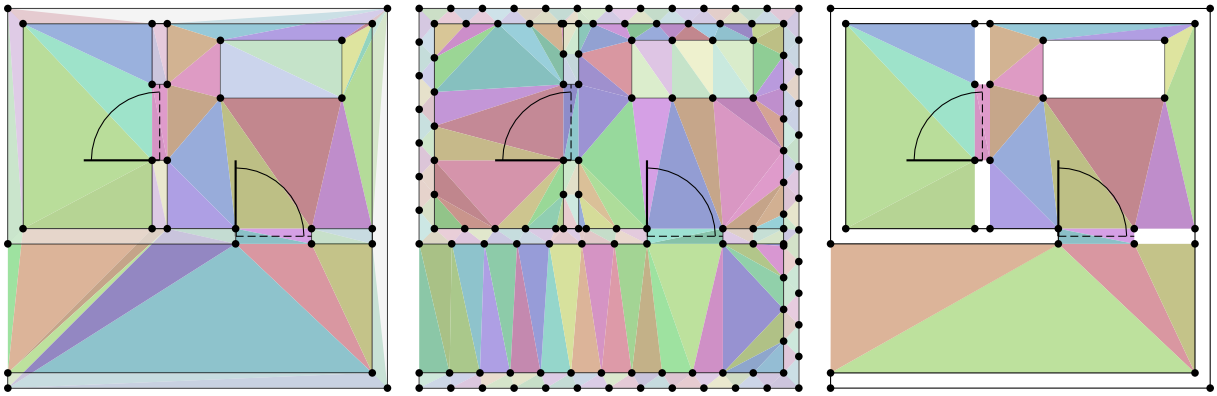
cessing. Triangular meshes have other advantages for navigation, such as better approximation of curved surfaces, fewer problems with intersecting geometry, and improved path-finding accuracy. As a result, many game engines and localization systems use triangular meshes for navigation and collision detection [Rab02; Rab19].

Another major advantage, particularly for localization, is the ability of navigation meshes to accurately represent very complex environments, e.g. the trees in Figure 4.1. Triangles of any size can be arranged in any configuration to capture the characteristics of arbitrarily shaped obstacles. The number of triangles used to represent a given area in a mesh is determined solely by the characteristics of that area, rather than its size. This allows greater flexibility in adjusting the level of detail in different areas of the mesh without affecting the overall size of the model. As a result, meshes have a dramatically reduced memory footprint compared to regular tessellated spatial models [Ebn21].

4.3.1 Creating a Navigation Mesh

There are many different approaches for creating a navigation mesh. Similar to the grid approach, the starting point is the same, a topological representation (cf. Listing 4.1) of the building as introduced in Section 4.2. Probably the best known algorithm for creating a triangular mesh is *Delaunay triangulation* [Del34]. Given is a set of to-be-triangulated vertices. The goal is now to connected them in such a way that no vertex is inside the circumcircle of any triangle formed by the connections, also refereed to as the Delaunay condition. The process has a distinctive property of maximizing the minimum angle of all the triangles in the mesh, which helps to avoid creating sliver triangles that can cause problems for interpolation or rasterization algorithms [Mus97]. To apply Delaunay triangulation for creating a navigation mesh, the vertices can be defined by the corner points of obstacles in the topological representation of the building. The algorithm will then connect these vertices to form triangles that cover the entire surface (cf. Figure 4.4a).

However, the Delaunay triangulation is unable to distinguish between walkable surfaces and obstacles, as triangles are simply created between the vertices. For example, if a straight wall segment is described by the vertices at its four corners, Delaunay triangulation does not distinguish between the order or whether a possible triangle is simultaneously inside and outside the wall. If the distances between the vertices are very large, such overlaps can occur. The effect can be observed especially in the lower left corner and in the upper right corner of Figure 4.4a. This often results in a mesh that is not well-formed and may contain narrow corridors or unreachable areas.



(a) (b) (c)

Figure 4.4: Examples of different approaches to creating an irregular spatial model based on a triangulation mesh structure. (a) was created using a standard Delaunay triangulation, where the vertices for the algorithm are defined by the corners of obstacles. In (b), interpolation points are placed at equal distances between the vertices to create a more even distribution of triangles and to avoid intersections with obstacles. Instead of using a heuristic as in (b), a more accurate solution using the constrained Delaunay triangulation is shown in (c), with triangles in obstacles already removed. Figures (a) and (b) are taken from [Ebn21] with modifications.

A straight forward solution to solve this, is interpolating between the vertices to generate more of them. The interpolation points are placed at equal distances between the vertices to create a more even distribution of triangles and to avoid intersections with obstacles. In 4.4b it can be seen, that the walkable surface can clearly be distinguished to the area covered by the walls and other obstacles. However, in a real-world scenario, generating such an interpolation is not straightforward and could still lead to the above problem if the interpolation fails on complex obstacles.

Constrained Delaunay triangulation is an improved solution that builds on the Delaunay triangulation approach [Che87]. It is a more flexible version that includes required segments (constrains) as edges in the triangulation. In contrast, Delaunay triangulation only considers the position of vertices without considering their required connections. By incorporating these segments, constrained Delaunay Triangulation ensures that the final mesh meets specific requirements while still retaining the desirable properties of Delaunay triangulation, such as avoiding sliver triangles. An example is illustrated in Figure 4.4c.

There are many different algorithms available for generating both constrained and standard Delaunay triangulations. These algorithms can vary in terms of their running time, accuracy, and ease of implementation. Some of the most common algorithms include divide-and-conquer [Che87; GS85], sweepline [For86], and incremental approaches [Ang97; Dev98]. While the divide-and-conquer and sweepline algorithms are considered asymptotically optimal with a run-

ning time of $\mathcal{O}(n \log n)$, they may require complicated data structures and may not be the most practical choice for all applications [KBT04]. Incremental algorithms, on the other hand, are often simpler to implement and can be competitive with the optimal approaches if optimized point location data structures are used [MSZ99]. For simple polygons, there are even approaches that can be calculated in linear time [CW98].

One of the best-known and most frequently cited implementations for generating a constrained Delaunay triangulation incremental is Triangle [She96]. The algorithm proceeds in several steps. First, it creates a convex hull around the set of input vertices, which provides a boundary for the triangulation. Next, it adds any user-specified constraints as additional edges to the triangulation. These constraints can represent obstacles, boundaries, or other features that should be incorporated into the triangulation. Once they have been added, the algorithm computes the Delaunay triangulation of the resulting point set, using an incremental approach. This involves adding points one at a time and updating the triangulation to maintain the Delaunay property. Thereby, the algorithm checks each triangle in the triangulation to ensure that it does not violate any of the constraints. If a triangle intersects a constraint, it is removed from the triangulation and replaced by two new triangles along the constraint edge. This process continues until all triangles in the triangulation satisfy the specified constraints.

Corresponding extensions for the 3D space are presented in [GSG06] or [DW04], for example. For use as a navigation mesh, however, a similar modification as for the grid graph is sufficient to be able to model stairs and escalators. Triangulation is performed separately for each `stair` element of the topological floorplan (cf. Listing 4.1), with the vertices projected back into the (x, y)-plane. The z-values of the triangle points are then determined by interpolation at the corner points. Stair and floor triangles are finally reconnected in the underlying data structure using the topological information and the (x, y, z)-coordinates of the stairs start and end.

When constraints are added to a triangulation, not all resulting triangles may conform to the Delaunay criterion. This can lead to undesirable sliver triangles near constrained edges and a loss of the associated *Voronoi Diagram*. The Delaunay triangulation corresponds to the dual graph of the Voronoi diagram, meaning that they can be generated from each other. The Voronoi diagram divides space into a set of regions described by polygons, where each region contains all positions of the surface that are closer to a particular vertex in the input set than to any other. Based on this, a *generalized Voronoi graph* can be created that represent the connectivity between the regions in the space [CB95]. It is defined by taking the corners of the polygons as vertices and the boundaries between the polygons as edges. The Voronoi graph is highly used in indoor scenarios for creating a symbolic spatial representation of the building and for path-finding applications like navigation [LLF19; WKS15; YW15].

To overcome the above problem, the authors of [Rog99] developed a technique to restore Delaunay optimality by adding synthetic points along constraint edges. The edges are then subdivided into smaller edges, resulting in triangles that conform to the Delaunay criterion. This method is similar to the interpolation approach shown in Figure 4.4b. However, it also tends to increase the number of triangles used in the triangulation, often unnecessarily, especially for complex polygons.

While Delaunay triangulation approaches are often referred to as exact solutions for creating a navigation mesh, there are also many approaches that attempt to approximate the walkable space. A very popular example is the open-source library Recast Navigation [MC13]. It generates its mesh using a technique called voxelization, where the space is divided into small, uniformly-sized 3D volumes called voxels. The library then determines which voxels are occupied by obstacles and which are not, creating a 3D representation of the world (e.g. a building or game level). Once the voxel representation is created, Recast Navigation uses a method called contouring to generate a 2D mesh that represents the walkable surfaces in the game world. This mesh is created by tracing the edges of the occupied voxels and creating a polygonal outline of the resulting shape. The resulting mesh is then triangulated to create a final navigation mesh.

Despite the lose of geometric details, approximation approaches like Recast Navigation often offer for useful parametrization options, e.g. controlling the density and placement of triangles. In certain cases, it may be advantageous to represent a large open space using several small triangles instead of just a few larger ones. This approach can be beneficial when a triangle needs to carry metadata in a more efficient manner. Another practical benefit is that most of these approaches are able to generate the mesh directly from a topological representation as used in this work, as well as from common 3D models. Especially in the 3D case this is beneficial, as for an Delaunay approach the set of vertices should only describe the walkable area. Other include vertices, e.g. describing the appearance of an obstacle in z-direction, will be added to the mesh and will produce unfavorable results. Depending on the topological representation, a pre-processing step is also required, what further increases the complexity.

4.3.2 Transition Model

The movement model for answering the transition density $p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i, \boldsymbol{o}_{t-1})$ considers a given navigation mesh, representing only the walkable regions of the building. A grid graph (cf. Section 4.2) allows well defined walking along adjacent edges with regular tessellated vertices, i.e. the distances covered and the angular varieties for each random walk are known exactly. In contrast, using random walks in a similar way along the triangle edges of a navigation mesh would fail in most scenarios because the triangles are irregularly shaped, resulting in large edge

lengths for large free space regions and small edges for narrow corridors or environments with many complex obstacles. Instead of treating the mesh solely as a graph where movement is restricted to edges and vertices, true continuous movement can be achieved by allowing particles to walk to any destination, on the assumption that it lies within a triangle that is actually walkable from the starting position. Of course, this requires an additional check if a potential destination is contained within a triangle, however, in contrast to arbitrary polygons, these checks can be calculated at least in linear time $\mathcal{O}(n)$.

Such a check can be achieved using barycentric coordinates, which describe the position of a point based on the edges of a triangle, making it easy to check whether it is inside the triangle or not [Fet18]. Another approach based on a Delaunay triangulation is described by [MSZ99]. The procedure finds the desired triangle by "walking through" the triangulation in the direction of the searched point. This is based on the idea that you can quickly find nearby triangles if the triangulation is kept in a doubly connected edge list or similar structure [DMB98]. The lack of additional data structures makes the implementation very simple.

Thanks to good understanding and widespread use in other research areas, there are efficient methods for many operations on the navigation mesh. In the upcoming movement model process, we also require the operation of extracting sub-meshes from the direct neighborhood of an initial triangle based on a given radius. Considering a tree representation of the mesh, where each triangle is a node storing the (x, y, z)-position of each vertex defining that triangle and its direct neighbors along the edges, maximum three. The root node is now considered to be the initial triangle, and each of its neighbors is a child node, their neighbors are also children, and so on. Now a search algorithm for tree structures can be used to find all children that do not exceed a certain radius between the position of a particle within the initial triangle and at least one of the vertices of the children. For small radii, containing only a few triangles, a recursive approach like *breadth-first search* provides a suitable and fast solution [BW84]. It is interesting to note, that this approach resamples a shortest-path search per recursive call and can also be utilized for path-finding purposes [Tri21].

Given a navigation mesh and the above tools, we are now able to simulate a particles movement, i.e. updating to position $\boldsymbol{\rho}_t = (x_t, y_t, z_t)^T$ and heading Θ_t, according to

$$\begin{aligned} x_t = &\ \overbrace{x_{t-1}}^{\text{old pos.}} + \overbrace{d_{\text{walk}}}^{\text{distance}} \cdot \overbrace{\cos(\Theta_t)}^{\text{direction}} \\ y_t = &\ y_{t-1} + d_{\text{walk}} \cdot \sin(\Theta_t) \\ z_t = &\ \text{interpolated} \\ \Theta_t \sim &\ f_{\text{mises}}(\Theta_{t-1} + \Delta\theta_{t-1}, \kappa_t)_{(4.6)} \ , \end{aligned} \tag{4.9}$$

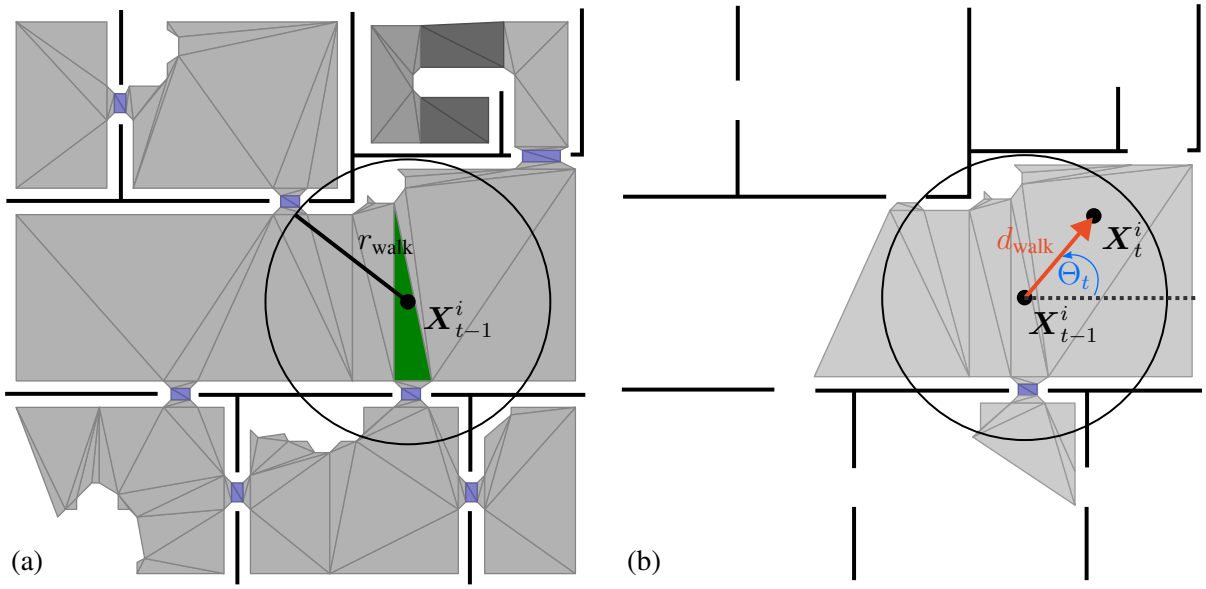


Figure 4.5: Movement model on a navigation mesh. In (a) the corresponding triangle (green) to the particle $\boldsymbol{X}^i_{t-1}$ is found. Base on some radius r_{walk} a sub-mesh is extracted that models only realistic walks, i.e. no walking through walls or big jumps. After that, the movement is carried out using the current estimated heading Θ_t (cf. (4.6)) and distance d_{walk} (cf. (4.5) or (4.3)) according to (4.9) in (b).

where the distance d_{walk} is again denoted using either (4.5) or (4.3). This satisfies $p(\boldsymbol{X}^i t \mid \boldsymbol{X}^i_{t-1}, \boldsymbol{o}_{t-1})$ in a truly continues way for the (x, y)-plane. Because the navigation mesh only contains the walkable surface, 3D positions are not arbitrary, but are defined by triangles that model stairs or elevators, for example. Therefore, the z-coordinate can be interpolated based on a corresponding triangle, if it exists. Similar to the graph-based movement model (cf. Section 4.2), the above assumes that a pedestrian walks mostly straight ahead between $t-1$ and t and turns only occasionally. Again, this will fail for large filter updates (e.g. $10\,\text{s}$) that involve walking around corners. A possible way to deal with this would be to store the current relative heading at each detected step in a vector and then create a trajectory to the destination instead of a straight line [Ebn22]

To only model movement that is actual feasible, i.e. no walking through walls and no big jumps, the new position $\boldsymbol{\rho}_t$ has to have a corresponding triangle within a sub-mesh that is defined by a radius r_{walk} surrounding the starting position $\boldsymbol{\rho}_{t-1}$. The size of the sub-meshes, and thus r_{walk}, depends on the update rate of the particle filter. So it has to answer the question which triangles are realistically reachable between time t and $t-1$. One obvious approach is to simply use $r_{\text{walk}} = d_{\text{walk}} \cdot k$, where k is some positive constant factor > 1 to compensate for the variance.

If $\boldsymbol{\rho}_t$ is contained within a triangle of the sub-mesh, its z_t can be obtained using the barycentric coordinates of $(x_t, y_t)^T$ within a 2D projection of the triangle to which the position belongs.

Applying the barycentric coordinates to the original 3D triangle and converting them back to the euclidean space then results in an interpolation of z_t. A successful movement simulation for a single particle is illustrated in Figure 4.5. It can be summarised in four steps. First, the triangle corresponding to the initial particle $\boldsymbol{X}_{t-1}^i$ is found. Then the radius r_{walk} is determined to get a sub-mesh of the navigation mesh. Next, the new position and heading of $\boldsymbol{X}_t^i$ is calculated as shown in (4.9). Finally, it is checked whether $\boldsymbol{X}_t^i$ has a corresponding triangle within the sub-mesh. If so, the process is completed by interpolating the z-coordinate. If this is not the case, an additional step to handle these failures must be performed to deal with it.

There are several possible strategies to deal with such an illegal movement, e.g. caused by running into a wall or covering too long a distance. The simplest option is to eliminate the particles from the particle set. However, if a building has many walls, this can quickly lead to a massive reduction in particles, possibly to the point where no particles remain. When resampling, this can be remedied relatively easily by not drawing as many particles as there are in the total set, but always drawing a constant base size. This automatically produces duplicates of particles that had an successful movement. Although, or perhaps because, this strategy is very simple, it accelerates a well-known problem, weight degeneracy (cf. Section 2.3.2). Resampling itself is known to be a possible countermeasure, but not an absolute solution. In a building with many walls, i.e. a large number of particles that are likely to hit these walls, it may happen that only a few particles remain from which resampling can take samples. If the variance for heading and distance is then set very small, the dispersion of the particles can quickly collapse.

A variant of this strategy is called *sink or swim* [Fet18]. If there is no corresponding triangle after the movement, a new position for the particle is drawn cumulatively (i.e. inverse transform sampling) from the set of successful simulations. However, all other parameters, such as the heading, are retained. This makes it possible for the particle set to recover some variance towards the next transition step by moving from the same position but in a presumably different direction. In a scenario with many narrow rooms and therefore many obstacles, this results in a potential loss of dispersion and therefore posterior representation in the system.

The last strategy to handle a failed movement presented in this work is called *debase* [Fet23a]. For each particle that has no corresponding triangle, we begin by randomly selecting a reachable position within the radius r_{walk}. To add more variation to the heading we further increase the noise in (4.6) by some constant factor, e.g. doubling it. The resulting particle, $\boldsymbol{X}_t^i$, is then flagged so that its weight will be halved during the evaluation process, making it uncertain within the posterior and more likely to be removed during resampling. Over time, this process eliminates particles that collide with obstacles or walls, thereby enhancing the accuracy of the positioning system. This method is especially effective in buildings with numerous small rooms where the motion model can quickly converge from the uniform prior $\mu(\boldsymbol{X}_1^i)$ (cf. Algorithm 5)

and generate a reasonable posterior, even without the use of absolute positioning models in the evaluation process.

In the grid, incorporating further knowledge to improve e.g. floor changes or other activities could easily be done by manipulating the weight of the edges, cf. (4.8) and ongoing discussion. Unfortunately, there is no such elegant solution for the navigation mesh due to the irregular size of the triangles. Each triangle can be given meta-information, such as whether it is a staircase or an outdoor area, but the particles do not move along edges but continuously from triangle to triangle. In large areas of a building, movement can even be simulated within a single triangle. As discussed at the beginning of this section, it makes little sense to use the edges of the triangles for walking. In order to still be able to manipulate the movement of the particles based on observations and meta-information, we use a little workaround. As anticipated in Section 3, we introduce a sensor model of the form $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}}$. This allows us to manipulate the weights of the particles $\{W_{1:t}^i, \boldsymbol{X}_{1:t}^i\}_{i=1}^N$ based on the last and current state of the particle. For performance reasons, this sensor model is implemented directly in the movement model of the transition step. Otherwise all operations on the navigation mesh would have to be performed again in the evaluation. As transition and evaluation are performed directly one after the other in the CONDENSATION particle filter (cf. Algorithm 5), there are no disadvantages.

The weight based on the activity can then be assigned in the same way as for the grid. So, based on a comparison of the topological information stored in the triangle with the currently detected activity $\boldsymbol{\Omega}_{t-1}$, a weight can be obtained similar to (3.30) or, if there is continuous activity detection, by assigning the corresponding probability. Again, there are many other ways to do weighting, i.e. finding a solution for $p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i, \boldsymbol{X}_{t-1}^i)_{\text{pdr}}$. For example, if there is prior knowledge such as a navigation path or areas of barrier-free access. The former can be solved similarly to the grid, where particles whose distance to the target is reduced are weighted higher by a fixed factor [Ebn21]. In the resampling step, these particles are then drawn more frequently than others, leading in the long run to a convergence of the posterior towards the navigation target.

Compared to the grid graph, the navigation mesh requires further processing to be suitable for path-finding. The problem is that direct paths along the edges of the triangles would lead to strange zigzag patterns, especially in large open spaces where the triangles are sparse. A straightforward solution is by artificially increasing the number of triangles, e.g. by halving all triangles of a certain size in a pre-processing step. A graph is then derived from the midpoints of the triangle edges, which can then again be used for shortest-path calculation. This improves the zigzag patterns, but cannot completely eliminate them. Particularly narrow, directly adjacent triangles will add to this effect.

A more promising approach is proposed by [Kal10]. They introduce a novel type of triangulation, called a *local clearance triangulation*, which uses an arbitrary navigation mesh that satisfies the Delaunay constraint and modifies it according to a local clearance property. Similar to approaches following the continues Dijkstra paradigm like [HS99] or [Mit93] they improve the shortest path by skipping improper vertices and creating new connections directly through triangles. The clearance property allows efficient and accurate determination of whether a disc of any size can pass through narrow mesh passages. Therefore, the calculation of a navigation path takes into account the size of an agent, i.e. in the case of an ILS, the minimum distance between two obstacles that is acceptable for a pedestrian to walk through. This automatically results in more realistic walking paths, without sticking to walls and obstacles. Lastly, as the authors note: "In larger environments, this extended search can be significantly slower [...] because each expansion requires many more geometric computations [...]" [Kal10].

Another popular approach for path-finding is to utilized the Voronoi graph. As described above, it is the dual graph of a navigation mesh that satisfies the Delaunay condition. It divides space into a set of regions described by polygons, where each region contains all positions of the surface that are closer to a particular vertex in the input set than to any other. Based on this, a *generalized Voronoi graph* can be created that represent the connectivity between the regions in the space [CB95]. It can be thought of as a rail network through the building, in which a corridor, for example, is represented only by a bidirectional rail, i.e. an undirected edge. Navigation can only take place on the edges that have branches, i.e. vertices, into other areas of the building. Sometimes it is also referred to as a connectivity and accessibility graph, since only the possible connections and their topological relationships are described symbolically. There are several solutions that use this type of graph for navigation purposes [LLF19; WKS15; YW15]. This may be sufficient for many applications, but if navigation to any location in the building is desired, such a representation severely limits the accessible area.

4.3.3 Limitations and Discussion

In fact, many of the advantages of the navigation mesh are disadvantages of the grid graph and vice versa. While the grid graph requires a lot of memory for large buildings, the mesh requires only a fraction of the memory [Fet18]. On the other hand, the grid offers optimal conditions for storing meta-information in an intuitive way [Ebn16]. For example, with a sufficiently small granularity, almost any information such as room names, points-of-interest or areas-of-interest can be stored directly at a vertex or edge. Due to the irregular size of the triangles, this is only possible to a limited extent with a mesh. Further data structures have to be introduced in order to assign the information as directly as possible to a triangle and thus use it efficiently for the

ILS, e.g. a point-of-interest within a triangle can be described by its barycentric coordinates and the triangles unique identifier in a hash map.

Including elevators to the navigation mesh is an open research question [Pel16]. There are several possibilities: to represent the elevator shaft by triangles, to simply draw vertical edges upwards, which do not fulfill the triangle condition, or to construct a zigzag structure that behaves like a staircase. Pure vertical motion, as required for the first two suggestions, cannot be simulated with the movement model proposed above, since only distances in the (x, y)-plane are assumed.

In addition, stairs are approximated as ramps, as can be seen in Figure 4.1 or 4.5. This is due to the topological description used in this work (cf. Listing 4.1). Theoretically, stairs can be modeled exactly by triangles, i.e. providing detailed measures for each element like steps or plateaus. However, this can not be simulated with the proposed movement model, as it would require real continues z-coordinates. It should be noted, that the same applies to model used by the grid graph. Especially for stairs, a distinction would have to be made between horizontal and vertical motion on associate triangles. In the case of an ILS for smartphones, the sensor accuracy is not sufficient for this. Neither the barometer nor the IMU are currently able to provide the accuracy required for a handheld device (cf. Section 3.3 and Section 3.4). A foot-mounted solution with high accuracy IMUs would be more appropriate.

Lastly, a final limiting factor arises from the strategies presented. Similar to the grid graph, particles are restricted from passing through obstacles. This prevents free movement within the dynamic system, which can lead to particles getting stuck in the corner of a room, for example due to an access point malfunction or a wrongly detected turn. As the localizing pedestrian moves further away from this corner, absolute observations such as Wi-Fi measurements become increasingly inappropriate, resulting in a low weighting of all particles, the PDR continues to cause movement leading to a final misalignment. In other words, the particle set is unable to provide a good approximation of the underlying posterior density and thus provides a poor position estimate. This problem was introduced earlier in this thesis as sample impoverishment, which refers to the situation where there are not enough particles in areas of the system that are likely to affect the sensor measurements and hence the evaluation density.

Chapter 5

Multiple Model Fusion

So far, we have assumed a particle filter containing all sensor and movement models for state estimation. The former were combined via the assumption of statistical independence, as discussed in Chapter 3. In practice, this can lead to problems with sensors of very different accuracies, requiring a balanced configuration between the different sensor models. To explain this, it is worth recalling the example at the end of Section 1.2.2. Here, we consider an evaluation procedure that combines a Wi-Fi multilateration and a magnetic matching sensor model. A pedestrian is walking into an isolated corridor where the Wi-Fi measurements are highly attenuated, while the magnetic fingerprints continue to provide reliable results. Combining the two sources of information results in a worse, or rather more uncertain, representation of the state space, consequently leading to a less accurate estimation of the position compared to relying solely on magnetic matching. A similar issue arises when dealing with different sensor types that can only be sporadically and unpredictably integrated into the system, such as Wi-Fi and surveillance cameras. The intermittent nature of their input further exacerbates the challenges in achieving accurate position estimation. Although it is possible to mitigate this phenomenon by adjusting the sensor model parameters, for example, by assigning weights based on the building's specific areas, determining such parameters often lacks generalizability and depends heavily on the specific scenario [Fet23a].

Another limitation of classical filter methods is that only one movement model can be specified. Of course, the transition can theoretically be any probability density, which in principle allows a combination under certain assumptions. However, this combination of movement models is not trivial, as we would have to simulate a movement from two different transition densities. In practice, this may be quickly implemented, as both models simply draw new particles into a common density in parallel, but for very disjoint models it inevitably leads to a

drifting apart of the particle set. Ultimately, this leads to unwanted multimodal posterior and, if the number of particles is too small, quickly to sample impoverishment.

One possible solution is to run multiple filters in parallel that perform the same state estimation instead of a single filter. This allows for different movement models and sensor combinations per model. This then results in completely different filter implementations to represent the respective modes or behavior of the system to be estimated. By combining the respective models in a joint posterior, it is expected that the robustness and accuracy of the estimation will be improved. In the literature, such approaches are known as *interacting multiple model* (IMM) filters. They were first introduced in [PB84] as an extension of the Kalman filter for abruptly changing systems in radar-based aircraft tracking. Since then, IMMs have become very important, especially in the field of maneuvering target tracking, and have become the de facto standard for nonlinear motion models in the Kalman filter environment. Different models such as constant velocity, constant turn and constant acceleration motion are combined to model the non-linearity in an EKF or UKF.

An extension to particle filters and thus to non-Gaussian distributed systems has been proposed in [BD03] as *interacting multiple model particle filter* (IMMPF). In the following, we first discuss the general IMM approach, then derive IMMPF, and finally adapt it to the localization and navigation use case. We extend IMMPF with a dynamic mixing process and propose a weighted state estimation.

5.1 Interacting Multiple Model Particle Filter

The basic idea behind the IMM filter is to maintain a set of filters, called modes $m_t \in M \subset \mathbb{N}$, each associated with a specific transition and evaluation model. These modes operate independently and estimate the state of the system based on their respective models. The IMM filter then combines the estimates using a mixing mechanism to obtain the final estimate of the posterior distribution and thus the state of the system. The mixing is typically based on the likelihood of each mode given the available observations $\boldsymbol{o}_t$. Therefore, the main advantage of the IMM filter is its ability to handle situations where the system behaviour is uncertain and/or the environment such as sensors or dynamics changes. By using several different models in independently running filters, the IMM filter can adapt to different behaviours and provide more accurate estimates compared to a single filter approach.

Consider a jump Markov non-linear system that is represented by a set of modes M, describing the state space. That means, transitions between single modes m_t can occur through random jumps at any time according to a Markov chain. The key characteristic of a Markov jump process is that the probability of making a transition from one state to another depends only on

the current state and the elapsed time, but not on the history of states before that. Therefore, it follows the same assumptions given by the HMM as in Bayes filtering. Lets further consider that each mode m_t refers to a particle filter as described in Section 2.5. The joint posterior distribution of the IMMPF can then be denoted using the identity

$$p(\boldsymbol{q}_t, m_t \mid \boldsymbol{o}_{1:t}) = \underbrace{P(m_t \mid \boldsymbol{o}_{1:t})}_{\text{mode}} \underbrace{p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t})}_{\text{filtering}} \ . \tag{5.1}$$

The main advantage of this form, compared to directly solving $p(\boldsymbol{q}_t, m_t \mid \boldsymbol{o}_{1:t})$, e.g. by simply extending the standard CONDENSATION particle filter with the discrete mode variable m_t, is the decoupling of the probability representation of the system and the exchange between different modes. As we will see, this introduces a mixing stage that allows direct control of the number of particles per mode.

5.1.1 Mode Conditioned Filtering

Given our posterior distribution for filtering in (2.72), the mode conditioned filtering stage of (5.1) can be written as

$$\begin{aligned} &p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t}) \propto \\ &\underbrace{p(\boldsymbol{o}_t \mid m_t, \boldsymbol{q}_t, \boldsymbol{q}_{t-1})}_{\text{evaluation}} \int \underbrace{p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, m_t, \boldsymbol{o}_{t-1})}_{\text{transition}} \underbrace{p(\boldsymbol{q}_{t-1} \mid m_{t-1}, \boldsymbol{o}_{1:t-1}) d\boldsymbol{q}_{t-1}}_{\text{recursion}} \ , \end{aligned} \tag{5.2}$$

where we again extend the transition by the previous observation $\boldsymbol{o}_{t-1}$ and the evaluation by the previous state $\boldsymbol{q}_{t-1}$ according to a second order Markov assumption. The above enables for independently and in parallel running particle filters identified by m_t. This not only allows for arbitrary combinations of different sensor and movement models, but also allows the number of particles in each mode to be chosen independently. For example, very restrictive movement models such as constant velocity on routing graphs [WKS15] do not benefit from a large number of particles, unlike the movement models presented in this thesis. However, as we will see later, if the particle numbers are spread too widely, it is easy for filters with higher particle numbers to be over-represented in the mixing step.

The other term of (5.1), $P(m_t \mid \boldsymbol{o}_{1:t})$, provides a discrete probability distribution describing how likely it is that a considered mode represents the systems current state. According to [BD03] it can be derived recursively from the posterior mode probabilities

$$p(m_t \mid \boldsymbol{o}_{1:t}) = \frac{p(\boldsymbol{o}_t \mid m_t, \boldsymbol{o}_{1:t-1}) P(m_t \mid \boldsymbol{o}_{1:t-1})}{p(\boldsymbol{o}_t \mid \boldsymbol{o}_{1:t-1})} \ , \tag{5.3}$$

using an approximation of the form

$$P(m_t \mid \boldsymbol{o}_{1:t}) = \frac{\omega_t^{m_t} P(m_t \mid \boldsymbol{o}_{1:t-1})}{\sum_{m=1}^{M} \omega_t^{m_t} P(m_t \mid \boldsymbol{o}_{1:t-1})} \ . \tag{5.4}$$

Here the denominator provides the normalization to 1, which is important because the mode probabilities also give us an indication at each time step of how much influence each filter has on the overall posterior. It is quite obvious that the approximation in (5.4) follows the same principles as the derivation of SIS in Section 2.3.1. But instead of per sample, it is per mode, so $\omega_t^{m_t}$ is the unnormalized weight given by the state evaluation of the respective mode m_t. Due to its form, it requires an a priori $P(m_1 \mid \boldsymbol{o}_1)$, which is often chosen to be a simple uniform distribution over M.

5.1.2 Mixing of Modes

To provide a solution for the probability distribution $P(m_t \mid \boldsymbol{o}_{1:t-1})$ in (5.3), the recursion for m_t in (5.1) is now derived by the so called mixing stage [DB05]. We compute

$$p(\boldsymbol{q}_t \mid m_{t+1}, \boldsymbol{o}_{1:t}) = \sum_{m_t} P(m_t \mid m_{t+1}, \boldsymbol{o}_{1:t}) p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t}) \tag{5.5}$$

with

$$P(m_t \mid m_{t+1}, \boldsymbol{o}_{1:t}) = \frac{P(m_{t+1} \mid m_t) P(m_t \mid \boldsymbol{o}_{1:t})}{P(m_t \mid \boldsymbol{o}_{1:t-1})} \tag{5.6}$$

and

$$P(m_t \mid \boldsymbol{o}_{1:t-1}) = \sum_{m_t} P(m_{t+1} \mid m_t) P(m_t \mid \boldsymbol{o}_{1:t}) \ , \tag{5.7}$$

where (5.5) is a weighted sum of mode conditioned filtering density and the weights are provided through (5.6). The transition probability $P(m_{t+1} = k \mid m_t = l)$ in (5.7) is given by the Markov transition matrix $[\Pi_t]_{kl}$. The matrix Π_t is a real square matrix, with each row summing to 1. It gives the probability of moving from m_t to m_{t+1} in one time step, thus denoting a Markov switching process on the underlying Markov chain.

It is interesting to note that for a time-continuous Markov chain this is called a Markov jump *process* and for time-discrete it is called a Markov jump *chain*. However, some time-discrete Markov chains are not Markov jump chains [Gag17]. They refer to a Markov process where the transitions can occur at arbitrary or random times, which is typically not true for regular intervals such as every second, every hour or every day. As discussed in Section 2.2.1, we have defined the sensor fusion application as a time-discrete HMM with a measurable state space. This means that we are able to model time-continuous state spaces as long as we can

measure/sample them at discrete time intervals. Consequently, this allows for irregular but discrete interval sizes between t and $t+1$, as well as arbitrary timed observations in between, and thus ultimately classifies our proposed ILS as a Markov jump chain.

The mixing stage itself can now be implemented using a direct sampling approach [DB05]. Sampling from a sum of distributions such as (5.5) is straightforward, first a new mode m_t is drawn from the discrete distribution (5.6), and second, a new sample $\boldsymbol{X}_t^{i,m_t}$ is drawn according to $\boldsymbol{X}_t^{i,m_t} \sim p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t})$, with respect to that m_t. Since $p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t})$ is represented by a particle set, the above is simply equivalent to sampling, e.g. based on a strategy presented in Section 2.3.2, from the discrete distribution $P(m_t \mid m_{t+1}, \boldsymbol{o}_{1:t})$ defined by the particle weights. In short, this makes it possible to draw new particles from all available filters and not just one's own, hence the name mixing step.

The complete IMMPF with direct sampling is given in Algorithm 7. It starts with M initial particle sets $\{W_{1:t}^i, \boldsymbol{X}_{1:t}^i\}_{i=1}^N$, one for each mode, drawn from the a priori filter distribution $p(\boldsymbol{q}_1 \mid m_1, \boldsymbol{o}_1)$ with corresponding initial mode probabilities $P(m_1 \mid \boldsymbol{o}_1)$. The process starts a new iteration at time t with the mixing step (lines 1 to 9). As explained above, at first a mode m_{t-1}^i is drawn according to (5.6) in line 4 and then a corresponding sample $\boldsymbol{X}_{t-1}^{i,m_t}$ from the previous mode conditioned filtering posterior in line 5. Since the newly drawn particles of the currently considered mode can also originate from other modes, the particles must be equally weighted (see line 6) due to the independence assumption between the filters. After the mixing, each $m_t \in M$ runs in parallel starting form line 10. Although the IMMPF can use any suitable particle filter scheme, the algorithm includes our extended CONDENSATION filter as this is the main inference tool in this thesis. In addition, the IMMPF was first introduced by [BD03] in a similar form, which allows for a good comparison. It is essential that the independent filters (lines 10 to 19) complete their processing within the previous iteration $t-1$ for the new iteration to start.

As introduced in [Fet23a] we further extend the IMMPF algorithm in line 1 by adding a conditional statement based on the average effective sample size over all involved modes $\bar{N}_{\text{eff}}$, if it exceeds a predefined threshold $\bar{\hbar}$. The main reason for this is to ensure some variance in particle weights between filter updates. In the standard algorithm, the mixing step is performed at each iteration (update) of the filter, resulting in all particle weights being equally weighted. However, for a movement model such as a navigation mesh using the debase strategy presented in Section 4.3.2, equating all weights per update is detrimental. The debase strategy penalizes particles that hit a wall or object with a lower weight. This penalty/information evolves over time, as explained in Section 4.3.2, resulting in an overall more robust transition density with less risk of impoverishment than, for example, directly removing a particle that runs into the wall. Some intentional variance in the posterior is therefore desirable and even beneficial in such

Algorithm 7 IMMPF Algorithm with Direct Sampling

Input: A Priori $P(m_1 \mid \boldsymbol{o}_1)$ and $p(\boldsymbol{q}_1 \mid m_1, \boldsymbol{o}_1)$

if $\bar{N}_{\text{eff}} \leq \bar{\hbar}$ **then** ▷ Mixing
  for $m_t = 0$ **to** M **do**
    for $i = 1$ **to** N_{m_t} **do**
      Sample $m_{t-1}^i \sim P(m_{t-1} \mid m_t, \boldsymbol{o}_{1:t-1})$
      Sample $\boldsymbol{X}_{t-1}^{i,m_t} \sim p(\boldsymbol{q}_{t-1} \mid m_{t-1}^i, \boldsymbol{o}_{1:t-1})$
      Set W_{t-1}^{i,m_t} to $\frac{1}{N_{m_t}}$
    end for
  end for
end if

Run: Parallel filtering for each $m_t \in M$ ▷ Filtering
for $i = 1$ **to** N_{m_t} **do**
  Sample $\boldsymbol{X}_t^{i,m_t} \sim p(\boldsymbol{X}_t^{i,m_t} \mid \boldsymbol{X}_{t-1}^{i,m_t}, \boldsymbol{o}_{t-1})$ ▷ Transition
  Compute $W_t^{i,m_t} \propto p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^{i,m_t}, \boldsymbol{X}_{t-1}^{i,m_t})$ ▷ Evaluation
end for
Calculate $\omega_t^{m_t} = \sum_{i=1}^{N_{m_t}} W_t^{i,m_t}$
Normalize W_t^{i,m_t} using $\omega_t^{m_t}$
if $N_{\text{eff}} \leq \hbar_{m_t}$ **then** ▷ Resampling
  Resample $\{W_t^{i,m_t}, \boldsymbol{X}_t^{i,m_t}\}$ to obtain N_{m_t} new equally-weighted particles $\{\frac{1}{N_{m_t}}, \overline{\boldsymbol{X}}_t^{i,m_t}\}$
end if
Calculate $P(m_t \mid \boldsymbol{o}_{1:t}) = \frac{\omega_t^{m_t} P(m_t|\boldsymbol{o}_{1:t-1})}{\sum_{m=1}^{M} \omega_t^{m_t} P(m_t|\boldsymbol{o}_{1:t-1})}$

a scenario. However, if all particles are now equally weighted by a continuous mixing step, this temporal information is lost and ultimately prevents the movement model from converging.

Another approach for implementing the mixing stage is the hybrid form introduced in [BD03]. Here, the particle sets are represented by a mixture of N Gaussian densities, approximating (5.2) according to

$$p(\boldsymbol{q}_t \mid m_t, \boldsymbol{o}_{1:t}) = \sum_{i=1}^{N} W_t^{i,m_t} \mathcal{N}(\boldsymbol{X}_t^{i,m_t}, v^{m_t} \operatorname{Cov}(\boldsymbol{q}_t, m_t)) \ , \tag{5.8}$$

where $v^{m_t} = 0.5 N^{-2/d^{m_t}}$ and d^{m_t} is the dimension of the state space. The covariance of the state $\operatorname{Cov}(\boldsymbol{q}_t, m_t)$ is given for each m_t according to

$$\operatorname{Cov}(\boldsymbol{q}_t, m_t) = \sum_{i=1}^{N} W_t^{i,m_t} (\boldsymbol{X}_t^{i,m_t} - \mathrm{E}_{(2.60)}(\Upsilon_t^{m_t}))(\boldsymbol{X}_t^{i,m_t} - \mathrm{E}_{(2.60)}(\Upsilon_t^{m_t})) \ , \tag{5.9}$$

where $\mathrm{E}_{(2.60)}(\Upsilon_t^{m_t})$ is the weighted average state over the particle set $\Upsilon_t^{m_t}$ of a respective mode m_t calculated using (2.60). Given (5.8) the joint posterior in (5.1) as thus solved using a sum of Gaussian probability densities weighted by the posterior mode probabilities. Obviously, this is identical to applying a KDE with Gaussian kernel to each mode. Mixing between modes can therefore be achieved by directly sampling new particles for each m_t from (5.5), as by inserting (5.8) into (5.5) it is expressed in closed form. Without proper approximation schemes like binned KDE [FM94] or boxKDE [Bul18] this however comes with high computational costs, especially in high dimensional state spaces. Furthermore, assuming a Gaussian kernel introduces additional approximations [DB05]. It is important to note that although this mixture of Gaussians is used to represent the posterior, it still accounts for the non-linear and non-Gaussian noise and dynamics of the system, as each mode is represented by an independently running particle filter. This is also the reason why it is called a hybrid form, as it assumes a hybrid between particle and mixture Gaussian representation.

5.1.3 Joint State Estimation

The state estimation of the hybrid form can be obtained by calculating the weighted average over all means of the involved Gaussian density functions. This is identical to the expected value $\mathrm{E}_{(2.60)}(\Upsilon_t^{m_t})$ weighted by the posterior mode probabilities (5.3), which can be applied as estimation for direct sampling. So for both mixing approaches, we can write

$$\bar{\boldsymbol{q}}_t^{\mathrm{wa}} = \mathrm{E}_{(5.1)}(\boldsymbol{q}_t) = \frac{\sum_{m=1}^{M} \mathrm{E}_{(2.60)}(\Upsilon_t^{m_t}) \cdot P(m_t \mid \boldsymbol{o}_{1:t})}{\sum_{m=1}^{M} P(m_t \mid \boldsymbol{o}_{1:t})} \ , \tag{5.10}$$

denoting a minimum (biased) variance estimator. As already discussed on Section 2.3.1, the variance greatly depends on the choice of the importance density, which is also the reason why the estimator is biased. This is also true here, but it should be noted that the IMMPF allows for different importance densities per mode, which worsens the joint posterior variance if an inappropriate importance is chosen.

In contrast to the hybrid form, the direct sampling approach allows for a wider range of estimation methods, as both the joint and mode posteriors are not represented by a mixture of Gaussians, but by the particles itself. In 2.3.3 we discussed different estimation strategies on particle sets. In principle, by including the posterior mode probabilities $P(m_t \mid \boldsymbol{o}_{1:t})$, they are also applicable to the IMMPF. The simplest approach is to replace the expected value $\mathrm{E}_{(2.60)}(\Upsilon_t^{m_t})$ in the (5.10) with another point estimator. This can be the same for each m_t, but it is also possible to use an individual estimator for each filter, emphasizing the modularity of the IMMPF approach. Depending on the application, the total number of particles over all modes can also

be considered to investigate effects such as multimodalities in the joint posterior. For example, in (multi-)target tracking, multiple position hypotheses can be generated using a mode-seeking algorithm such as mean-shift [GC11]. Although there are no proposals in the literature yet, another interesting use case would be a cluster analysis on video data, where each mode is based on different feature detectors and motion models. For single particle filters, approaches such as [TZ11] or [STW07] have already been presented in the literature, but not considered the IMMPF.

Depending on the application, there are countless other heuristics that can be used to make estimates. As we will see later, it is also valid not to use the joint posterior at all for point estimation, but to use one filter that plays a dominant role, while the others serve only as support, and have their particles correct the dominant filter via the mixing stage, depending on the events that occur [Fet17].

5.2 Model Fusion in Localization

In the derivation of IMMPF and its main tools above, we have seen the potential of this approach for localization and navigation. The property of being able to combine arbitrary movement and sensor models independently, as mentioned in the introduction to this chapter, is fulfilled by IMMPF. It does this by offering a high degree of modularity and creative freedom to achieve a joint estimate. One of the most exciting use cases for localization is the combination of spatially unevenly distributed sensors, e.g. a building may only have very accurate FTM or UWB transmitters in a certain area, while (not so accurate) BLE signals can be received everywhere else. The aim is therefore to prioritise a higher accuracy technology as soon as it becomes available. The transition between areas should be smooth and not abrupt. A similar strategy can be used in areas where some sensors are difficult to use or fundamentally susceptible to effects such as path loss or multipath, such as some wireless technologies in areas such as stairwells or underground car parks.

As mentioned at the end of Section 1.2.2, despite its good properties, the IMMPF is rarely used in sensor fusion applications compared to single-mode particle filters. To the best of our knowledge, there is no other work, other than our own, that explicitly addresses the indoor localization use case [Fet17; Fet23a]. However, there are a number of publications dealing with IMMPF-based methods that are, in principle, suitable for indoor use. For example, the authors of [XW19] proposes a TDOA-based positioning method using an IMMPF approach for tracking of a non-cooperative mobile transmitter. The method considers three motion models and two channel models to describe the trajectory of the terminal, e.g. a smartphone, in mixed LOS/NLOS conditions, which could be urban or indoor environments. By combining three

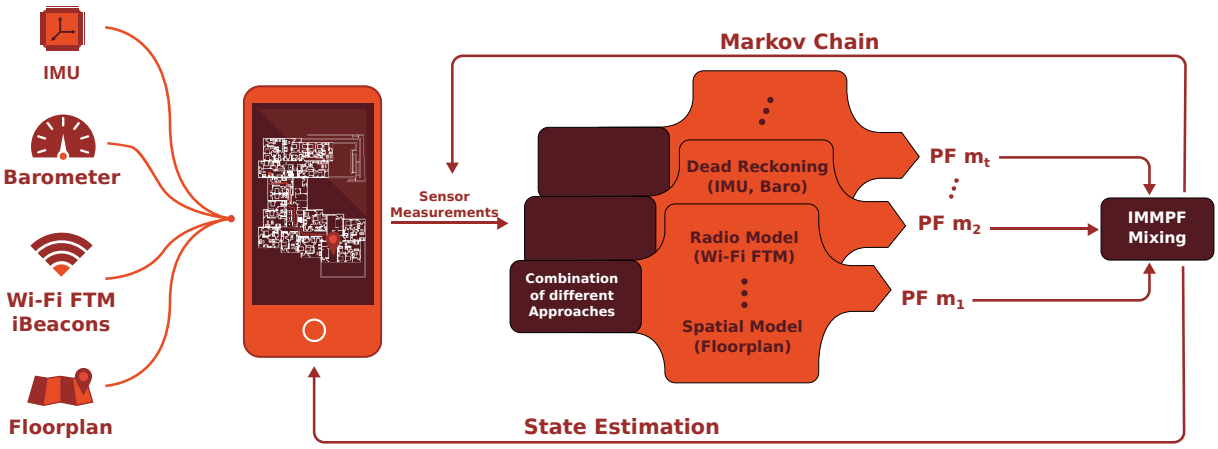


Figure 5.1: Simplified architecture overview of a smartphone-based ILS using IMMPF for state estimation. The different sensor measurements are input to an arbitrary set of parallel particle filters, each with an any number of sensor models. Their results are combined in the mixing stage, resulting in the joint state estimation. The relationships between the individual filters are given by a non-trivial Markov chain.

particle filter within the IMMPF, they achieve an average improvement of over $40\,\%$ throughout the entire tracking period compared to other state-of-the-art methods.

Another interesting approach which could also be applied in the field of indoor localization was proposed by the authors of [CT18]. They use an IMMPF for visual tracking of single pedestrians in videos, e.g. captured by surveillance cameras. They combine multiple movement models (constant velocity, constant acceleration and compensated motion model) and visual cues (color similarity and scale- and rotation-invariant SURF features) to achieve a more accurate approximation of the target state probability distribution. Experimental results demonstrate the scheme's robustness against various changes in target motion, occlusion, rotation, scaling, and illumination, outperforming other methods in terms of average RMSE and overlap ratio.

We have already introduced a general framework for building an ILS using particle filters in Section 2.5. They are defined by an arbitrary combination of different sensor models as presented in Chapter 3 and a motion model as presented in 4. This completes the filtering stage of the IMMPF for a single filter. Therefore, the main task left is to define the mixing process between these filters. For this, we developed a novel non-trivial Markov switching chain, that uses dynamic quality metrics for transitioning between modes. A simplified architectural overview of a smartphone-based ILS using all this is shown in Figure 5.1.

5.2.1 Mixing via Quality Metrics

Looking at the mixing process between (5.5) and (5.7), the transition matrix is primarily responsible for the combination between the respective modes. It controls the probability of the transition from m_t to $m_t + 1$, i.e. in simple terms, the purpose of this matrix is to decide how many particles are drawn from other filters and how many from its own particle set. The choice of $[\Pi_t]_{kl}$ is therefore crucial and forms the heart of the IMMPF.

In the literature, we only find IMMPF-based sensor fusion approaches that use a static $[\Pi_t]_{kl}$, i.e. a matrix that is constant in time and has fixed values. Examples are the work of [XW19], [CT18], [Wan12] or [KMK16] discussed above. The dynamics of mode switching is thus determined by $P(m_t \mid \boldsymbol{o}_{1:t})$, i.e. the posterior mode probability, which, as can be seen in the formula (5.4), strongly depends on the unnormalized weight sum $\omega_t^{m_t}$ of the particles, i.e. the evaluation. While the evaluation is able to model the noise through the sensor models, other events, such as sudden sensor failure or entering areas where sensors with increased accuracy are present, as described in the introduction to this section, are difficult to model and are often not considered.

As another example, let us take the Wi-Fi and Bluetooth models presented in Section 3.1. Entering a room with high signal attenuation will result in observations with high RSSI values, i.e. the multilateration would assume a position that is far away from the actual position of the pedestrian. However, if the particle set has a good representation in that region, e.g. assuming a simple constant velocity movement model for the transition, then according to $\omega_t^{m_t}$ the filter "thinks" it has a good approximation of the posterior, when in fact it does not. To put it more generally: When events occur that severely degrade the quality of a filter or its observations, this is often not reflected in the resulting posterior estimate of that filter because it is not accounted for in the respective models. As a consequence, the approximation error of the joint posterior becomes larger due to the unhandled events.

In practice, the unnormalized weight sum $\omega_t^{m_t}$ in (5.4) depends strongly on the implementation of the evaluation and its representation by the particle set. Note that the following considerations only apply if the modes (particle filters) of the IMMPF use different sensor models for the evaluation. Consider a mode that models the sensor noise with a very flat Gaussian, e.g. $\sigma = 10$. With the same set of particles scattered mainly around the mean and not representing the outer regions of the distribution, it will produce a lower $\omega_t^{m_t}$ (density) compared to a more compact distribution, e.g. $\sigma = 2$. This would therefore favor the mode with the compact distribution in the IMMPF mixing process. Of course, at this point one could argue that asymptotically both would give the same results, but this is only theoretically relevant. Also, the argument that a larger standard deviation σ automatically indicates that this filter is more uncertain and therefore produces less accurate results is not true in practice. Just because a sensor has more

noise does not necessarily mean that the estimate is worse. Factors such as update rate, noise behavior and ultimately the sensor model used are influencing factors. Particularly complex and non-Gaussian distributed sensor models, such as those proposed in this thesis, cannot be evaluated by variance alone, as has often been discussed in this thesis.

Therefore, when using a form like (5.4), it is desirable to provide a way to give an indicator of the quality of a filter, including all the models involved. If the quality of a filter decreases, i.e. it is suspected to make the joint posterior worse, then it should be given less weight in the mixing stage. An approach that makes this possible was proposed in our previous works [Fet23a] and [Fet17]. The novel idea is to define the Markov transition matrix $[\Pi_t]_{kl}$ not with constant values, but with a set of filter-specific quality metrics. These metrics are variable over time and are updated per filter step based on information such as the current point estimate, particle set or observations. This means that each filter is assigned a single or a combination of metrics that describes how good its quality is at the current time t and then influences the mixing process accordingly. Due to the resulting time-sequential properties for $[\Pi_t]_{kl}$, a non-trivial Markov chain results [RC13].

We distinguish between sensor-specific and non-sensor-specific metrics. The former are based directly on the physical properties and measurements of a sensor that could be observed, while the latter use information provided by the filtering process, such as the particle set or the analysis of the involved probability distributions. A metric itself is defined as a scalar $q \in [0, 1]$, i.e. a probability. In the following we will introduce several of these metrics, one for each sensor presented in Section 3 and a non-sensor specific one that can be applied to movement models as seen in Section 4. The choice of a quality metric obviously depends on the application, but we try to provide approaches that are as general as possible and at least applicable to the whole context of localization and navigation.

5.2.2 Quality Wi-Fi and Bluetooth

In Section 3.1 we have discussed several effects such as multipath and path loss, that influence the radio signal, adding noise to the RSSI measurements. In particular, the attenuation of the signal by materials such as reinforced concrete, fire doors and metalized glass through which the signal passes in an NLOS scenario increases the uncertainty of our proposed sensor models based on the radio propagation models introduced in Section 3.1.3. Furthermore, low coverage of transmitters degrades position estimation using multilateration, especially when having measurements from less then 3 transmitters. This often happens in isolated areas such as staircases or in corners of a building.

In general, this leads to two assumptions about the quality of Wi-Fi and BLE sensors. Firstly, if the RSSI values are low (e.g. $< -80\,\mathrm{dB}$) in all measurements, either there is no good coverage of transmitters in the vicinity, which means that the signal has to travel a longer distance and is therefore exposed to more noise factors, or there are strong attenuation effects. Both lead to increased uncertainty and ultimately poorer position estimation. Secondly, for optimal multilateration there needs to be a minimum number of transmitters in the building, in practice the more the better, but the absolute minimum in a two-dimensional scenario is obviously receiving 3 distinct measurements.

Based on the above considerations, we define the quality factor for Wi-Fi (BLE is defined identically) by the following combination

$$q_{\text{wifi}}(\boldsymbol{o}_t) = (q_{\text{rssi}}(\boldsymbol{s}_{\text{wifi}}) + q_{\text{num}}(\boldsymbol{s}_{\text{wifi}})) \cdot 0.5 \tag{5.11}$$

where

$$q_{\text{rssi}}(\boldsymbol{s}_{\text{wifi}}) = \max\left(0, \min\left(\frac{\bar{s}_{\text{wifi}} - l_{\text{min}}}{l_{\text{max}} - l_{\text{min}}}, 1\right)\right) \tag{5.12}$$

and

$$q_{\text{num}}(\boldsymbol{s}_{\text{wifi}}) = 1.0 - \exp(-0.1 \cdot |s_{\text{wifi}}|) \ . \tag{5.13}$$

Here, $\bar{s}_{\text{wifi}}$ is the average signal strength and $|s_{\text{wifi}}|$ the number of all signal strength measurements received from the observation $\boldsymbol{o}_t$. An upper and lower bound is given by l_{max} and l_{min}. Thus, (5.12) indicates the quality by using the measured RSSI values and (5.13) the number of measurements. Depending on the scenario, the latter could also be easily adapted to provide a statement about the number of different transmitters, instead of measurements.

These metrics are of course very simplified, as the propagation of WiFi and BLE signals is complex. A more sophisticated approach is the creation of a radio map as used by fingerprinting-based localization [KGA17] or for optimal placement of wireless local network access points [YLY19]. They provide a spatial representation of the signals propagation and thus offer a good starting point for more advanced analysis about the quality, e.g. by calculating the variance of the measurements with corresponding fingerprints. However, these approaches are not applicable in the general sense required above, as calibration is performed on a per-building basis.

A more general approach is proposed by the authors of [Yan18]. They use *channel state information* (CSI) to detect and classify interference in the $2.4\,\mathrm{GHz}$ frequency band. This can be a useful quality metric, especially in buildings with a large number of different wireless protocols (WiFi, BLE, Zigbee, etc.) and/or many receivers, such as an airport or exhibition center. In practice, however, consumer device operating systems such as iOS or Android do

not provide API access to more sophisticated information such as CSI. Finally, we have chosen the above quality metrics because they are easy to apply to any localization scenario, use only data that is already included in the filter observations, and are suitable for black-box APIs of consumer devices.

5.2.3 Quality Wi-Fi FTM

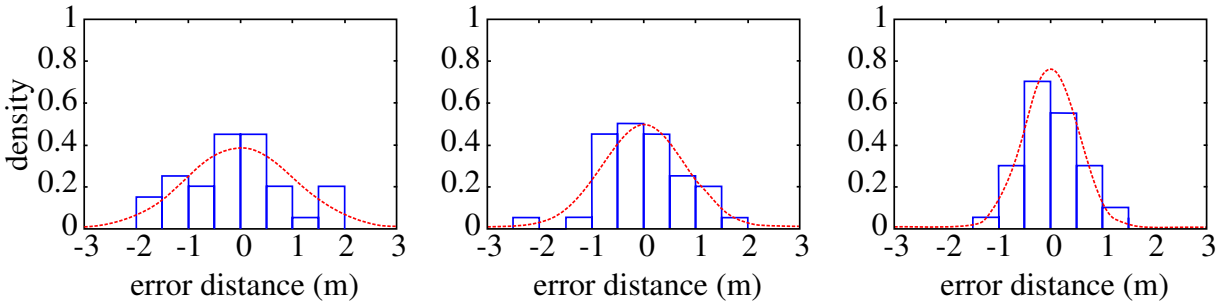
As FTM was purposely designed with localization in mind, existing APIs like the Android RTT API offer a convenient way to receive a distance measure by sending a ranging request to nearby access points. If the responding access point supports *location configuration information* (LCI), even the position of the responder is returned [Goo19]. As described in Section 3.2.1, most implementations use at least the default value of 8 for the number of FTM exchanges in a burst, although the IEEE 802.11-2016 specification allows a variety of configurations to be sent with the initial FTM request. Other parameters such as burst per session, burst interval or burst duration are mostly behind a black box.

The results of a successful measurement, i.e. a single FTM burst to an access point i, of the Android RTT API ranging request are the average distance $d_{i,m}$, its standard deviation $\sigma_{i,m}$ and the average RSSI $s_{i,m}$ value over all recorded signal exchanges of the burst, as well as other useful information such as the number of successful FTM exchanges, the bandwidth used to transmit the TWR frame or the channel center frequency. Here m of M_i is the count for each of these values within a single observation $\boldsymbol{o}_t$ of a filter update. The most obvious solution for a quality metric given the above is to use the standard deviation $\sigma_{i,m}$ as it provides a high level value and therefore does not require any special knowledge of the environment or FTM itself. A probability can again be expressed using an exponential function such as

$$q_{\text{dist}}(\boldsymbol{o}_t) = \exp(-1 \cdot \bar{\sigma}_{\text{ftm}}) \ , \tag{5.14}$$

where $\bar{\sigma}_{\text{ftm}}$ is the average for all sigma within the received observation $\boldsymbol{o}_t$. In rare cases, where the number of measures within the FTM burst is lower then 2, the Android RTT API returns 0 for $\sigma_{i,m}$. In the case where each $\sigma_{i,m} = 0$, we set $q_{\text{ftm}}(\boldsymbol{o}_t^{\boldsymbol{s}_{\text{ftm}}})$ to a heuristically chosen value such as 0.01.

As we have discussed in Section 3.2, the main error sources for FTM measurements are multipath, channel interference and relative permittivity. A high standard deviation $\sigma_{i,m}$ within a burst is more indicative of multipath and channel interference than the influence of the relative permittivity of intersecting materials. This is because, in theory at least, the permittivity on the straight line between transmitter and receiver should not change much within a very short



(a) 20 MHz bandwidth (b) 40 MHz bandwidth (c) 80 MHz bandwidth

Figure 5.2: Equal width histogram of the error distance between the measured FTM distance and the ground truth position for three different bandwidths. Measurements were taken at fixed intervals every $0.5\,\mathrm{m}$ from $0\,\mathrm{m}$ to $20\,\mathrm{m}$ along a straight line between initiator and responder. The red dashed line is a Gaussian distribution fitted to the histogram in blue. It can be seen that as the bandwidth increases, the variance of the Gaussian is reduced and therefore a lower range error can be assumed. The figure is based on [Ma22].

burst and is thus similar for each FTM interchange. However, as mentioned in Section 3.2.2, the analysis of these effects is not trivial and is still controversially discussed in the literature. Nevertheless, in order to obtain other quality metrics that can provide a statement about the position estimation of an FTM filter, especially due to NLOS scenarios, the frequency band used and the bandwidth might be valid indicators.

As shown by several authors like [Hor20b], [Bul22] or [Ibr18], a higher bandwidth allows for more accurate distance measurements. Here, the $5\,\mathrm{GHz}$ band provides the most options with $20\,\mathrm{MHz}$, $40\,\mathrm{MHz}$ or $80\,\mathrm{MHz}$, were the sampling rate increases with the size of bandwidth. This can be seen in Figure 5.2 for a simple LOS experiment. Here the authors of [Ma22] took distance measurements at fixed intervals every $0.5\,\mathrm{m}$ from $0\,\mathrm{m}$ to $20\,\mathrm{m}$ along a straight line between initiator and responder. Fitting a Gaussian distribution (red) to an equal width histogram (blue) of the distance error between measured distance and ground truth supports the above statement of improving accuracy with increasing bandwidth. Therefore, the position estimation of a filter using FTM benefits from a large bandwidth. According to [Hor22], the overall quality should also improve, as a higher sampling rate prevents multipath scenarios. Here, two or more signals often have to be distinguished with a separation smaller than the raw resolution. The finer this resolution, the less these effects should theoretically occur. The disadvantage of high bandwidth is the reduced number of radio channels. This can be a problem in scenarios where Wi-Fi is also used for wireless networking, resulting in high traffic causing access points to reject range requests [Ma22].

The use of bandwidth as a measure of quality ultimately depends on the configuration of the Wi-Fi infrastructure. For the ILS presented here, we use Espressif ESP32v2 microcontrollers

as Wi-Fi FTM access points, all configured in the $2.4\,\mathrm{GHz}$ band with a fixed channel width of $40\,\mathrm{MHz}$. The $5\,\mathrm{GHz}$ band and higher bandwidths are not supported with these devices. As all distance measurements are therefore made over the same bandwidth (and also channel), it makes no sense to consider it as quality metric for our application. Other devices available to end users that support FTM, such as Google Wi-Fi or the Aruba 500 series, use $5\,\mathrm{GHz}$ for FTM according to the IEEE 802.11-2016 protocol and, unless otherwise configured, automatically select channels based on proprietary and therefore unknown functions. A quality metric could be specified as the average probability given by a function $f_{\mathrm{B}}(B_{\mathrm{ftm}})$ that assigns a probability to each distance measurement in $\boldsymbol{s}_{\mathrm{ftm}} \in \boldsymbol{o}_t$ according to the bandwidth B_{ftm} reported for the FTM burst from which that distance originated. This can be written as

$$q_{\mathrm{B}}(\boldsymbol{o}_t) = \frac{1}{|\boldsymbol{s}_{\mathrm{ftm}}|} \sum_i \sum_m f_{\mathrm{B}}(d_{i,m} \mapsto B_{\mathrm{ftm}}) \ , \tag{5.15}$$

where $|\boldsymbol{s}_{\mathrm{ftm}}|$ is the number of all distance measurements received from a responder i with count m of M_i in the observations $\boldsymbol{o}_t$. A simple piecewise example for $20\,\mathrm{MHz}$, $40\,\mathrm{MHz}$ and $80\,\mathrm{MHz}$ of $f_{\mathrm{B}}(B_{\mathrm{ftm}})$ could be given by

$$f_{\mathrm{B}}(B_{\mathrm{ftm}}) = \begin{cases} \alpha_{\mathrm{ftm}} & B_{\mathrm{ftm}} = 80\,\mathrm{MHz} \\ \beta_{\mathrm{ftm}} & B_{\mathrm{ftm}} = 40\,\mathrm{MHz} \\ \gamma_{\mathrm{ftm}} & \text{otherwise} \ , \end{cases} \tag{5.16}$$

where α_{ftm}, β_{ftm} and γ_{ftm} are obviously the probabilities to assign. They are chosen heuristically and highly depend on the given scenario. A very conservative choice would be something like $\alpha_{\mathrm{ftm}} = 1.0$, $\beta_{\mathrm{ftm}} = 0.9$ and $\gamma_{\mathrm{ftm}} = 0.8$, slightly devaluing more narrow bandwidths.

Another factor influencing the error and thus filter quality is the choice of the channel (see again Section 3.2.2). As shown by [Hor22], distance measurements from different channels have uncorrelated error contributions. However, due to the complex behavior of signal propagation in unknown environments and to account for the automatic channel selection of different access point vendors, it is difficult to integrate this into a sensor model. This makes it a good fit for a quality metric that assigns a probability based on the variance of the included channels in $\boldsymbol{s}_{\mathrm{ftm}} \in \boldsymbol{o}_t$, which are identified by their channel center frequency. This could be given by a logarithmic interpolation of the form

$$q_{\mathrm{C}}(\boldsymbol{o}_t) = 1.0 + (\delta_{\mathrm{ftm}} - 1.0) \cdot \frac{\log(n_C)}{\log(N_C)} \ , \tag{5.17}$$

where $N_C \in \mathbb{N}_+$ is the number of all available channels of the Wi-Fi infrastructure, $n_C \in \mathbb{N}_+$ is the number of non-duplicate channel center frequencies from which the distances in $\boldsymbol{s}_{\text{ftm}}$ are reported and δ_{ftm} is the chosen minimum probability if all distances match a unique channel, i.e. $N_C = n_C$. If N_C is unknown due to no access to an existing Wi-Fi infrastructure, one could also choose $N_C = |\boldsymbol{s}_{\text{ftm}}|$, where $|\boldsymbol{s}_{\text{ftm}}|$ is again the number of all distance measurements in the observations $\boldsymbol{o}_t$.

Either with bandwith or channel information, it is again difficult to derive a statement about the relative permittivity. In general, the frequency dependence of the permittivity of many materials can be considered to be rather small and therefore would not allow any indication of quality. However, composite materials are highly frequency dependent, e.g. FR-4 (epoxy resin and glass fiber fabric) changes its permittivity by $10\,\%$ per GHz [Seb10]. Ultimately, creating a quality metric or sensor model for this would require impractical simulations and the creation of a spatial model with unfeasible information on all the materials used in walls, ceilings, stairs and so on. Instead, applying Wi-Fi and BLE quality metrics as given by (5.11) to the RSSI value received for each FTM burst is often more appropriate. This takes into account RSSI, i.e. attenuation through walls, as well as the number of nearby responders.

5.2.4 Quality Barometer

Although the barometer can be considered as a rather simple sensor model, it has enormous potential, since the correct and, above all, continuous detection of a floor change often prevents a complete failure of the particle set correctly approximating of the posterior [Fet23b]. Recalling the barometric formula (3.26), atmospheric pressure depends on a number of different parameters. These are strongly influenced by spatial and temporal factors, resulting not only in a continuous fluctuation in atmospheric pressure, but also in a cumulative error. The use of low quality MEMS barometers to measure the pressure also results in strong non-Gaussian distributed noise that is very sensitive to rapid changes such as the opening of a window or door.

The sensor model proposed in Section 3.4 can ignore some of the above factors by considering only the difference between two barometric pressure measurements. This relative consideration is based on the assumption that there is not too much time between the current measurement and the reference measurement, which could include a large change in external conditions, e.g. a major weather change. Therefore, for a quality metric, it may be useful to consider the standard deviation between a number of directly consecutive measurements, similar to FTM. This can again be described by an exponential function of the form

$$q_{\text{baro}}(\boldsymbol{o}_t) = \exp(-k_{\text{baro}} \cdot \bar{\sigma}_{\text{baro}}) \ , \tag{5.18}$$

where k_{baro} controls the rate of the decay (sharpness) and $\bar{\sigma}_{\text{baro}}$ is the above mentioned standard deviation, e.g. for a time frame of $2\,\text{s}$ or the last 10 measurements. Obviously other functions can also be used to provide a probability, such as a simple fractional of the form

$$q_{\text{baro}}(\boldsymbol{o}_t) = 1 - \frac{\bar{\sigma}_{\text{baro}}}{k_{\text{baro}} + \bar{\sigma}_{\text{baro}}} \ , \tag{5.19}$$

where k_{baro} now represents the pressure at which the probability is halved or a Taylor series expansion like

$$q_{\text{baro}}(\boldsymbol{o}_t) = P(k_{\text{baro}}) + P'(k_{\text{baro}})(\bar{\sigma}_{\text{baro}} - k_{\text{baro}}) \ , \tag{5.20}$$

where $P'(k_{\text{baro}})$ controls the steepness of the curve and $P(k_{\text{baro}})$ is the probability of the atmospheric pressure given by k_{baro}, e.g. $P(k_{\text{baro}}) = 0.5$ again represents the pressure at which the probability is halved. Essentially, the use of a variety of different functions to represent a probability applies to all quality metrics. The function should be chosen depending on the application and the sensors involved.

However, all of the above suffer from the same drawback when using the standard deviation of the relative pressure readings. While this works great for a $\bar{\sigma}_{\text{baro}}$ that is expected to be as small as possible over the entire time, changing a floor will necessarily increase the relative pressure for the short period of time it takes to start and stop the ascent/descent. This leads to an increasing $\bar{\sigma}_{\text{baro}}$ and ultimately a lower quality for a short period of time. Of course, this can actually be an advantage, since a rapid change in altitude can lead to an increasing error, e.g. in an elevator or an isolated staircase. To compensate for this behavior, we need to manipulate k_{baro} in such situations so that the function is flatter and does not drop so sharply. A good solution is to use activity recognition as presented in 3.5 to detect these situations and then set k_{baro} accordingly. It is important to ensure a good synchronization between the activity determined by the window function and the standard deviation, e.g. by using an identical window size.

Another value that can indicate the quality of the use of relative pressure readings is the age of the reference measurement used. The older the reference, the less reliable the barometric sensor model and its implementing filter. However, this is highly dependent on the current weather conditions, the building and also the sensors used, and therefore requires a very individual metric, which we try to avoid.

5.2.5 Quality Activity Recognition

There are many factors in the entire pipeline of the activity recognition process that influence the quality. In the case of the analytical transformation in Section 3.5, this starts with the training data used: Are they large enough, do they have enough diversity from as many subjects

as possible, or do the IMUs used match those in the live system? For the sake of simplicity, we assume a virtual black-box sensor for activity recognition, as may be the case in practical applications. This means that we have no insight into the training data or the exact implementation of the pipeline. Thus, it is similar to physical sensors that deliver better or worse results depending on the hardware or driver.

The result of the activity recognition is either the discrete activity itself or a vector with the corresponding probabilities per activity. As a sliding window is utilized, the update rate depends on the slowest sensor involved. This makes it possible to derive a quality metric in the discrete case by observing the number of changes within a given time frame of successive detected activities. The more often the reported activity changes within a short period of time, the greater the uncertainty and the worse the quality of the result. Similar to the quality metric of the barometer, such behavior can happen especially in edge cases, i.e. when the activity changes and artificially worsens the quality. A probability can be given analogous to (5.17).

In contrast, the continuous case can be handled more easily without requiring for a history. A quality metric can be derived from the given probabilities of the result vector $\boldsymbol{\Omega}_t$, which is normalized to 1. The more probable an activity is, the more certain the model is that it is the activity the pedestrian is currently performing. However, if two or more values of $\boldsymbol{\Omega}_t$ are similar, no definite statement can be made, as the uncertainty increases and therefore the quality of the recognition result decreases. Based on this, a quality metric can be specified using the maximum of $\boldsymbol{\Omega}_t$ as a variation of the sigmoid function as follows

$$q_{\text{act}}(\boldsymbol{o}_t) = \frac{1}{1 + e^{2+\left(10\cdot\left(\max(\boldsymbol{\Omega}_t)+\frac{1}{|\boldsymbol{\Omega}_t|}\right)\right)}} \quad . \tag{5.21}$$

The constant value of 10 controls the slope of the sigmoid, resulting in a curve that is close to 1.0 if $\max(\boldsymbol{\Omega}_t)$ is also close to 1.0. The other constant values of 2 and $\frac{1}{|\boldsymbol{\Omega}_t|}$, where $|\boldsymbol{\Omega}_t|$ gives the number of activities that can be recognized, control the shift of the s-shaped curve so that the resulting probability of $q_{\text{act}}(\boldsymbol{o}_t)$ is somewhere near 0.1 when $\boldsymbol{\Omega}_t$ is uniformly distributed.

5.2.6 Quality Movement Model

The last quality metric presented is of a more general nature and could be used in many different scenarios. It describes the compactness of a particle set by examining the variance of the Euclidean distances of all particles to each other. The metric is given by

$$q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) = \exp\left(-\text{Var}\left(\sum_{i=1}^{N_{m_t}}\sum_{j=1}^{N_{m_t}} ||\boldsymbol{\rho}_i - \boldsymbol{\rho}_j||\right)\right) \quad , \tag{5.22}$$

where $\boldsymbol{\rho} = (x, y, z)^T$ is again the position of a particle. Such a compactness measure is excellent for evaluating the quality of a movement model [Fet23a]. Without considering sensor noise, all particles should move the same distance in the same direction. Given a known starting point, this would result in a single position. The greater the uncertainty due to noise and model error, the greater the variance. For example, if a step is not correctly detected, a large number of particles could run into a wall or miss a change in direction. This often leads to multimodal densities that greatly affect state estimation. In particular, the movement model with its relative sensor data then takes some time to converge again via strategies such as sink or swim (see Section 4.2.3), i.e. the quality of the filter improves only slowly.

In addition to noise and model errors, the spatial model in particular affects the quality of a movement model. In narrow corridors, particles have very little freedom of movement, which inevitably leads to a more compact set of particles and thus higher quality. This is in contrast to large open areas, such as an airport lobby, where the current uncertainty is hardly constrained by walls or other obstacles. Ultimately, it depends on the application and the required accuracy of the system. In general, however, it can be said that our proposed ILS, at least, will benefit from more angled buildings with many small rooms, based on a movement model with a corresponding spatial model [Fet18].

5.2.7 Choices of Transition Matrix

As discussed above, the transition matrix in (5.6) and (5.7) is primarily responsible for the combination between the respective modes. It controls the probability of the transition from m_t to m_{t+1}, i.e. in simple terms, the purpose of this matrix is to decide how many particles are drawn from other filters and how many from its own particle set. The choice of $[\Pi_t]_{kl}$ is therefore crucial and forms the heart of the IMMPF.

The structure of the matrix depends of course on the setup of the respective filters. In the simplest case you have only one sensor per filter, i.e. one filter that e.g. only evaluates Wi-Fi FTM and one filter that only does the transition using a movement model. A matrix for this combination of the two filters would then look like this:

$$\Pi_{\text{pdr+ftm}} = \begin{pmatrix} q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) & 1 - q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) \\ 1 - q_{\text{ftm}}(\boldsymbol{o}_t) & q_{\text{ftm}}(\boldsymbol{o}_t) \end{pmatrix} \quad . \tag{5.23}$$

The quality metrics are chosen according to (5.14) and (5.22). A similar two-dimensional real square matrix for BLE and a movement model is given by

$$\Pi_{\text{pdr+ble}} = \begin{pmatrix} q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) & 1 - q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) \\ 1 - q_{\text{ble}}(\boldsymbol{o}_t) & q_{\text{ble}}(\boldsymbol{o}_t) \end{pmatrix} \quad , \tag{5.24}$$

using (5.11) and (5.22) respectively. Here, $[\Pi_t]_{1,2}$ and $[\Pi_t]_{2,1}$ give the probability for the respective modes to draw particles from each other, i.e. if $[\Pi_t]_{1,2} = 1 - q_{\text{pdr}}(\boldsymbol{X}_t^{m_t})$ is near 1.0, the PDR quality is low and thus particles are more likely to be drawn from the radio-based filter into the PDR-based filter, instead from the PDR-based filter itself. Thus, within the matrix, the row index i indicates the mode m_{t+1} under consideration, while the column index j indicates the conditioned mode m_t from which particles are drawn into m_{t+1}.

Obviously, the choice of $[\Pi_t]_{kl}$ highly depends on the given scenario, i.e. involved sensors, environment and particle filter design. The degrees of freedom in the construction of the matrix are manifold and allow for numerous combinations of arbitrary particle filters. For example, the matrices $\Pi_{\text{pdr+ble}}$ and $\Pi_{\text{pdr+ftm}}$ given above are built with office buildings in mind, where radio-based technology often lacks coverage in areas such as stairwells, but has good coverage in more open areas such as a cafeteria. Thus, the PDR-based filter provides accurate results when radio quality is low, and the radio-based filter does the opposite when the uncertainty of the movement model increases due to missing constraints such as walls or other obstacles [Fet23a].

In general, it can be said that the approach we have developed for applying the IMMPF allows true modularity, since even complete ILS can be combined if a quality metric is provided. As an example, consider two particle filters, one that uses only relative data provided by the smartphone's IMU, including motion model, activity detection, and barometer, and another that uses both BLE and Wi-Fi FTM, as these are not entirely similar but are often affected by the same environmental influences. The transition matrix could be given by

$$\Pi_{\text{rel+abs}} = \begin{pmatrix} q_{\text{rel}}(\boldsymbol{o}_t, \boldsymbol{X}_t^{m_t}) & 1 - q_{\text{rel}}(\boldsymbol{o}_t, \boldsymbol{X}_t^{m_t}) \\ 1 - q_{\text{abs}}(\boldsymbol{o}_t) & q_{\text{abs}}(\boldsymbol{o}_t) \end{pmatrix} \quad , \tag{5.25}$$

where $q_{\text{rel}}(\boldsymbol{o}_t, \boldsymbol{X}_t^{m_t}) = \frac{1}{3}\, q_{\text{pdr}}(\boldsymbol{X}_t^{m_t}) q_{\text{act}}(\boldsymbol{o}_t) q_{\text{baro}}(\boldsymbol{o}_t)$ denotes the quality metric of the relative filter and $q_{\text{abs}}(\boldsymbol{o}_t) = \frac{1}{2}\, q_{\text{ble}}(\boldsymbol{o}_t) q_{\text{ftm}}(\boldsymbol{o}_t)$ for the absolute radio-based filter accordingly. Other constellations, such as combining different sensor models, e.g. a BLE model based on radio propagation and a fingerprint-based model for buildings where not every area is fingerprinted, are also conceivable.

By using our approach of dynamically updating Π_t for each filter update, other interesting things can be achieved. For example, in Section 3.3 and Section 4 we discussed that it is hard

to provide an accurate step length detection, so we decided to use fixed values that change according to the currently detected activity. For this we introduce the discrete piecewise function (4.5), which provides a distance per particle by sampling from the activity $\mathbf{\Omega}$. In a more general and continuous sense, this can also be achieved using an IMMPF on a per-filter basis. Here each detected activity is represented by its own particle filter, which has a movement model with a unique parameterization for step length μ_{step} and sigma σ^2_{step} according to (4.3). The transition matrix is then denoted by

$$\Pi_{\text{act}} = \begin{pmatrix} \mathbf{\Omega}_t^{\text{walk}} & \mathbf{\Omega}_t^{\text{stand}} & \mathbf{\Omega}_t^{\text{asc}} & \mathbf{\Omega}_t^{\text{desc}} \\ \mathbf{\Omega}_t^{\text{walk}} & \mathbf{\Omega}_t^{\text{stand}} & \mathbf{\Omega}_t^{\text{asc}} & \mathbf{\Omega}_t^{\text{desc}} \\ \mathbf{\Omega}_t^{\text{walk}} & \mathbf{\Omega}_t^{\text{stand}} & \mathbf{\Omega}_t^{\text{asc}} & \mathbf{\Omega}_t^{\text{desc}} \\ \mathbf{\Omega}_t^{\text{walk}} & \mathbf{\Omega}_t^{\text{stand}} & \mathbf{\Omega}_t^{\text{asc}} & \mathbf{\Omega}_t^{\text{desc}} \end{pmatrix} , \tag{5.26}$$

using the probabilities given by the activity recognition as a quality metric. Each row automatically sums to 1, since $|\mathbf{\Omega}| = 1$ is already normalized. Another advantage of this notation is that it is easy to add new activities, even from different machine learning models.

There is still an open question about the transition matrix: How to construct a matrix like (5.25) but with a larger number of quality metrics, i.e. not just a two by two matrix. Since the rows of Π_t must sum to 1, a row-by-row normalizer is required. Assume that, similar to (5.26), each value of a column is denoted by the quality metric of the corresponding filter m_t. In the simplest case, a normalization can be done by using a row-wise linear function of the form

$$f_{\text{linear}}(\Pi_t)_{i,j} = \frac{x_{i,j}}{\sum_k x_{i,k}} \tag{5.27}$$

where x denotes an arbitrary quality metric at position $[\Pi_t]_{i,j}$ within the transition matrix Π_t. If non-linearity is desired, e.g. to make small values less important, something like a softmax function of the form

$$f_{\text{softmax}}(\Pi_t)_{i,j} = \frac{e^{x_{i,j}}}{\sum_k e^{x_{i,k}}} , \tag{5.28}$$

can be applied. However, if all the filters have a low uncertainty, i.e. the quality is high, the particles will be drawn evenly across all modes. This can dilute the results and we lose diversity between modes. To prevent this, we need a different normalization strategy that only draws from another mode when the quality of its own filters is low, i.e. an approach similar to (5.23) and (5.24) only for arbitrarily large matrices. For this, we suggest a function of the form

$$f_{\text{keep}}(\Pi_t)_{i,j} = \begin{cases} \frac{(1-x_{i,i}) \cdot x_{i,j}}{\sum_{k \neq i} x_{i,k}} & i \neq j \\ x_{i,j} & i = j \end{cases} , \tag{5.29}$$

where we keep the quality value of the diagonal of Π_t, i.e. the transition probability of drawing particles from itself, and normalize all other remaining quality values within the current row i depending on the difference $(1 - x_{i,i})$. This still ensures a row-wise normalization to 1 of Π_t, by keeping up the diversity between modes. If the quality of one mode becomes too low, the other modes are used according to their current quality. The lower the own quality, the bigger the difference $(1 - x_{i,i})$ and the more influence the other modes have. So if all modes have a high uncertainty, i.e. a low quality, the particles will primarily be drawn from the other modes and not from the own mode. If this behavior is not desired, the difference can be limited to a lower limit, e.g. with $\max((1 - x_{i,i}), 0.5)$.

Chapter 6

Particle Distribution Optimization

Despite the introduction of SIS/SIR extensions like APF (see Section 2.4.2) or CONDENSATION (see Section 2.4.1) in the last decades, the handling of the systematic problems of degeneracy and impoverishment is still an open problem. Both problems have been discussed in great detail several times during Chapter 2. In a nutshell: When using an SIS approach (cf. Algorithm 3) for state estimation, the recursive behavior leads to a concentration of weights on only a few particles, while the remaining particles have very low weights. To avoid such a poor representation of the target posterior, the resampling step was introduced in Section 2.3.2. Based on a predefined scheme, particles are *re*sampled to discard low weighted particles and replicate high weighted ones. This leads to a better representation of the target, as the resampled particle set is distributed according to the approximation and not just the importance distribution, and makes resampling an inevitable step in SMC. However, as the authors of [Che03] pointed out very early on, resampling does not completely solve the degeneracy problem, but leads in the long run to a sometimes even worse phenomenon, impoverishment. This occurs due to a decreasing diversity of particles, as low weight particles are discarded and only duplicates of high weight particles are preserved. This leads to a high concentration of them at similar positions and therefore a poor representation of the underlying probability density and ultimately worse estimation results. In the context of sensor fusion, the problem can be so severe that the estimator completely loses track, gets stuck, and never recovers [Fet17].

Depending on the scenario, other effects besides the resampling itself can exacerbate the impoverishment problem. An example in the context of indoor localization is illustrated in Figure 6.1. Here we use a movement modal that restricts walking through walls and other obstacles, similar to the approaches presented in Section 4. At time $t-1$, the position estimation trajectory (green line) deviates from the ground truth (black line) and the posterior distribution becomes multimodal, i.e. the particle set splits into two subsets, one that keeps walking straight

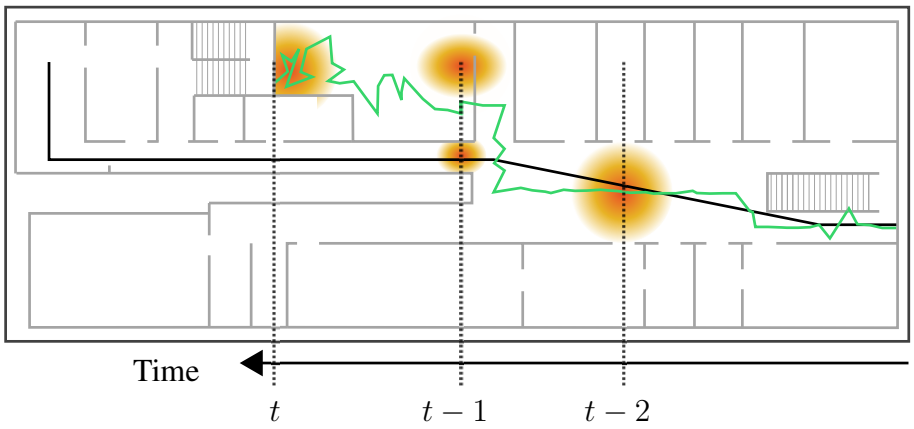


Figure 6.1: Sample impoverishment due to a movement model that restricts walking through walls in an indoor localization scenario. At time $t-1$, the approximate position trajectory (green line) deviates from the true trajectory (black line) due to the influence of uncertain measurements and an incorrectly detected turn. As a result, the posterior distribution at time t is confined to the spatial boundaries of the environment and is unable to recover as no particles are sampled outside.

and one that goes up into a separate space. This happens due to the influence of uncertain measurements and an incorrectly detected turn. As a result, the posterior distribution at time t is confined to the spatial boundaries of the environment and is unable to recover, since no particles remain outside the space.

In a general sense, solving for effects that affect the approximation behavior of the particle set is known as *particle distribution optimization* (PDO). In the following, we will discuss several such approaches. We start with general methods and then move on to data-driven methods specifically designed for the target tracking and localization use case.

6.1 General Approaches

In PDO, an approach is general in the sense that it is independent of the application and the sensor and movement models used. In particular, the particle itself, its weight, the quality of the approximation, and the overall uncertainty of the state estimate are considered and inferences are derived. It should be noted, however, that while these approaches are general in nature and therefore can be applied to any use case, they do not necessarily represent a true improvement for the use case. The complexity of the sensor fusion systems discussed here is still very high and finding the universal solution for PDO is still an open and perhaps unsolvable task.

Some PDO approaches like deterministic resampling or mean-shift estimation have already been described in Section 2.3.2 and Section 2.1. They improve the particle filter overall with

the goal of a more accurate state estimation. However, the methods presented in this chapter explicitly aim at fighting the impoverishment effect. A very good, but unfortunately somewhat older, overview of PDO methods can be found in [Li14].

6.1.1 Kullback-Leibler Sampling and Resampling

Consider again a situation like the one shown in Figure 6.1. One way to still have particles near the true trajectory at time t would be to have a larger total number of particles. This allows for more different hypotheses and would allow some particles to still follow this trajectory. However, having a very large number of particles over the entire filtering process brings several drawbacks, most notably an increase in computational complexity and higher sensitivity to outliers [Che03; TBF05].

For this reason, the authors of [KF98] and [Fox99] came up with the idea of an adaptive particle size over time. They adjust the size based on the sum of the non-normalized likelihoods (particle weights) of the evaluation step. This is easily done by predefining a threshold for the sum, so that particles are either removed or added to get as close to that threshold as possible. The main idea is that if the sample set has a good representation of the distribution provided by the sensor models, then each individual particle weight will be rather large and a small number of particles will be sufficient. On the other hand, if the sensor models have a large uncertainty due to erroneous sensor readings, which leads to impoverishment, the weights of individual particles will be small, and thus the sample set will become larger.

According to [Fox01] this approach directly relates to the property that the variance of the evaluation step is a function of the mismatch between the importance density function $q(\boldsymbol{q}_{1:t})$ and the target distribution $\tau(\boldsymbol{q}_{1:t})$ that is being approximated. Unfortunately, this mismatch is not always an accurate indicator for the necessary number of particles. Imagine a situation where the particle set approximates a multimodal density, i.e. there are for example three different point clouds with some distance between them, one of which reflects the true position. The sum of the weights resulting from the evaluation is then similar to the sum that would result from a single point cloud around the true position. The number of particles is therefore similar, although in the multimodal situation a higher number of particles would be required due to the greater uncertainty. Furthermore, determining a suitable threshold depends highly on the scenario and the sensors used.

To overcome this limitations, the authors of [Fox01] introduced the *KLD-sampling* approach. Its name is derived from the well-known *Kullback-Leibler divergence*, which is a non-symmetric non-negative difference (often called distance) between two probability distributions

Q and P [CB21]. For the discrete case it can be written according to

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{Q}} P(x) \cdot \log \frac{P(x)}{Q(x)} \ , \tag{6.1}$$

where $\mathcal{Q}$ denotes the state space, which needs to be the same for P and Q. It is also stated as the amount of information lost when Q is used to approximate P. The main idea of KLD-sampling is to determine the distance, in this case one could also speak of approximation error, between the current posterior $p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ and the true target distribution $\tau(\boldsymbol{q}_{1:t})$. For each filter update, the number of particles is then determined using a statistical bound ϵ (threshold) on approximation quality, so that with probability $1-\delta$, the error between $p(\boldsymbol{q}_{1:t} \mid \boldsymbol{o}_{1:t})$ and $\tau(\boldsymbol{q}_{1:t})$ is less then ϵ [TBF05].

The derivation of KLD-sampling can be found in [Fox01]. To derive the bound ϵ it is assumed that the true posterior $\tau(\boldsymbol{q}_{1:t})$ is based on a discrete, piecewise constant distribution such as a histogram, or grid, overlayed over the multi-dimensional state space. The probability for each bin k of the histogram is specified by $\bar{p} = p_1 \dots p_k$, providing the true distribution. In contrast, the posterior, in this case the MLE of $\bar{p}$, is denoted by $\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}} = n^{-1}\bar{\boldsymbol{x}}$, where the vector $\bar{\boldsymbol{x}} = (x_1 \dots x_k)$ holds the number of particles drawn from each bin, e.g. according to a multinomial distribution $\bar{\boldsymbol{x}} \sim f_{\mathrm{multi}_k}(n, \bar{p})$ with $N = |\bar{\boldsymbol{x}}|$. By means of (6.1) the likelihood-ratio λ_N for testing $\bar{p}$ is given by

$$\log \lambda_N = \sum_{j=1}^{k} x_j \log\left(\frac{\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}}}{\bar{p}}\right) = N \sum_{j=1}^{k} \hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}} \log\left(\frac{\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}}}{\bar{p}}\right) \ , \tag{6.2}$$

and as $N \mapsto \infty$, it converges to a chi-square distribution of the form

$$2\log\lambda_N \mapsto \chi^2_{k-1} \ . \tag{6.3}$$

To determine the probability that the KLD-divergence $D_{\mathrm{KL}}(\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}}\|\bar{p})$ is smaller then the threshold ϵ, given that N particles are drawn from $\bar{p}$, we can assume that

$$P_{\bar{p}}(D_{\mathrm{KL}}(\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}}\|\bar{p}) \le \epsilon) = P_{\bar{p}}(2N\ D_{\mathrm{KL}}(\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}}\|\bar{p}) \le 2N\ \epsilon) \approx P(\chi^2_{k-1} \le 2N\epsilon) \ . \tag{6.4}$$

To guarantee with a probability of $1-\delta$ that the above is true we utilize the quantiles of the chi-square distribution according to

$$P(\chi^2_{k-1} \le \chi^2_{k-1,1-\delta}) = 1-\delta \tag{6.5}$$

and by choosing N such that $2N\epsilon$ is equal to $\chi^2_{k-1,1-\delta}$, (6.4) and (6.5) are combined to result in the definition

$$P_{\bar{p}}(D_{\mathrm{KL}}(\hat{p}_{\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}} \| \bar{p}) \leq \epsilon) \approx 1 - \delta \tag{6.6}$$

The number N of particles can now be determined using

$$N = \frac{1}{2\epsilon} \chi^2_{k-1,1-\delta} \ , \tag{6.7}$$

where it is guaranteed that with probability $1 - \delta$, the KLD-divergence is less then ϵ. To efficiently compute the quantiles of the chi-squared distribution, a Wilson-Hilferty transformation can be used as an approximation, yielding

$$N = \frac{1}{2\epsilon} \chi^2_{k-1,1-\delta} \approx \frac{k-1}{2\epsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta} \right\}^3 \ , \tag{6.8}$$

where $z_{1-\delta}$ is the upper $1 - \delta$ quantile of the standard Gaussian $\mathcal{N}(0, 1)$.

The original paper by [Fox01] gives further notes on the properties of the (6.8), most importantly that the required number of particles is proportional to the inverse of the ϵ bound, and first order linear in the number k of bins with support. Each bin k of the histogram H, i.e., the discrete distribution of the true posterior, is either with support, i.e., $n > 0$ particles, or without support, i.e., $n = 0$ particles. In the context of particle filtering, this means that the true posterior, which is of course unknown, is not required to be known, and it is sufficient to determine k. This can be done either during sampling [Fox01] or during resampling [LSS13] by estimating k by counting the number of particles that fall into an empty bin or not. We then sample new particles as long as the threshold in (6.8) is not exceeded.

With this, KLD-sampling is able to provide an adaptive number of particles for each iteration of the particle filter. The complete process is shown in Algorithm 8 as an extension of the CONDENSATION particle filter shown in Algorithm 5. Besides the a priori $\mu(\boldsymbol{X}_1^i)$, the KLD-sampling requires the parameters δ, ϵ, an initial number for N, a maximum size N_{max}, and the discrete distribution in histogram form H. In the context of indoor localization, H can be implemented as a three-dimensional grid, similar to the one presented in 4.2. Before starting the loop, these parameters must be reset. From line 3 to line 9, the normal filtering process is performed for each particle, i.e. a new particle is sampled according to the transition and weighted based on the evaluation. The KLD-sampling itself is implemented in lines 10 to 16. If the newly sampled particle falls into an empty bin b of H, the number k of bins with support is incremented and b is marked accordingly. In line 13 we then calculate the number of particles N needed to reach our bounded ϵ. This number is updated for each newly drawn particle that

Algorithm 8 CONDENSATION with KLD-Sampling

Input: A Priori $\mu(\boldsymbol{X}_1^i)$, δ, ϵ, H, N, N_{max}

$k = 0$, $M = 0$, Empty bins b in H ▷ Reset
repeat
  if $t = 1$ **then** ▷ Initialization
    Sample $\mu(\boldsymbol{X}_1^i)$
    Compute $W_1^i \propto p(\boldsymbol{o}_1 \mid \boldsymbol{X}_1^i)$
  else if time $t \geq 2$ **then**
    Sample $\boldsymbol{X}_t^i \sim p(\boldsymbol{X}_t^i \mid \boldsymbol{X}_{t-1}^i)$ ▷ Transition
    Compute $W_t^i \propto p(\boldsymbol{o}_t \mid \boldsymbol{X}_t^i)$ ▷ Evaluation
  end if
  if $\boldsymbol{X}_t^i$ falls into empty bin b of H **then** ▷ KLD-Sampling
    $k = k + 1$
    $b =$ non-empty
    $N = \frac{k-1}{2\epsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta} \right\}^3$
  end if
  $M = M + 1$
until $M \geq N$ or $M \geq N_{\text{max}}$

if $N_{\text{eff}} \leq \hbar$ **then** ▷ Resampling
  Resample $\{W_t^i, \boldsymbol{X}_t^i\}$ to obtain N new equally-weighted particles $\{\frac{1}{N}, \boldsymbol{X}_t^i\}$
end if

falls into an empty bin, and can thus be thought of as a moving target/threshold for M. While at the beginning of the processing almost all bins are empty, this leads to a continuous increase of k and therefore N. As more and more bins have support, N is updated only occasionally until M finally exceeds it, ending the update loop.

The time at which the KLD-sampling is done obviously depends on the variance of the particles. The more scattered they are, the more bins will be supported and the higher N will be. In the case of CONDENSATION, this depends only on the transition function, since the weight has no direct effect on the KLD-sampling. This means that, similar to the reasons we introduced resampling in Section 2.3.2, the particles are drawn from the importance distribution rather than the (true) posterior. Thus, the mismatch between the two is ignored in the above algorithm. For this reason, the authors of [LSS13] incorporate KLD-sampling into the resampling step; called *KLD-resampling*. The implementation is straightforward and can be seen in Algorithm 9.

This approach is very easy to implement and offers a high degree of modularity, since it does not change other parts of the particle filter. However, what [LSS13] does not mention is that by resampling particles from the particle set denoted by the last filtering process, a delay

Algorithm 9 CONDENSATION with KLD-Resampling

Input: A Priori $\mu(\boldsymbol{X}_1^i)$, δ, ϵ, H, N, N_{max}

for $i = 1$ **to** N **do**
    Run CONDENSATION as in Algorithm 5
end for

if $N_{\text{eff}} \leq \hbar$ **then** ▷ Resampling
    $k = 0$, $M = 0$, Empty bins b in H ▷ Reset
    repeat
        Sample $\{\frac{1}{N}, \boldsymbol{X}_t^i\} \sim \Upsilon_t = \{W_t^i, \boldsymbol{X}_t^i\}_{i=1}^N$ based on a resampling strategy
        if $\boldsymbol{X}_t^i$ falls into empty bin b of H **then** ▷ KLD-Sampling
            $k = k + 1$
            $b =$ non-empty
            $N = \frac{k-1}{2\epsilon} \left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta}\right\}^3$
        end if
        $M = M + 1$
    until $M \geq N$ or $M \geq N_{\text{max}}$
end if

of one update step is introduced to the adaptive sample size. Unlike KLD sampling, where an increase in N directly causes new particles to be drawn, N is determined in the resampling step and thus after filtering. The problem is that during resampling, new particles are drawn e.g. multinomially from a known set, so no new position is created. New positions are only calculated in the transition and thus the filtering process. As a result, the adaptive size of the particle set determined based on the weight and particles at time t is only used at time $t+1$. For short update intervals this may not be noticeable, for larger ones with many measurements it is.

In summary, KLD-sampling or -resampling provides a general tool that can be used by any particle filter. In a very simplified sense, they prevent having too many particles in a situation where this is not necessarily required, and on the other hand allow a larger size when having a high uncertainty. The property of having a larger number of particles available with increased uncertainty may limit some impoverishment effects. However, since both methods rely exclusively on the transition/importance distribution to sample new particles, the restrictive movement models used in this work still limit the ability to obtain particles in spatially truncated regions. Thus, any impoverishment effects caused by such assumptions cannot be corrected.

6.1.2 Kernel Density Resampling

From an analytical point of view, instead of resampling from the particle set Υ_t, resampling directly from the probability density $\hat{\tau}(\boldsymbol{q}_{1:t})$ of the posterior or even better from the true target $\tau(\boldsymbol{q}_{1:t})$ would give a higher diversity among particles without losing focus on high probability areas. A classic method to estimate a probability density based on a set of (weighted) samples is the *kernel density estimator* (KDE). The KDE uses a kernel function K, such that $\int K(u)du = 1$, placed at each particle $\{W_t^i, \boldsymbol{X}_t^i\}$ with a specified bandwidth $h_1 \dots h_d \in \mathbb{R}^+$, one for each dimension d of the state space, to approximate the target. In context of time-sequential filtering the kernel estimator is given as

$$\hat{\tau}(\boldsymbol{q}_t) = \frac{1}{\sum_{k=1}^{N} W_t^k} \sum_{i=1}^{N} \frac{W_t^i}{h_1 \dots h_d} \left[\prod_{j=1}^{d} K\left(\frac{q_{j,t} - X_{j,t}^i}{h_j} \right) \right] \quad , \tag{6.9}$$

where the state $\boldsymbol{q}_t$ represents the point to be estimated and the index j access the scalar elements of the state and particle vectors. While the choice of kernel is typically not a major factor, since the kernel estimate inherits the properties of the chosen kernel, the bandwidth h plays a critical role in determining the quality of the estimate. Therefore, it is very common to choose a Gaussian kernel for simplicity:

$$K_G(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2} \right) \quad . \tag{6.10}$$

The above (exact) KDE approach is very easy to implement and offers great flexibility. However, the complexity is $\mathcal{O}(NM)$ and thus the computation time depends on the number of particles N and the number of points M. To speed up the computation it is often recommended to reduce N, e.g. by eliminating duplicates. This inevitably has a negative impact on the accuracy of the estimation. For this reason, several approaches to reduce the complexity of KDE have been proposed in the literature [BP11; GM03; GS91; O'B16]. A very popular and extensively studied approach, e.g. by [FM94], [Wan94] or [HW96], is the *binned KDE* (BKDE), originally proposed by [Sil82]. Similar to KLD-sampling, it uses a piecewise discrete distribution with binning. Here, the goal is to reduce N by considering only the bins, rather than all particles within the set.

The bins can be implemented as an equidistant grid with spacing/granularity g_s, where each bin represents the count of the particle set at grid points $\boldsymbol{v}_j$ (vertex), indexed by $j \in \mathbb{Z}$. Depending on the dimensionality and data structures, $\boldsymbol{v}_j$ can either be represented as scalar, a multivariate vector or n-tuple. To map the particles onto the grid, a binning rule $r_j(\boldsymbol{X}_t^i, \delta)$ is applied. There are several different ways to define such a rule, the most basic is the simple

binning rule

$$r_j(x,\delta) = \begin{cases} W_t^i & \text{if } x \in [(j-\frac{1}{2})g_s, (j+\frac{1}{2})g_s] \\ 0 & \text{otherwise} \end{cases} , \tag{6.11}$$

and the linear binning rule

$$r_j(x,\delta) = \begin{cases} W_t^i(1-|g_s^{-1}x-j|) & \text{if } |g_s^{-1}x-j| \leq 1 \\ 0 & \text{otherwise} \end{cases} . \tag{6.12}$$

for the one-dimensional case, where x denotes a sample and we define a grid point as $v_j = jg_s$. Extending the above to multivariate data is straightforward since they scale with dimensionality [Wan94], but is omitted here for simplicity. The linear binning rule in (6.12) divides the particles into two fractional weights shared by the nearest grid points. Both rules have been well studied in terms of their impact on the approximation error, with linear binning often providing higher accuracies [HW96]. They can be computed in $\mathcal{O}(N)$, since simple binning is essentially the quotient of an integer division, and the fractional weights of linear binning are given by the remainder of the division [Bul18]. Having a spatial model like the one introduced in Section 4 allows to change the upper case of the piecewise function of the simple binning rule (6.11) to $\boldsymbol{\rho}_t \in \boldsymbol{v}_j$, where $\boldsymbol{v}_j \in V$ and $G = (V, E)$ is the graph structure of the mesh or grid and $\boldsymbol{\rho}_t$ is the particle position, cf. Section 4.2.

Given the above, the BKDE $\hat{\tau}(\boldsymbol{u})$ computed pointwise at a grid point $\boldsymbol{u} = (u_1, \ldots, u_d)$ can be denoted as

$$\hat{\tau}(\boldsymbol{u}) = \frac{1}{\sum_{i=1}^N W_t^i} \sum_{j=1}^{|V|} \frac{C_j}{h_1 \ldots h_d} \left[\prod_{k=1}^d K\left(\frac{u_k - v_{j,k}}{h_k} \right) \right] , \tag{6.13}$$

where $|V|$ is the number of grid points. The count C_j at grid point $\boldsymbol{v}_j$ is calculated according to

$$C_j = \sum_{i=1}^N r_j(\boldsymbol{X}_t^i, \delta) \tag{6.14}$$

such that $\sum_{j=1}^{|V|} C_j = \sum_{i=1}^N W_t^i$ [HW96]. By comparing (6.9) and (6.13) it can be seen that the binning process introduces an approximation error, which can be controlled mainly by the number of grid points V and therefore the granularity g_s of the grid. Thus, the coarser the grid, the less computational effort is required to evaluate the kernels, but the greater the approximation error. Unfortunately, there is no standard recipe for which g_s offers the best compromise between speed and accuracy, as this depends strongly on the structure of the unknown posterior distribution and the size of the particle set [HW96].

The main advantage of (6.13) comes from the underlying equidistant grid. This allows for advanced data structures and eliminates duplicate computations for identical kernel evaluations. It also enables more advanced operations, such as using an FFT algorithm to efficiently compute the estimate in $\mathcal{O}(|V|\log|V|)$ [Sil82]. Based on this concept, an even faster approach than BKDE was presented by the authors of [Bul18]. They interpret the computation of the density estimation as a signal filtering problem, which allows further approximation schemes for the Gaussian kernel, namely iterated box filters. This not only leads to a simplification, since (6.13) can be reduced to a single convolution per dimension without the need to compute the exponential function, but also allows for straightforward parallel computation. They have shown that due to the linear complexity $\mathcal{O}(|V|)$ of the box filter, their approach, called boxKDE, outperforms the BKDE in terms of computational time while having a similar approximation error. The boxKDE is so fast that it can even be used for real-time use cases, e.g. directly on a smartphone for self-localization.

In *kernel density resampling* (KDR) one of the fast computational methods presented above is used to resample particles according to $\boldsymbol{X}_t^i \sim \tau(\boldsymbol{q}_{1:t})$. Unlike KLD-resampling, this approximates in a continues function, allowing new particles to be drawn at arbitrary positions within the state space. To prevent sampling in non-walkable areas not covered by the spatial model, e.g. between walls or outside the building, a validation check must be included for each newly drawn particle. If the particle has a position matching the spatial model, keep it; if not, draw another particle to keep the number of particles constant over time. Due to its continues properties, KDR allows for resolving a situation like Figure 6.1 by resampling particles at time t also in adjacent areas, ignoring the spatial boundaries. The number of these particles depends on the chosen bandwith h and the thickness of the wall. Unlike the resampling schemes presented in Section 2.3.2, drawing new particles from $\tau(\boldsymbol{q}_{1:t})$ instead of the particle set Υ_t generally slows down simple impoverishment. If the diversity of the particles is low, e.g. a compact particle set with many duplicates, drawing from Υ_t will further decrease the diversity by discarding low weight particles and increasing the number of duplicates of high weight particles. By using KDR, all particles are redrawn and no particle is retained or duplicated. This always results in an asymptotically optimal representation of $\tau(\boldsymbol{q}_{1:t})$.

Despite the good properties of KDE in resolving impoverishment caused by spatial restrictions, this introduces drawbacks in situation where such restrictions come in handy, e.g. in buildings with many small rooms such as office spaces. Here, KDR with a large bandwidth h leads to a higher susceptibility to false measurements by absolute sensors such as Wi-Fi or BLE. In areas where there should be no particles based on the movement pattern of the pedestrian and the spatial restrictions, they may be present due to KDR and may have a high weight due to

false measurements. Basically, this means that the movement model and the resampling model work against each other.

6.2 Data-driven Approaches for Localization

In this section, we discuss approaches to PDO that are explicitly designed for target tracking and localization. They are data-driven in the sense that we use context and/or sensor knowledge to solve the problem of impoverishment. The following approaches claim to have at least some degree of general applicability. This means that they work independent of the building and at least partially independent of special sensor models. Again, our restrictive movement model and the associated impoverishment problem play an important role. In the literature, such PDO approaches are referred to in different ways, e.g. the authors of [Li14] simply call them intelligent methods.

6.2.1 Randomized Filter Extensions

The most straightforward approach for tackling the problem of impoverishment is by sampling a few particles (e.g. $1\,\%$) randomly within the state space, either in the sampling or resampling step of the particle filter. By having a spatial model, this can be restricted to walkable areas of the building. With a certain probability, depending on the size of the building and the number of these random drawn particles, a region that does not have any representation by particles at time $t-1$, will have one at t. Especially in situation, where the filter (particle set) completely lost track of the true posterior this comes in handy.

Imagine a situation where a floor change using a staircase from the first to the second floor could not be detected due to incorrect barometer readings. All particles are on the first floor because the barometer does not report the floor change. Therefore, the particles bypass the staircase and move further into the building, where there are no other stairs. The problem: The restrictive movement model only allows for floor changes at staircases or elevators. Therefore, the filter loses track because it is on the wrong floor and has no chance to change it. This impoverishment situation can be resolved by randomly sampling a few or even a single particle on the correct floor level near the mode of the true posterior report of the sensor models. This leads to a high weight of these particles computed by the state evaluation, and the resampling step will duplicate them, so that over time the filter will recover and slowly converge.

We used this approach for our ILS at the EvAAL offline indoor location competition [TS17]. This was one of the main reasons why we were able to win the competition, as the use of the elevator in particular was a major drawback for many other systems that relied on particle

filters, as the modulation of elevators in movement models is often very difficult and is therefore modeled by simple vertical connections in the spatial model. As a result, particles were often too slow or too fast to correctly model the elevator ride and ended up on the wrong floor. However, the random factor can cause unpredictable behavior and may not always work when needed, e.g. an erroneous Wi-Fi measurement that predicts a completely unrealistic position would normally be ignored because there may be no particles in that region, but with a randomly drawn particle there, this could cause large approximation errors.

Another approach, especially for situations in which the particle set is trapped inside a room, is adding a slight chance (e.g. $0.5\,\%$) for particles to walk through walls. This can be easily implemented as an extension to the transition strategies of the navigation mesh presented in Section 4.3.2. As before, this makes the system more susceptible to false measurements. Especially in buildings with many small rooms, one quickly loses the advantages of the movement model and thus accepts a generally higher estimation error / worse position estimation for the purpose of correcting a possible impoverishment. Finally, despite the simplicity of randomized approaches, they should be omitted as they often cause multimodal scenarios and headaches when evaluating strange behavior in experiments [Fet18].

6.2.2 Support Filter Approach using IMMPF

As mentioned before, sample degeneracy and impoverishment are a pair of contradictions that can be described as a trade-off between the need for diversity and the need for focus. An approach presented in our paper [Fet17] to overcome these contradictions is to combine two corresponding particle filters using IMMPF (cf. Chapter 5). First, we define the so-called dominant filter, which is an ILS (cf. Section 2.5) with a restrictive movement model (cf. Chapter 4) and is configured to achieve the highest possible accuracy in position estimation. It satisfies the requirements for a high focus on high probable areas of the posterior, but is thus highly susceptible to sample impoverishment. The second filter is called the support filter and is used with a very simple movement model with no environmental constraints and the same absolute sensor model, e.g. BLE or Wi-Fi, as the dominant one. In the simplest case, the model can be a multivariate Gaussian distribution that simulates the motion of a particle based on a random direction and distance. This is basically a model as defined in (4.9), but without a spatial model. This means that the support filter provides high diversity with a very robust but uncertain/not very accurate position estimate. It can recover very quickly from erroneous measurements, and getting stuck is a very unrealistic scenario. Both filters are based on the CONDENSATION principle, as shown in Algorithm 5.

The idea of the *support filter approach* (SFA) is now obvious. While the task of the dominant filter is to provide the most accurate position estimate, the support filter should allow as little impoverishment as possible. The IMMPF then combines these filters using the non-trivial Markov switching process. The unique feature is that only the estimate of the dominant filter is used for position estimation in ILS, and the support filter is used only to resolve possible impoverishment. This can be achieved by comparing the posteriors of the two presented filters to get an idea of how much they differ from each other. If the estimation results are good, the similarity should be high, since both filters use an identical state estimation procedure. Conversely, if the filters are not similar to some degree, it is very likely that one of the filters is blocked or has lost track due to sample impoverishment or degeneracy.

A measure of the similarity of two probability densities was introduced with the Kullback-Leibler divergence $D_{\mathrm{KL}}(P\|Q)$ in (6.1). Assuming P to be the support filter and Q the dominant filter, both distributions are represented by a disjoint set of particles and thus require sampling a density at an arbitrary position. For this, we can use a KDE of the form (6.9) with a Gaussian kernel as presented in Section 6.1.2. As mentioned in [Fet17], if the computational power and/or the number of particles is too high to use a KDE efficiently, a simpler and faster method is to assume a multivariate Gaussian distribution for the complete posterior. Here, the mean of the Gaussian is given by the weighted arithmetic mean of the particles and the variance is defined by the sample covariance matrix. Of course, this is a very simplified representation of the distribution, but it should be sufficient for measuring the divergence in our context.

The above behavior of mutual support between filters is achieved in the mixing stage of the IMMPF, as derived in Section 5.1.2. That is, if we recognize that the dominant filter has diverged from the supporting filter and is likely stuck or lost, then particles from the supporting filter are more likely to be sampled while mixing the new set of particles for the dominant filter. To do this, we define a quality metric for the dominant filter based on the Kullback-Leibler divergence D_{KL} using an exponential function such as

$$q_{\mathrm{KL}}(D_{\mathrm{KL}}) = e^{-\lambda D_{\mathrm{KL}}} \quad . \tag{6.15}$$

Within the Markov transition matrix Π_t, this gives the probability of the dominant filter drawing particles from its own. Consequently, drawing particles from the support is then given by $1 - q_{\mathrm{KL}}(D_{\mathrm{KL}})$. We have chosen λ heuristically, as it is highly dependent on the noise of the sensor models. A reasonable range for most applications is somewhere between 0.01 and 0.10.

In Figure 6.2 this process is illustrated for the impoverishment scenario shown in Figure 6.1. Each plot shows a different timestamp t within the filtering process. At the beginning, at $t = 25$ in Figure 6.2a, the dominant (orange) and supporting (blue) filter perform quite similarly,

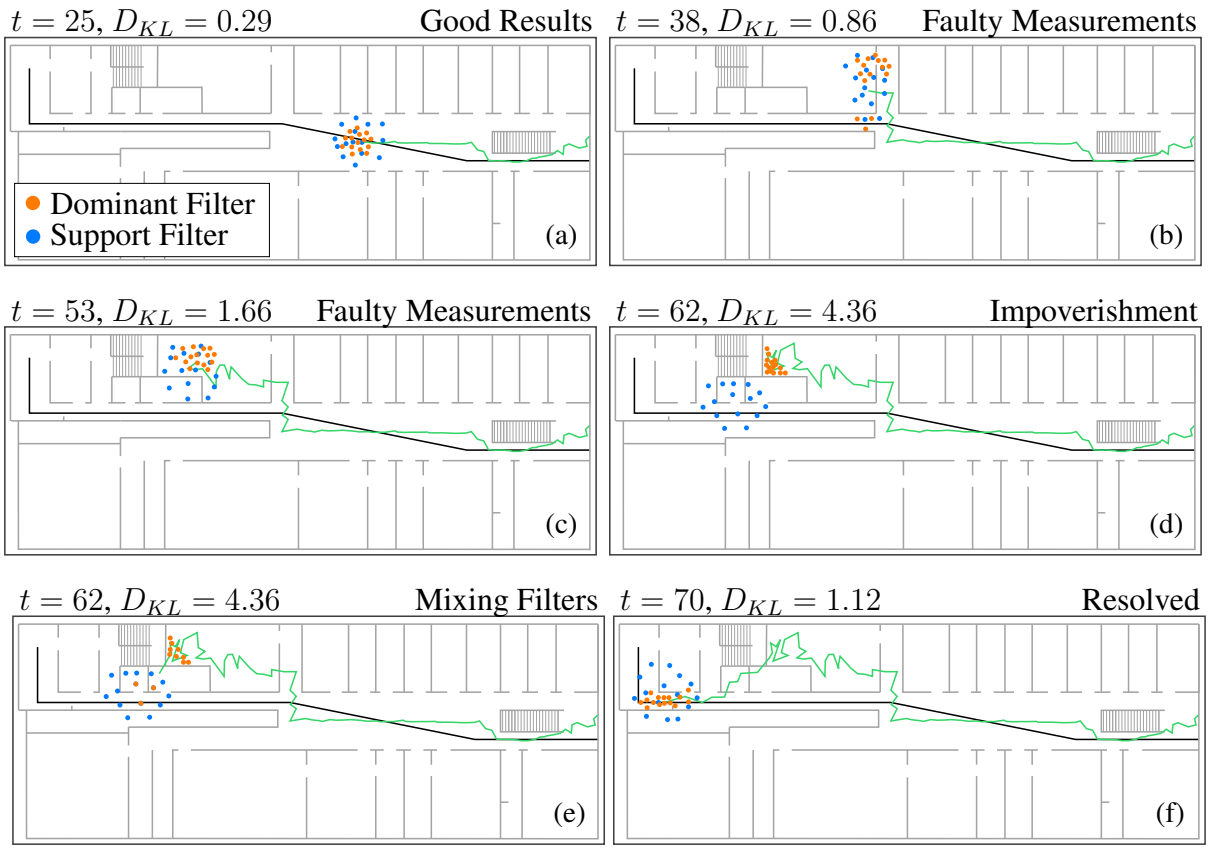


Figure 6.2: Solving the impoverishment scenario illustrated in Figure 6.1 using the *support filter approach* (SFA). Each plot shows a different timestamp t of the filter and provides the current Kullback-Leibler divergence D_{KL} between the support (blue) and dominant (orange) filters. In (a) both filters are still quite similar and thus D_{KL} is low. Due to some erroneous measurements, both filters are dragged into the upper space. While the supporting filter is able to recover in (c) and (d) when the measurements are good again, the dominant filter remains stuck due to the restrictive movement model. This causes an increasing D_{KL} and thus a mixing between both filters in (e), which then leads to a full recovery of the dominant filter in (f).

resulting in a small D_{KL}. It can be seen that the dominant filter has more focus compared to the support filter which has more scattered particles. In Figure 6.2b some erroneous measurements occur, dragging both filters into the upper part of the floorplan. As discussed at the beginning of this chapter, the dominant filter, which implements our ILS, is unable to recover as the particle set is moved further into the room in Figure 6.2c. In Figure 6.2d this results in the dominant filter getting stuck due to the restrictive movement model. This leads to an increasing D_{KL} as both filters start to drift apart and thus a large exchange of particles from the supporting filter into the dominant filter. This can be seen in Figure 6.2e. Finally, after some more timesteps, this leads to a full recovery in Figure 6.2f.

The quality metric $q_{\text{KL}}(D_{\text{KL}})$ in (6.15) is only reliable if the quality of the support filter remains high. Since we assume a very simple movement model, the quality depends mainly

on the sensor models used and thus on whether the measurement noise is within reasonable limits. This means that not only the dominant filter can deviate from the support filter due to impoverishment, but also the support filter can deviate from the dominant filter due to erroneous measurements. Consider Wi-Fi RSSI as sensor model for making inferences about a pedestrians position. As discussed in Section 3.1 RSSI readings are prone to a multitude of different effects such as attenuation or multipath. Especially in challenging environments without a good coverage and materials such as steel-enforced concrete this can lead to bad RSSI readings and thus causing bad estimation results for the support filter. Here, the dominant filter benefits from its high focus based on the more advanced movement model and should ultimately provide a better approximation of the posterior. Therefore, the quality of the support filter is given by the used sensor models, in the above example the Wi-Fi quality $q_{\text{wifi}}(\boldsymbol{o}_t)$ as introduced in (5.11).

Finally, the transition matrix $\Pi_{\text{KL+wifi}}$ of the IMMPF can be denoted as

$$\Pi_{\text{KL+wifi}} = \begin{pmatrix} q_{\text{KL}}(D_{\text{KL}}) & 1 - q_{\text{KL}}(D_{\text{KL}}) \\ 1 - q_{\text{wifi}}(\boldsymbol{o}_t) & q_{\text{wifi}}(\boldsymbol{o}_t) \end{pmatrix} . \tag{6.16}$$

It controls the behavior of mutual support between the dominant and the supporting filter described above. If the dominant filter suffers from sample impoverishment, e.g. by getting stuck or losing track, the probability $[\Pi_{\text{KL+wifi}}]_{12} = 1 - q_{\text{KL}}(D_{\text{KL}})$ increases with diverging support filters, and thus the number of particles drawn by the dominant filter from the support filter increases by $[\Pi_{\text{KL+wifi}}]_{12} \cdot 100\,\%$ [Fet17]. As shown again in (6.16), the modularity of the IMMPF and a dynamic transition matrix allow almost any combination of filter implementations. Depending on the use case, the supporting filter can be adapted without affecting the ILS implemented by the dominant filter, and vice versa. Of course, the complexity of this approach is higher compared to the general approaches presented so far, not only because a supporting filter has to be implemented according to the scenario, but also because the required computational capacity increases depending on the complexity and the number of particles used.

6.2.3 Uniform distributional Evaluation Resampling

The last proposed approach for data-driven PDOs is a compromise between the very simple but efficient randomized approach of Section 6.2.1 and the precise but complex SFA approach of the previous section. Instead of introducing a separate support filter using IMMPF, we extend the resampling step (cf. Section 2.3.2) and rather than randomly sampling a certain percentage of particles in the building, as suggested in Section 6.2.1, we sample from the evaluation density $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ of the particle filter. To sample from $p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$ we do not use an approximation based on the current particle set $\Upsilon_t = \{W^i_{1:t}, \boldsymbol{X}^i_{1:t}\}^N_{i=1}$ of the posterior, but by a static particle

set $\Upsilon_{\text{static}} = \{W^j, \boldsymbol{X}^j\}_{j=1}^{M}$ uniformly distributed in the building. Static in this context means that this particle set is not subject to any movement model, but all M particles have a constant position that never changes, similar to the nodes in the spatial grid. The concept is also similar to the occupancy grid [TBF05]. The static particle set can be generated in several ways: In a grid it can simply be the nodes or centers of the cells, in a triangulated mesh the triangles can be drawn cumulatively according to their area and the particles are uniformly distributed within the triangle using barycentric coordinates.

Impoverishment cannot occur because each region of the building is covered uniformly, and thus the state space is completely covered by static particles under asymptotic conditions. The evaluation density is therefore approximated by the static particle set according to the basic concepts of Monte Carlo simulation. When resampling, we now simulate the mixing of the IMMPF by drawing a small number of particles (e.g., $0.1\,\%$ of N) multinomially from the static particle set Υ_{static} in addition to the filter's posterior particle set Υ_t, and thus include them in the resampled particle set $\dot{\Upsilon}_t$.

In summary, the so-called *uniform distributional evaluation* (UDE)-resampling introduces a second set of particles Υ_{static}, which is static and uniformly distributed throughout the building. For each resampling step, the particles of Υ_{static} are weighted according to $W^j \propto p(\boldsymbol{o}_t \mid \boldsymbol{q}_t)$. Then a small number $\hbar_{\text{UDE}}$ of particles are sampled according to $[\frac{1}{N}, \boldsymbol{X}_t^i] \sim \Upsilon_{\text{static}}$, while the majority, $N - \hbar_{\text{UDE}}$, are sampled according to $[\frac{1}{N}, \boldsymbol{X}_t^i] \sim \Upsilon_t$, combined in the new set of particles $\dot{\Upsilon}_t$, which is used for the next filter update. Only the resampling step uses Υ_{static}, so other filter processes like state estimation do not require any modifications.

The advantages of UDE-resampling are obvious. The implementation of this approach is trivial, and arbitrary resampling methods can be easily extended, since only a small number of particles need to be drawn from another set using the same sampling principle (cf. Figure 2.5). This hardly increases the complexity of the computation, only the reweighting of all static particles by the sensor models included in the evaluation has to be taken into account. A disadvantage of random-based approaches is their susceptibility to measurement errors. While the support filter focuses on relevant areas within the building due to the simplified movement model, this information is completely missing here and may well lead to unpredictable behavior, e.g. unwanted short jumps across the building [Fet22].

Chapter 7

Experiments

As we have seen, a hybrid ILS consists of many different sensor fusion components. As a basis for position estimation and further inference, we agreed on the broad class of particle filters in Chapter 2. Although the introduced process is clearly defined with the steps of transition, evaluation, and resampling, we encountered a variety of different methods for the actual implementation. Due to the modular properties of the generic filter framework presented in Section 2.5, the possible combinations of these methods are manifold and subject to hardly any restrictions. Depending on the scenario, the sensor technology used, or the available computing power, the ”recip“ for the ILS may change. Despite changing the combination of sensor models, which is quite obvious, the estimation or resampling strategy might be different just as adding measures against impoverishment or degeneracy might be necessary or not.

In this context, it is important to know under which conditions the respective methods behave and how. Therefore, the goal of the experiments is to investigate the proposed methods in more detail. The focus is on a purposeful discussion and on giving an impression of what improvements, but also what deteriorations, can be expected. The specific accuracies of the individual sensor models, i.e. whether e.g. FTM provides a lower positioning error than Wi-Fi RSSI, are already frequently discussed in the literature and are therefore not necessarily the focus here. Corresponding literature references can be found in the ”Limitations and Discussion“ chapters of the respective sensor models in Chapter 3.

This chapter starts with a detailed description of the experimental setup. In addition to the hardware used, the test environment and the method for calculating the position error, this includes a presentation of the software tools used, since most of them are available as free software [Bul20a; sim; Ste]. Subsequently, the most relevant methods and models from Section 3 and Section 4 are evaluated and compared. Individual sensor models that are primarily concerned with position estimation performance, such as the data-driven model of the Wi-Fi FTM,

will be considered in some detail as part of the multi modal fusion in the IMMPF, but the focus will be more on improving the methods and the underlying particle filter than on the physical properties of the sensor. For the same reason, the experiments will conclude with the evaluation of methods to solve the much discussed problem of sample impoverishment.

7.1 Experimental Setup

All of the experiments shown here were conducted in real-world environments, namely an office building, a museum, and an industrial hall. The scenario has been carefully crafted for maximum realism. This means that the tests were conducted in all buildings during opening hours, i.e. when people other than the test subjects were present. Sources of interference such as other radio receivers or transmitters, e.g. smartphones or TVs, were not explicitly turned off, but merely filtered out using a whitelist of known receivers. Effects such as overlapping, spontaneous NLOS conditions caused by groups of people or crowded Wi-Fi channels can occur at any time. In addition, economic factors were also considered in the experimental considerations and setups. Besides the number of access points to be used, these include the time required to install and calibrate the ILS in the building. In particular, the optimization model proposed in Section 3.1 brings massive time savings compared to a classical fingerprinting approach, as we will see.

Of course, this form of data acquisition is not without its critics and raises the question of reproducibility. However, monitoring factors such as sensor noise, physical properties like the influence of the weather on a barometer, or human movement patterns are difficult. Measuring these would require an inconceivable amount of effort and would make the evaluation of big buildings and complex paths almost impossible. Many mathematical publications in the field of target tracking therefore often use synthetic data to show individual effects such as convergence behavior or general behavior over time. We will also refer to such data at some point, but they can rarely simulate the extreme complexity and interaction in a real-world environment. The focus is therefore always on conditions that are as close to reality as possible.

In our experience with real-world applications for smartphone-based self-localization, the only sources of information are the smartphone's built-in sensor and a blueprint or architectural drawing of the building provided by the customer. Therefore, good reproducibility can be achieved by providing a data set that includes as many sensor readings as possible from the smartphone, a floorplan of the building including access point locations, ground truth and other relevant information, and a video recording of the subject walking along the test path.

There are several public data sets in the literature that attempt to meet these requirements. One of the most comprehensive is RISEdb with 4 different buildings ranging from about $200\,\mathrm{m}^2$

to about $2500\,\mathrm{m}^2$ [SB21]. The measurements are collected with a special system worn on the back. This includes visual sensors such as an RGB camera, a stereo camera and two LiDAR sensors, as well as a standard smartphone. The focus of the data set is therefore primarily on indoor visual localization and less on our use case of a handheld smartphone for pedestrian self-localization, due to the fact that the smartphone is rigidly worn on the back.

More promising data sets are Fusion-DHL [Her21] and IPIN-2021 [TS21]. The latter is the result of the International Conference on Positioning and Navigation (IPIN) competition held since 2011, in which we participated in 2016 and 2023, winning each time the smartphone based track [Eva]. Unfortunately, for legal reasons, only the 2021 data is publicly available. It involved a university building where 16 different test tracks were recorded using one of 5 different handheld smartphones. Data is collected from all sensors available in the smartphone, specifically IMU, BLE, and Wi-Fi RSSI. In the case of Fusion-DHL, the data set consists of 3 different shopping malls with a total running time of $5.7\,\mathrm{h}$ and a covered distance of $16.2\,\mathrm{km}$. Special emphasis is put on recording IMU data, so unfortunately there are no records of BLE and Wi-Fi RSSI, only Google Fused Location Provider (FLP) is used to obtain a geolocation via Google's proprietary Play Services.

The common problem with both data sets is the lack of environmental information. The floorplan is only provided as a raster image, which seems reasonable at least for Fusion-DHL's proposed approach of a CNN for trajectory optimization. However, it does not provide much in the way of more detailed information, which would be highly relevant for a detailed discussion of the behavior of the ILS in difficult situations, such as the transition between two building areas with different wall materials. There is no information about open windows, closed doors, wall thicknesses, fire doors and so on. Since there are no videos of the respective trajectories, the evaluation results are difficult to understand and the data sets are primarily used to compare the accuracy of ILS with limited contextual knowledge. Finally, none of the data sets proposed above provide test paths in multi-story buildings with floor changes, which is one of the most complex and error-prone scenarios.

7.1.1 Setup, Tools and Data Recording

For this reason, we have recorded a large number of our own data sets that meet the conditions defined above and developed a set of software tools and file formats for standardization purposes. Explicit attention has been paid to difficult situations that can demonstrate the possibilities of the methods proposed in this work. To better understand how this data is collected, we'll go through each step in detail:

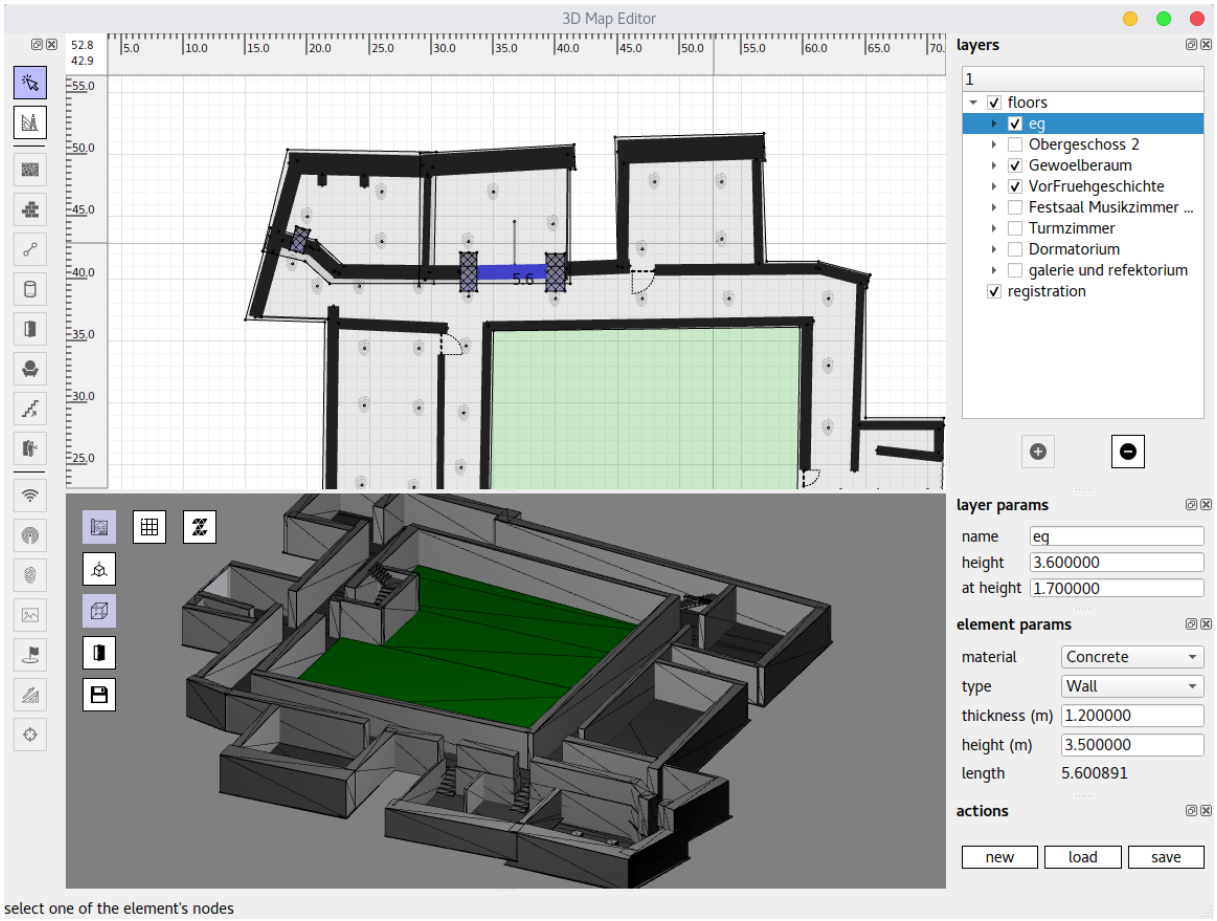


Figure 7.1: The 3D map editor for creating floorplans in simpleLoc XML format. The window is divided into toolbar (left), layers (top right), parameters of the current selection (bottom right), drawing mode (top center) and 3D view (bottom center). Working similarly to Inkscape or FreeCAD, lines and polygons are drawn and parameters such as material or thickness are added accordingly.

1. **Creating the floorplan.** Since we are not surveyors, we rely heavily on blueprints or architectural drawings of the building in which the data is to be collected. In modern buildings, these are often provided in some digital format from a CAD or GIS application. In pre-computer age buildings, however, they are often just an outdated scan as a raster image. For reasons of time and competence, we do not carry out a new topographic survey, but examine the existing scans for obvious errors or perspective distortions and correct these in the floorplan to be created. We use off-the-shelf laser distance meters to do this, and we check site conditions such as wall thicknesses or whether undocumented construction activities have taken place, such as the removal or addition of drywall. The simpleLoc XML format for the floorplan shown in Listing 4.1 is generated using our own

3D map editor (see Figure 4) by manually transferring all the information. The map editor is similar to applications such as Inkscape or FreeCAD.

2. **Installing the radio infrastructure.** Wi-Fi infrastructure is available in most buildings today. However, it is often not designed for good location performance, but only for the best possible WLAN coverage. As a result, areas such as stairwells, corridors and entrance areas are often poorly covered. We document the existing infrastructure with position, MAC address and if necessary UUID in the floorplan and can also add our own Wi-Fi beacons based on ESP32-DevKitC V4 microcontrollers at suitable locations, e.g. complete dead spots. These beacons will then be flagged so that they can be filtered as desired in the experiments to further evaluate the specific conditions on site.

   Since only a few buildings have a BLE infrastructure, we also supplement this with beacons based on the ESP32-DevKitC V4 microcontroller. These are mounted at a height of more than $2\,\mathrm{m}$ on walls and ceilings at a distance of about $10\,\mathrm{m}$ from each other. The beacons we use require a permanent power source, which is why their placement usually depends on the location of power outlets, and thus an even distribution in the building is rarely achieved. This often leads to suboptimal conditions with many NLOS situations, which we believe is a realistic scenario. The same approach is used for Wi-Fi FTM enabled beacons based on the ESP32-S3-DevKitC-1. As mentioned in Section 3.2, an accurate measurement of the beacon's position in the building is of great importance. They are also included in the floorplan with MAC address, UUID, position and other useful meta information. In this way, a whitelist can be created for the Wi-Fi access points of the existing infrastructure as well as for all temporarily installed beacons in order to exclude irrelevant radio transmitters.

3. **Recording reference measurements.** For many radio-based sensor models for localization, reference measurements at known positions within the building are of great importance. While classical fingerprinting benefits from a high density of reference measurements, e.g. in a $1\,\mathrm{m}$ grid, for the optimization proposed in Section 3.1, a few but well-placed reference measurements in areas with rather high pedestrian traffic are sufficient.

   We use a self developed open source mobile application for Android devices (see Figure 7.2a) [Bul23]. It shows the previously created floorplan with the positions for the reference measurements, which are also created in the map editor. By tapping on a reference, the recording can be started for an individually defined period of time, we assume $30\,\mathrm{s}$ per reference. We record all radio measurements received by Android's Bluetooth

(a) Reference application (b) Recording application

Figure 7.2: Our publicly available and open source mobile applications are used to record the data sets on Android devices. The reference app in (a) is used to record radio measurements at predefined positions in the floorplan [Bul23]. This screenshot shows the recording dialog. The purpose of the recording application shown in (b) is to create test walks [Bul20a]. The app allows free selection of which sensor of the smartphone should be recorded. Additionally, the passing of a ground truth marker as well as the current activity can be recorded via buttons.

scan and RTT API. These are Wi-Fi and BLE RSSI and Wi-Fi FTM distances. The frequency here varies depending on the smartphone used, but everything is configured to always receive the highest frequency possible. The reference measurements are stored with identifier, timestamp, position and sensor recordings.

4. **Set ground truth for test walks.** To verify the accuracy and characteristics of an ILS, test walks are recorded. Some publications work with test points, i.e. the accuracy at certain points in the building over a certain period of time. However, we believe that this type of accuracy evaluation is not very useful for the self-localization use case, since people are moving in most scenarios, e.g., navigating in an airport or path analysis. We usually choose the trajectory of a test walk to be as realistic as possible for the given scenario,

```
0;-2;Fri Sep 01 15:35:20 GMT+02:00 2023;Frank;THWS
<!-- Timestamp;EventId;Args... -->
0;50;STANDING;1
51272330;0;-0.30166942;4.3526587;8.734048
51272330;3;0.064983115;-0.23649593;0.16192514
56328330;1;-0.112231635;0.048184782;9.805753
56328330;2;-0.23463176;4.2904096;-1.3455414
61368330;0;-0.22505496;4.4101195;8.121132
61368330;3;0.06604841;-0.21732058;0.14807628
66408330;12;0.19772942;-0.16232795;-0.6445509;0.72045046
66408330;18;0.2370983;-0.09607227;-0.40076905
66408330;2;0.0047884034;-0.35913026;-0.62249243
69503530;50;WALKING;0
...
```

Listing 7.1: Short sequence showing the csv-like format used for test walk recordings.

allowing for some randomness such as quick 180 degree turns, sitting or standing for long periods of time. A scenario is often dictated by the purpose of the building. In a museum, people tend to walk short distances between exhibits, which also means that they spend a certain amount of time standing at the exhibit. This results in entangled paths that do not necessarily cover a very long distance, but are traveled over a longer period of time. In a large office building, on the other hand, longer distances are often walked from one area to another in the most optimal and quickest way possible.

In the building, we mark the path of a walk with numbered markers. Each walk has its own color and identifier. Small symbols on the markers indicate the direction of the path to the next marker and, if necessary, give instructions on how long to stay at this point or give hints on further activities, such as using the elevator. The positions of these markers are measured again and added to the floorplan.

5. **Collecting the sensor measurements.** The final step is to create recordings of the test walks by as many different subjects with as many different smartphones as possible. For this we use another self-developed open source recording application [Bul20a]. The so-called SensorReadoutApp is able to receive readings from most sensors supported by the Android SDK 33, including all sensors presented in this thesis. For convenience, it allows free selection of the sensor to be read. We use a csv-like file format where each incoming measurement is mapped to its corresponding event identifier with the current timestamp in nanoseconds and the measured values. Listing 7.1 shows a short sequence of a recording as an example.

   In addition to recording raw sensor data, the application provides functionality to manually add the events of changing activity, e.g. when walking on the first step of a staircase,

the activity changes to ascending staircase, and when a ground truth point has been passed to the recording file. This is done by simply clicking the buttons shown in Figure 7.2b. Of course, this could introduce human error, especially if the timing of clicking the button and creating the manual event is off. For this we often use an instructor who records the manual events on a second device while giving the necessary instructions to the tester. If the tester makes a mistake, an event is wrongly marked, or the delay between marking and making the event is greater than $500\,\mathrm{ms}$, the recording is discarded. The subject's and instructor's recordings are then merged to synchronize the timestamps.

To map the recording to one of the predefined test walks, it is necessary to add a timestamp to the file whenever a ground truth marker is passed. A constant speed of movement is then assumed between two consecutive markers, allowing a position and timestamp to be interpolated. Thus, the ground truth may not be $100\,\%$ accurate, but it is fair enough for error measurements. The approximation error is then calculated by comparing the interpolated ground truth position with the current estimate. An estimate on the wrong floor has a large impact on the pedestrian's positional awareness, but provides a relatively small error. Therefore, errors in the z-direction are penalized by tripling the z-value.

To summarize, a data set of a single test environment consists of a floorplan that includes the location of reference positions, ground truth markers, access points, beacons, and spatial information such as walls, stairs, elevators, windows, doors, or other obstacles. It also includes a file for each available radio technology, e.g. Wi-Fi RSSI, BLE RSSI and Wi-Fi FTM, containing measurements taken at the reference positions and other supplementary material such as the original architectural drawings, a readme.txt file with a description of the test conditions and other nice to know things that happen on the side. Most importantly, there is a recording for each successful walk performed by a tester, given in the file format shown in Listing 7.1. In most cases, we define 3 different walks through the entire building, with the tester both familiar and unfamiliar with the building, and using a set of different smartphones, but only a single device per walk. Depending on the duration of the walk, the sampling frequency of the sensors, and the number of sensors in the smartphone used, the file size can reach several hundred megabytes. To get a basic understanding of the conditions, each test walk comes with at least one video showing a tester walking along the path. At this time, our data sets are not publicly available because we have not been able to obtain permission from some of the building owners. A solution is being worked on.

Vendor	Model	Year	Android	IMU	Bar	Wi-Fi	BLE	FTM	SHL	ROT	IPA
Samsung	Galaxy S3 mini	2012	5.1.1	✓	-	✓	✓	-	✓	-	-
Samsung	Galaxy Note II	2012	7.1.1	✓	✓	✓	✓	-	-	✓	-
Google	Nexus 6	2014	7.1.1	✓	✓	✓	✓	-	✓	✓	✓
Samsung	Galaxy S5 Neo	2015	6.0.1	✓	-	✓	✓	-	✓	-	-
Google	Pixel 1 XL	2016	10.0.0	✓	✓	✓	✓	✓	✓	✓	✓
Google	Pixel 2 XL	2017	11.0.0	✓	✓	✓	✓	✓	✓	-	✓
Google	Pixel 3a	2019	12.1.0	✓	✓	✓	✓	✓	✓	-	✓
Google	Pixel 5	2020	13.0.0	✓	✓	✓	✓	✓	✓	-	-
Samsung	Galaxy S21	2021	13.0.0	✓	✓	✓	✓	✓	✓	-	✓

Table 7.1: Smartphones used to record the data sets. The table provides an overview of the device name, the year of manufacture, the version of the Android operating system used, the sensors included in the smartphone: IMU (accelerometer, gyroscope, magnetometer), barometer, Wi-Fi, BLE, or FTM and finally the test environment in which the device was used.

7.1.2 Test Environments and Devices

The experiments were mainly conducted in three different buildings: SHL, ROT and IPA. There is a temporal discrepancy in the records, as they were recorded for different studies and publications during the course of the dissertation. For example, the data from the SHL are from 2017, 2021 and 2022, from the Museum from 2018 and from the IPA from 2022 and 2023. For this reason, the wireless technologies installed, the age of the smartphones used, and possibly also differences in the recorded data, such as missing activity in individual recordings, differ to some extent. An overview of which smartphones were used in which test environments is shown in Table 7.1.

In the following, the buildings are briefly presented based on their infrastructure, age, materials used, building layout, and other special features.

SHL The abbreviation SHL stands for the address Sanderheinrichsleitenweg 20, where in 2011 the modern campus of the Technical University of Applied Sciences Würzburg-Schweinfurt was opened. As shown in Figure 7.3, it consists of two buildings connected by a large courtyard and an underground passage. The upper of the two buildings is the faculty building, which has four floors with a footprint of $77\,\mathrm{m} \times 50\,\mathrm{m}$ and mainly provides office space for the university's professors and staff, as well as some seminar rooms and laboratories. The lower building, with a floor area of $110\,\mathrm{m} \times 60\,\mathrm{m}$, houses the lecture halls and a cafeteria. The entire complex comprises approximately $12\,000\,\mathrm{m}^2$ of walkable area with outdoor facilities and $8500\,\mathrm{m}^2$ of non-walkable area. Beneath the campus is an underground parking garage and several basements, which are not used and therefore not modeled. With few exceptions, the exterior facade is made of metallized glass panels and the windows are aluminum. Inside, the floor slabs, staircases, and struc-

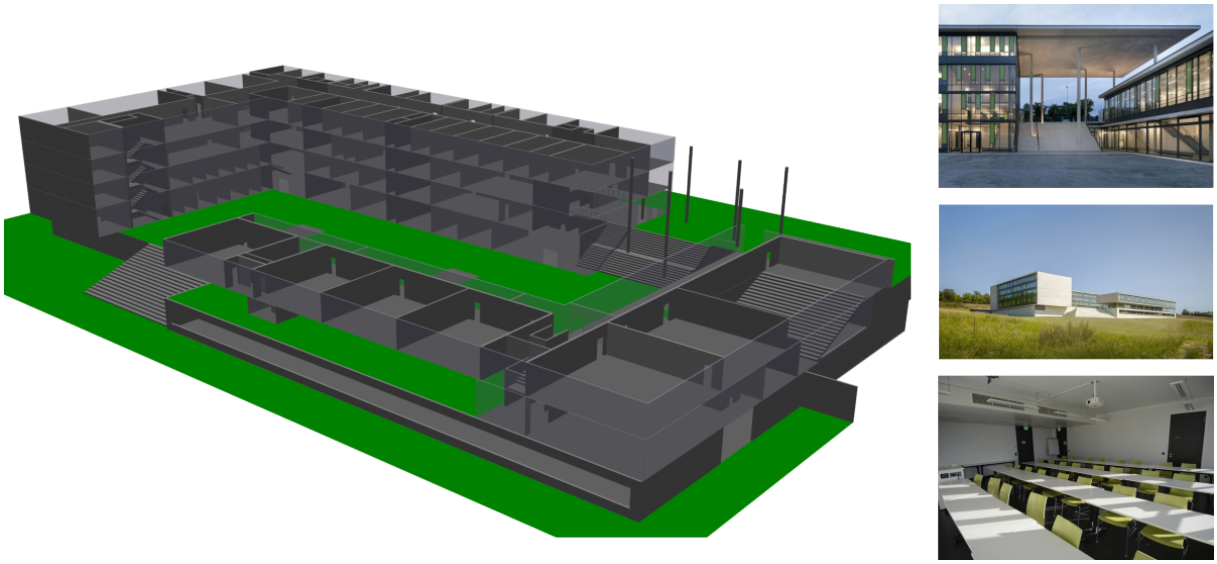


Figure 7.3: The SHL campus of the Technical University of Applied Sciences Würzburg-Schweinfurt. It has two buildings, one for faculty (top left) and one for lectures (bottom right), and a total accessible area of about $12\,000\,\mathrm{m}^2$. The facade is made of metallized glass panels and the interior is a mixture of reinforced concrete and drywall.

tural columns are reinforced concrete. The interior walls are predominantly drywall, with the exception of some fire doors and utility shafts, which are clad in aluminum. There are 34 access points distributed throughout the campus, using both $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$. Their locations cannot be published for legal reasons and are therefore not shown in the graphs. For some experiments, we therefore use temporarily installed beacons (BLE, Wi-Fi and FTM), also based on ESP32 microcontrollers.

ROT Built in the 13th century as convent, the historic building is used since 1936 as museum, called RothenburgMuseum, by the medieval city of Rothenburg ob der Tauber [Möh]. Over the centuries, the building has undergone several reconstructions from different architectural periods. On the first floor there are massive stone walls up to $2\,\mathrm{m}$ thick, as the building is directly connected to the town wall. Above and below, there have been several additions and alterations over the years, resulting in a rather chaotic picture. In total, there are 8 different units that are accessed by several staircases. The units vary in shape, size and altitude as can be seen in Figure 7.4. In particular, the upper floors, built in a timber and clay construction, are very winding and have a rather unusual visitor routing. The building has approximately $2000\,\mathrm{m}^2$ of accessible interior space, including a patio, measuring $70\,\mathrm{m} \times 50\,\mathrm{m}$. The museum has several thousand exhibits distributed in glass cabinets throughout the space. All in all, this is a particularly difficult scenario, both for the radio technology due to the many different materials used, and for the move-

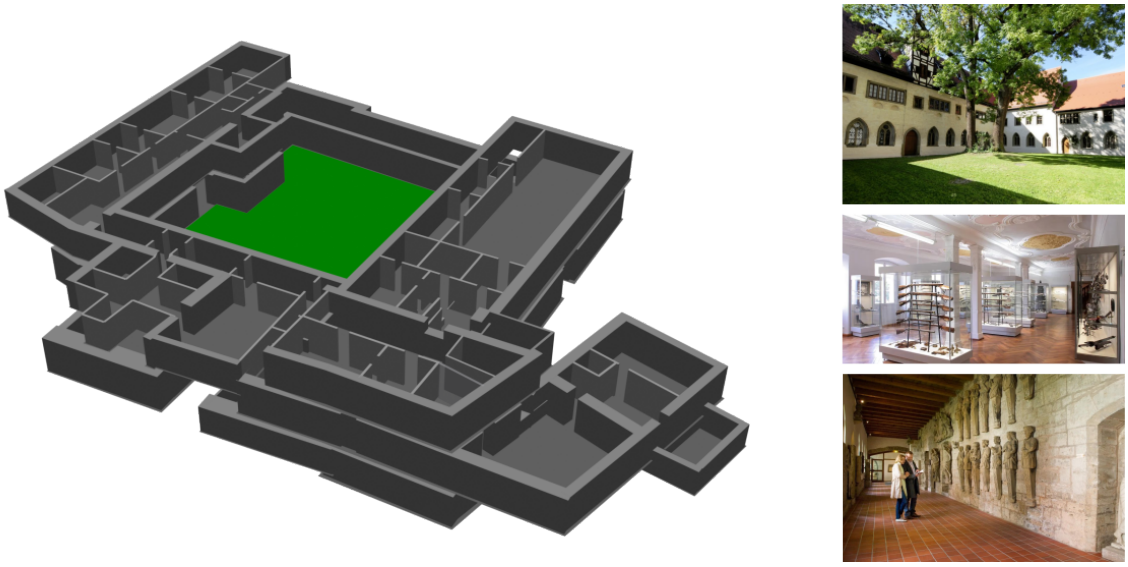


Figure 7.4: The building of the RothenburgMuseum in the medieval town of Rothenburg ob der Tauber was built in the 13th century and has a size of $70\,\mathrm{m} \times 50\,\mathrm{m}$ with $2000\,\mathrm{m}^2$ walkable space. The building has undergone several reconstructions from different architectural periods and thus consists of 8 different units that are accessed by several staircases. It is made of thick stone walls on the ground floor and wood framing on the upper floors.

ment model due to many nooks and crannies and half-height floors. Strangely enough, the last survey and thus the building plan dates back to the 1980s in paper form. Another peculiarity is that the building has no Wi-Fi infrastructure, so we installed 42 WEMOS D1 mini (ESP8266) microcontroller as Wi-Fi beacons. The placement of the beacons was chosen based on the available electrical outlets, with the requirement that at least 3 beacons could be detected by a Wi-Fi scan within a radius of $10\,\mathrm{m}$. The outlets are located at heights between $0.2\,\mathrm{m}$ and $2.5\,\mathrm{m}$.

IPA The former car workshop with salesroom and offices is now used by the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) for the project group KI-noW - Artificial Intelligence for Sustainable Optimized Value Creation. As can be seen in Figure 7.5, the building is divided into two main areas on one level, the office area with glass facade (left) and a workshop with brick walls and metal facade and ceiling (right). Overall, the areas have a lot of open space as well as some smaller rooms for offices, kitchen and storage. The workshop consists of a variety of electrical, toolmaking, and metalworking workstations. Of particular note is the CNC machine and rack storage room in the left corner of the workshop. The footprint of the entire building is $38\,\mathrm{m} \times 15\,\mathrm{m}$ with a ceiling height of $4\,\mathrm{m}$ in the offices and $6\,\mathrm{m}$ in the workshop. Although the IPA is the smallest building in terms of area, the industrial scenario is particularly interesting be-

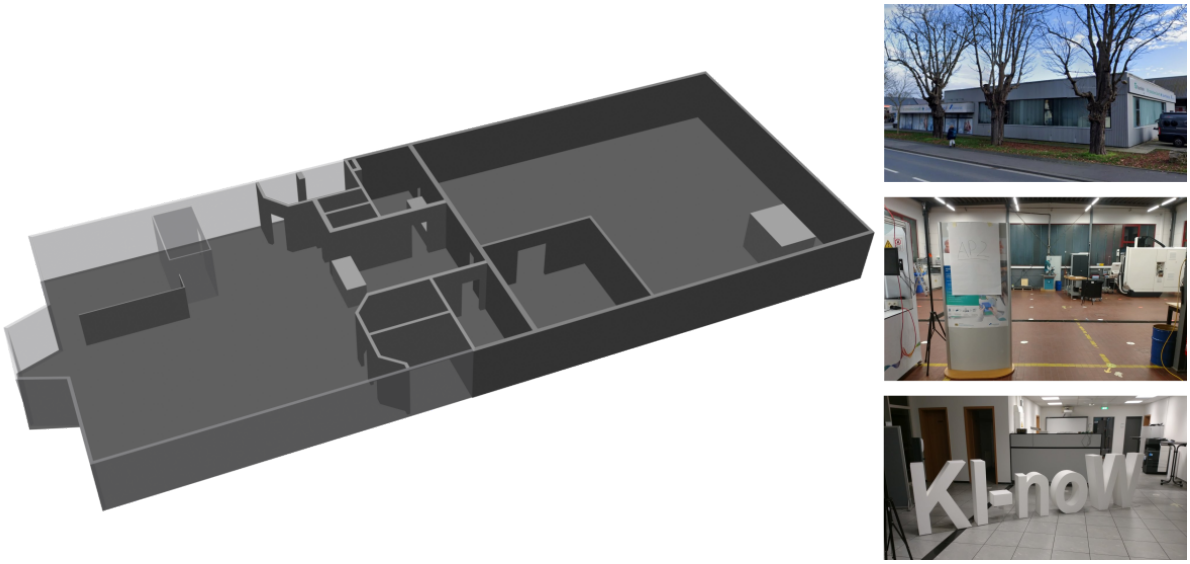
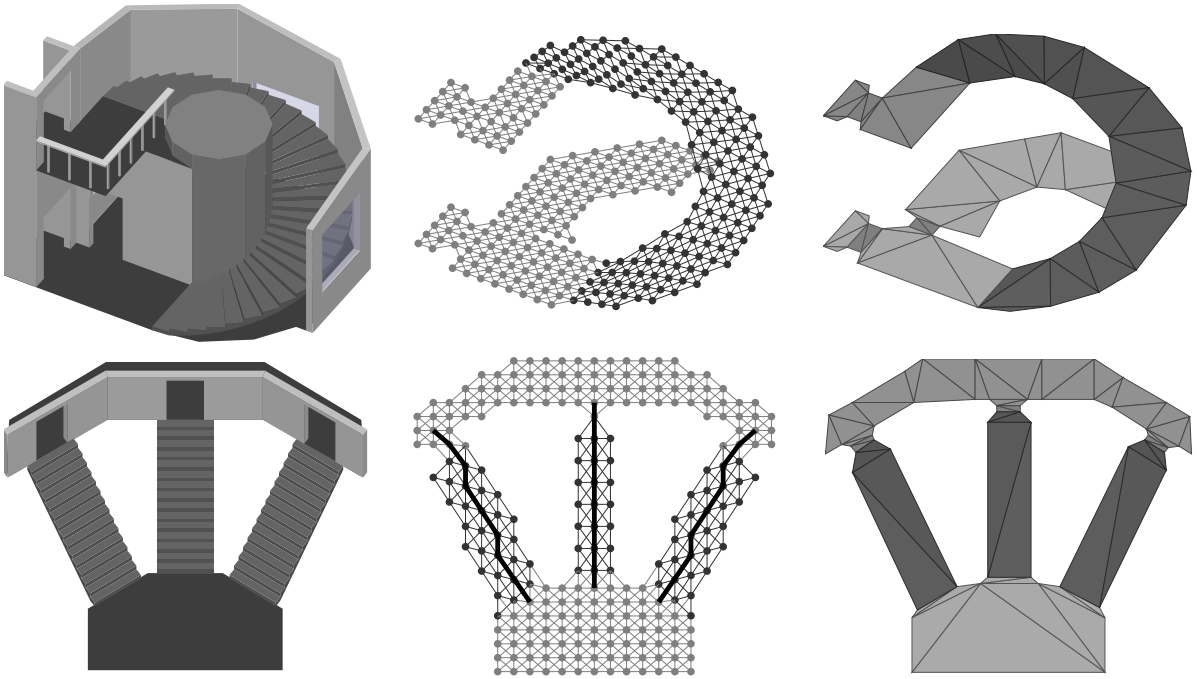


Figure 7.5: A former car workshop, now used by the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) as an industrial research and test environment. On one level and with a size of $38\,\mathrm{m} \times 15\,\mathrm{m}$, the building is divided into an office area with glass facade (left) and a workshop with brick walls and metal facade and ceiling (right).

cause during the recording, work was carried out on various machines and workstations. The testers performed the activities of the workers, so the smartphones were either in their trouser pockets or in a bag attached to their hips. To provide a BLE and Wi-Fi FTM infrastructure, 22 ESP32-S3-DevKitC-1 microcontrollers were installed as beacons.

7.2 Indoor Localization

In what follows, we evaluate the most relevant of the approaches proposed in Chapter 3 and Chapter 4 using the above experimental setup. In doing so, we reverse the order somewhat to follow the logical procedure for building a dataset as shown in Section 7.1.1. Thus, we start with a comparison of the spatial models and the corresponding movement models based on a given floorplan. Then, we investigate how the parameter optimization from Section 3.1 behaves under the extremely difficult conditions of the ROT building. Finally, an ILS will be built step by step with the help of sensor fusion, which can serve as a basis for all further experiments and discussions. This section concludes with an examination and comparison of the most popular estimation strategies, namely maximum particle, weighted average, and KDE, to produce a final position estimate.



(a) 3D floorplan (b) grid graph (c) navigation mesh

Figure 7.6: Comparison of the spatial representation between grid graph ($g_s = 0.4\,\mathrm{m}$, middle) and navigation mesh (right) on two challenging scenarios (left). The limitation of the graph to multiples of $45°$ is especially visible at the edges of the stairs. Figure is based on [Ebn21].

7.2.1 Comparison of Spatial Models

The choice of a spatial model is one of the most important decisions for an indoor localization system, as discussed in detail in 4.1. The implementation of the movement model, the characteristics of the sensor models, the use within a software application, and many other factors are affected or directly related to it. Therefore, the two types proposed in this paper, namely the grid graph and the navigation mesh, will be compared in the following.

First, the most obvious difference is the spatial representation. Figure 4.1 already compares the two spatial models for the first floor of the ROT building. Despite the coarse granularity of $g_s = 0.9\,\mathrm{m}$, the areas of the rooms are well represented by the 1700 cell grid (see Figure 4.1b). However, the doors are very poorly represented and often have only one connecting edge. Lower granularity solves this problem, but at the cost of memory. The irregular navigation mesh requires only 170 triangles to accurately represent the represented area of the building. In narrow areas, such as doorways, and in complex locations, such as corners, the number of triangles inevitably increases (see Figure 4.1c).

Both effects are illustrated in Figure 7.6 for two more complex architectural examples. Here the grid is created according to Section 4.2 with $g_s = 0.4\,\mathrm{m}$ and the navigation mesh as explained in Section 4.3. At the top you see a spiral staircase with three doors. Grid and mesh are both able to represent this scenario in a valid way. They differ mainly at the edges of the staircase, where the discreteness of the mesh to multiples of $45°$ is clearly visible. Since the steps of the staircase are relatively wide, an average of 4 nodes form a row of the grid, allowing enough degrees of freedom to model the actual path of the pedestrian over the edges. Because the topological description of the floorplan (see Algorithm 4.1) does not contain any information about the number or shape of the steps, they are not modeled. Instead, the meta-information about which vertex or triangle is a stair is used to adjust the step length μ_{step} accordingly. Consequently, the granularity of the mesh should not be too small in order to perform a suitable simulation as described in 4.2.3 even for small step lengths. If g_s is larger than the maximum step length d_{walk}, in the worst case the particles will not reach the next vertex cell and will remain in their current cell, simulating a standstill. Due to the actual continuous movement in the navigation mesh, this behavior cannot occur here.

The second scenario at the bottom of Figure 7.6 illustrates three stairs with different orientates leading the a door. Walking alongside the middle one results in a straight line, because of its $90°$ alignment. This might be a realistic path a pedestrian would walk and thus simulating the movement accuratly. On the others stairs this is different, as the orientation of the stairs does not match the $45°$ alignment of the edges and thus results in a more unrealistic zig-zag pattern, yielding a non-ideal path. Using a smaller granularity might improve approximating the ideal path, but this is obviously a general problem of the grid graph. Another thing to note here is that in the staircase in the middle, the walkway (door) between the stairs and the upper room is only connected by a single vertex. This leads to a high concentration of particles when passing through. In cases where the particle set is close to the true posterior, this can be beneficial, as it further improves the compactness, but in opposite situations it increases the risk of impoverishment, as the variance of the particles, i.e. the representation of the state space by particles, is low.

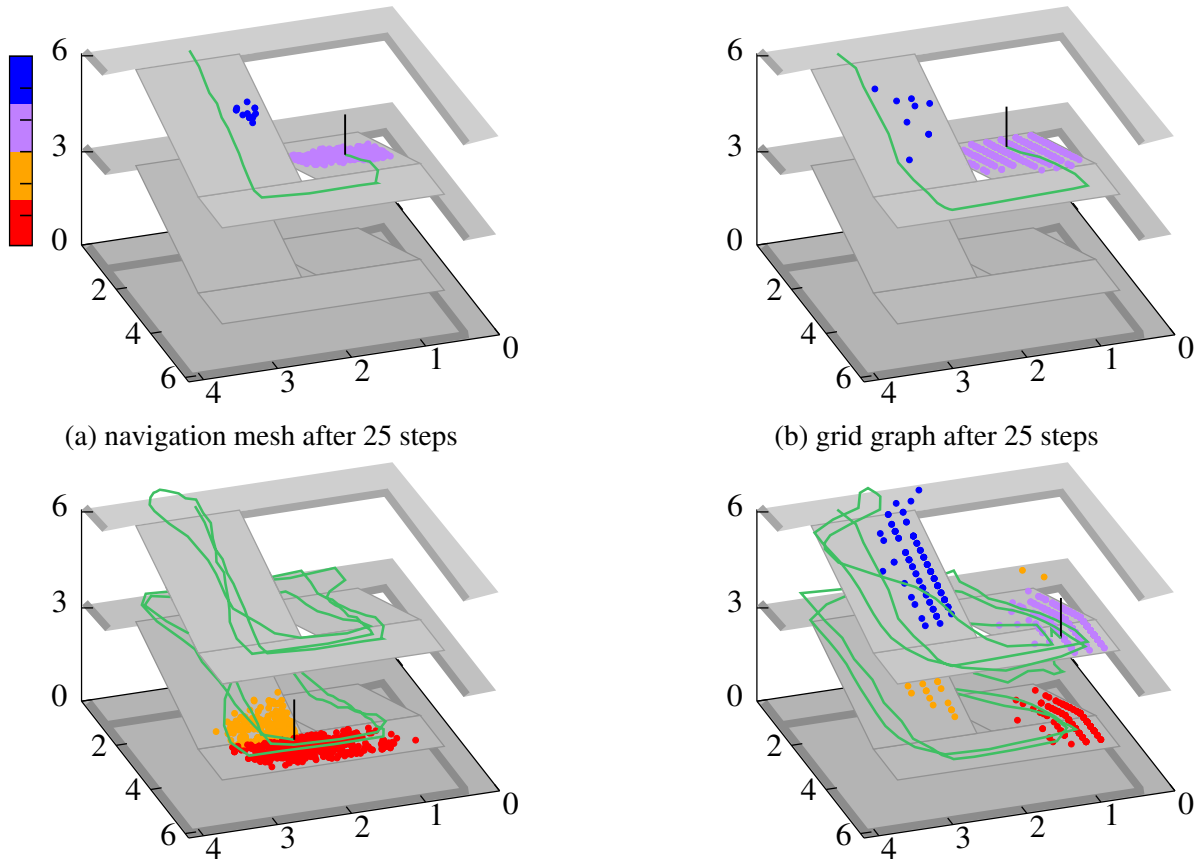
From a privacy perspective, a user's position information is extremely sensitive data, so there is a fundamental requirement that all calculations can be performed on the smartphone without interaction with an external server. But even without this privacy aspect, this requirement makes sense in order to avoid dependencies on data transmission rates or protocols and thus achieve the lowest possible latency between estimation and the actual position of the pedestrian. This also includes the storage of the spatial model in a suitable data structure on the device. An important aspect to consider in practice is memory consumption. As already explained in the text below Figure 4.1b in Section 4.1, the size of a regular tessellated spatial model such as the grid depends

mainly on the resolution of the cells and thus grows quadratically in two dimensions depending on the size and number of floors of a building. If a cell of a vertex overlaps with a wall or another obstacles, it is simply removed. On the other hand, the navigation mesh is an irregular modal and thus the number of requires triangles depends mainly on the architecture of the building, e.g. a large open space can be modeled with only two large triangles, while a structure like the spiral staircase in Figure 7.6 requires many smaller triangles.

The effect of different architectural designs can be seen by comparing SHL and ROT. Since $12\,000\,\mathrm{m}^2$ is four times the size of ROT, the number of triangles for SHL is 4164 compared to 857. That is $\approx 2.9\,\mathrm{m}^2$ per triangle for SHL and $\approx 4.7\,\mathrm{m}^2$ for ROT. This is due to the many small office rooms combined with open pillars throughout the SHL space, dividing larger triangles into several smaller ones. In contrast, the ROT building, although an old historic building with nooks and crannies, has many right angles, large open areas, and a clear, linear structure. Considering a data structure that requires $56\,\mathrm{B}$ per triangle, storing the vertex position, its neighbors, and meta-information about what type it is, e.g. whether it is a stair or an outdoor area, the memory required to store the mesh for SHL is $248\,\mathrm{kB}$ and $51\,\mathrm{kB}$ for ROT. Considering a granularity $g_s = 20\,\mathrm{cm}$, the grid requires $518\,\mathrm{k}$ vertices for SHL and $83\,\mathrm{k}$ for ROT. Using a similar data structure of $56\,\mathrm{B}$ for every vertex, this results in $30\,\mathrm{MB}$ and $5\,\mathrm{MB}$, respectively. As expected, the ratio between the building size and the number of vertices, i.e. the resulting file size, remains around 1 to 6. While the memory requirement of the grid is a multiple of that of the mesh, it can still be handled by any modern smartphone. However, adding more meta-information to each vertex, such as shortest path estimates [Ebn16] or Wi-Fi RSSI fingerprints [Bul22], can quickly exceed the manageable memory size. In an application with many different buildings in a large area such as a city, this requires some intelligent memory management.

7.2.2 Transition Models

For each of the spatial models used in this work, transition models have been proposed to simulate the movement of pedestrians within the particle filter, see Section 4.2.3 for the grid and Section 4.3.2 for the mesh. In the following experiment, these models will also be discussed and compared. We use the ILS proposed in Section 2.5, which is based on the CONDENSATION particle filter. In order to observe the pure behavior of the transition (step and heading), the evaluation step and the resampling step (see Algorithm 5) are not performed. The a priori distribution of the filter is set to a known fixed position so that each of the 5000 particles used for the MC simulation starts at the same position with the same heading. The number of particles was determined heuristically from empirical values under the condition that the computation can always be performed on a standard smartphone. For each newly detected step, a filter update



(a) navigation mesh after 25 steps

(b) grid graph after 25 steps

(c) navigation mesh after 180 steps

(d) grid graph after 180 steps

Figure 7.7: Staircase in the SHL building, where subjects walked up and down several times. The state of the filter is shown after 25 (top row) and 180 steps (bottom row), with the estimated trajectory (green), the current estimated position (black), and particles colored according to their z coordinate. While the mesh still gives good results after 180 steps, the grid suffers from multimodal density. All units are in meters. Figure is based on our findings from [Fet18].

$t \mapsto t+1$ is performed. For the variance/noise of the sensor models, we choose the parameters $\sigma^2_{\text{step}} = 0.1$ for the step length (see (3.20)) and $\sigma^2_{\text{turn}} = 0.1$ for the heading (see (3.25)). As discussed in Section 4.3.2, different strategies can be chosen to deal with illegal movements of a particle, e.g. caused by hitting a wall. For this experiment, we choose a simple but efficient strategy: every time a particle makes an illegal move, it is removed and the last correct move of the previous particle is duplicated. For the grid, we configure a granularity of $g_s = 20\,\text{cm}$; as we know, a special strategy is not necessary here, since particles only move between vertices on the edges.

The experimental setup is shown in Figure 7.7. In this simple scenario, subjects walk up and down a staircase in the SHL building several times using a Google Nexus 6. Subjects were instructed to position themselves as centrally as possible and to maintain a moderate walking

speed, at best the same in both directions. The estimated trajectory is shown in green and was obtained by weighted average estimation. The color of the particles is based on their z-coordinate to help the viewer distinguish between the different floors. Looking first at the snapshot after 25 steps in Figure 7.7a for the mesh and Figure 7.7b for the grid, the results for the particle distribution and the estimated trajectory are very similar. Already after this small number of steps, the first outliers (blue particles) appear in both cases. Due to the noise modeled in the sensor models, they cover too short a path several times, so that the second left curve leads upwards instead of downwards. The second part of the staircase, where the violet particles are located, is thus missed. A closer look also reveals that the violet particles in Figure 7.7b scatter farther away from each other due to the grid structure than the particles in the mesh, which are much more compact due to the continuous structure. The spread of both particle sets towards the edges of the staircase is explained by the relatively strong modeled noise of $\sigma^2_{\text{turn}} = 0.1$, which corresponds to just under $5°$.

After 180 steps, the outlier behavior of the grid has multiplied. As can be seen in Figure 7.7d, individual particle groups now lie on all areas of the staircase, implying a multimodal representation of the posterior density. This renders the position estimate useless, as it is only somewhere in the middle of the staircase due to the weighted averaging. Only by adding a state evaluation and an associated weighting of the respective particles could this situation be remedied, e.g. activity detection (ascending, descending). The mesh, on the other hand, has an overall larger variance of the particle set than before, which is understandable due to the increased uncertainty after so many time steps, but remains unimodal (see Figure 7.7c). Due to the strategy of preventing illegal movements, outliers can only occur for a short time, since they are removed as soon as they hit a wall. As a result, even after 180 steps, a good position estimate for the mesh is still obtained, independent of other sensors. This is a very interesting approach for a pure PDR system.

Applying such a strategy as used for the mesh to the grid is conceivable, but it has some drawbacks. As we have shown in [Ebn16], with an inverted graph it is possible to find the vertices that are in direct neighborhood of a wall. Since movement is still only possible on the graph, one strategy would be to give lower weights to particles moving directly towards or along a wall in order to remove them after some time steps in the resampling. Especially for small rooms, narrow corridors or coarse granularity, this will inevitably lead to a reduction of the degrees of freedom and thus to a less accurate position estimation. Another problem that can occur is the reduction of different particles, i.e., sample impoverishment, since, for example, when passing through a door, only those particles that take a path centered on the door remain highly weighted. This ultimately leads to many particles sharing the same position and thus a poor representation of the state space. With meshing, this problem is much less

pronounced, since the combination of distance to walk d_{walk} (cf. (4.5) and Θ_t (cf. (4.6)) also allows a destination for the movement simulation (transition) through walls (cf. (4.9)). This ultimately leads to a higher versatility of the particles in such a situation.

Considering these results and the comparison of the spatial models in Section 7.2.1, it can be concluded that the mesh is preferable to the grid due to its more flexible structures, lower memory requirements, and more accurate motion model. However, there are two disadvantages that need to be considered. First, as mentioned above, meta-information can be stored on the grid without much effort, whereas the mesh requires the cumbersome way of barycentric coordinates and correspondingly complex data structures. The second disadvantage is the increased computational effort caused by the computation of the destination position, i.e. the query whether the position computed according to (4.9) is part of the submesh (cf. Figure 4.5). Using the grid graph, only the direct neighbors are of interest. Therefore, the movement model can be implemented very efficiently using a tree structure. This becomes interesting when an ILS is to be displayed on low-power devices, e.g. in the embedded domain, or when a large number ($\gg 50\,\mathrm{k}$) of particles are to be used on a smartphone. More in-depth experiments on how both spatial models work for navigation can be found in [Ebn21].

7.2.3 Parameter Optimization

As we discussed in Section 3.1, measuring the transmitter parameters $(\boldsymbol{\varrho}, P_0, \gamma, \beta)$ required by a radio propagation model, e.g. (3.2) or (3.3), is a tedious task and more or less impossible without knowing anything about the building. For this reason we have introduced a parameter optimization in Section 3.1.5 with (3.6). In the following, we will discuss and compare the global localization method, which uses all references throughout the building, and a local method, which provides a set of optimized parameters of all transmitters per floor, using only references associated with the corresponding floor. For the evaluation we use the data set recorded at ROT. We use a whitelist filter that only considers the Wi-Fi beacons we have installed, ignoring other transmitters such as a visitor with an open hotspot on his smartphone, to reduce possible sources of error.

Figure 7.8 shows the test setup and optimization results for the ground floor of the ROT building. The first step was to install the 42 Wi-Fi beacons (black dots) at available power outlets throughout the building. We tried to place at least 2 per room or no more than $15\,\mathrm{m}$ apart. Then, 133 reference points (fingerprints) were placed $3\,\mathrm{m}$ to $7\,\mathrm{m}$ apart to achieve the most complete coverage of the entire building with corresponding reference measurements. A detailed analysis of how the number of reference points affects the optimization can be found in our paper [Ebn17]. There it was shown that increasing the number of reference points improves

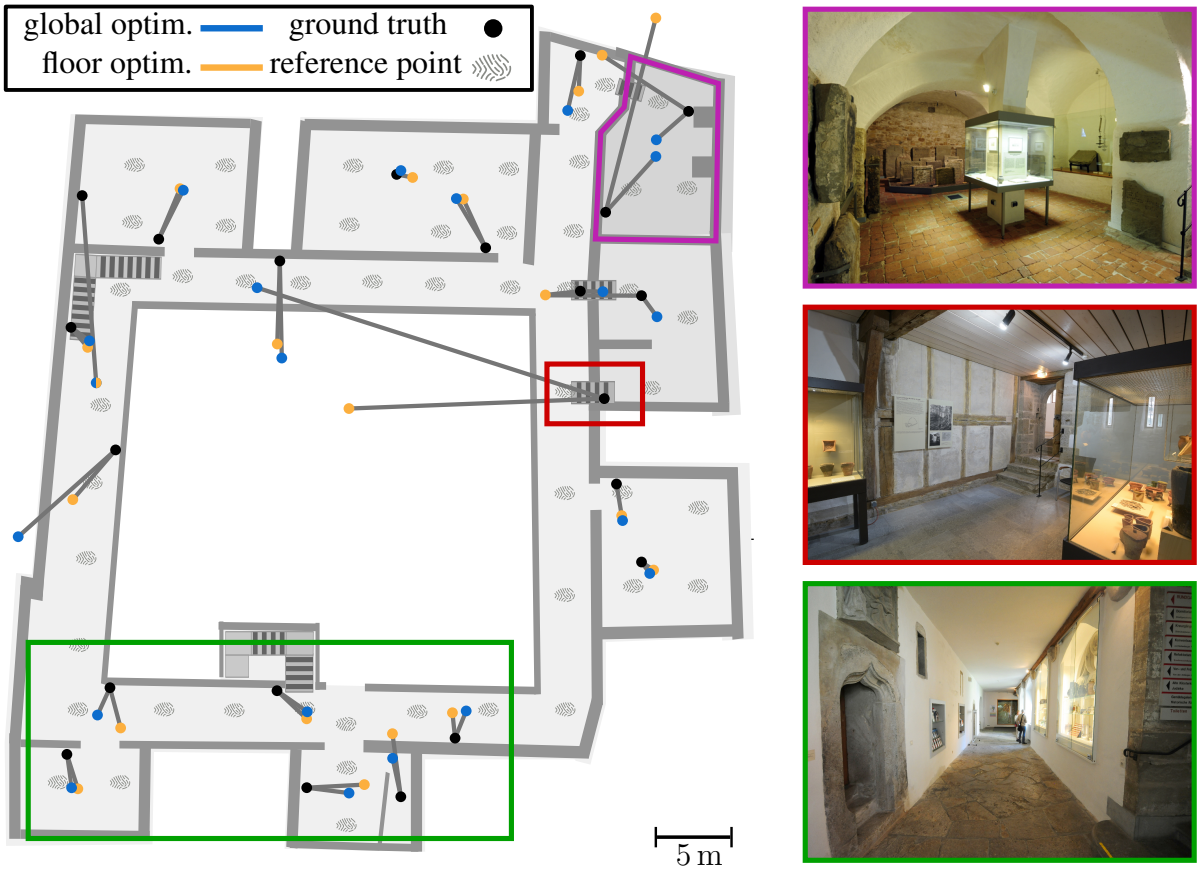


Figure 7.8: Floorplan of the ground floor of the ROT building [Fet18]. It shows the spatial arrangement of the reference points, the ground truth positions, and the optimized locations of the Wi-Fi beacons. The gray lines show the connections between each beacon and its corresponding optimization. The colored borders delineate regions of particular importance, which are further examined in the accompanying text. Corresponding images on the right side of the figure show the museum in these specific locations, providing visual context for the discussion.

the results only up to a certain threshold. Based on these results, a threshold was chosen for this evaluation that represents a trade-off between reasonable recording time of less then one workday and optimal accuracy. In addition to the number of points, the time to measure all references also depends on the duration of a single recording. To be independent of a smartphone's sample rate, 30 complete Wi-Fi scans were always recorded per reference point.

Using a Motorola Nexus 6, which operates exclusively in the 2.4 GHz band, approximately 1000 RSSI measurements per reference point were recorded over an average duration of 25 s. These were then grouped per physical transmitter and the average signal strength at that point was calculated. As ground truth, the exact locations of the Wi-Fi transmitters were accurately determined by laser scanning. This information allows a meaningful comparison of the optimized beacon positions with the actual ground truth position. Figure 7.8 illustrates the results

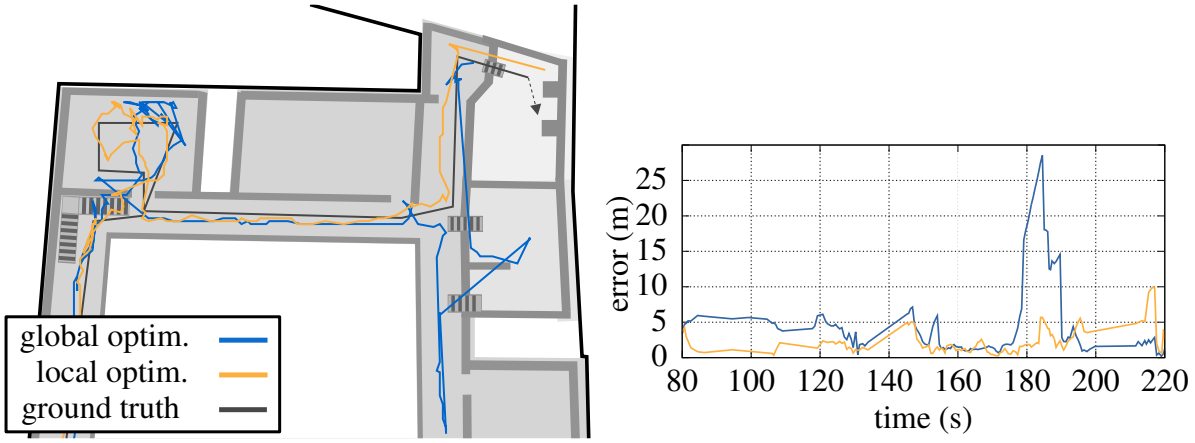
obtained by two different approaches: the global (blue) and the per-floor (orange) method for all Wi-Fi beacons installed on the ground level. The optimized positions $\boldsymbol{\varrho}$ are connected to their corresponding ground truth by gray lines, showing the positional error in the xy-plane.

Considering the average distance error in three dimensions (3D) between the actual positions of the beacons and the optimized ones, we observe a discrepancy of $5.4\,\mathrm{m}$ ($\sigma^2 = 5.1$) for the per-floor and $4.8\,\mathrm{m}$ ($\sigma^2 = 5.6$) for the global strategy. It's worth noting that the results show variation across different areas of the building. Areas highlighted in green have a better accuracy, as they contain a higher concentration of transmitters with clear line-of-sight conditions. In contrast, areas highlighted in red and purple are characterized by the presence of thick stone walls that prevent direct line-of-sight. The maximum position error occurs in the red rectangle, reaching $25.3\,\mathrm{m}$ for the global scheme and $18.4\,\mathrm{m}$ meters for the per-floor approach.

Obviously, this error alone does not provide a comprehensive understanding of the overall performance of the optimization methods. However, it does provide a very good visual representation for examining their behavior. For the actual evaluation of the overall error, the deviation between the model predictions and the actual RSSI value is evaluated for each reference measurement. The values can be off in either direction, both positive and negative. For this reason, we report the RMSE among all transmitters. This is $4.7\,\mathrm{dB}$ ($\sigma^2 = 3.8$) for the global method and $2.6\,\mathrm{dB}$ ($\sigma^2 = 2.7$) for the local approach. Again, the largest individual transmitter errors occur in the purple and red regions depicted in Figure 7.8. Within these areas, local maxima for the signed difference (in both directions) are observed, reaching $-31.4\,\mathrm{dB}$ and $17.5\,\mathrm{dB}$ for the global method and $-12.7\,\mathrm{dB}$ and $13.4\,\mathrm{dB}$ for local. It follows that the local optimization results in a lower overall error, although it reports a higher distance error.

The later can be explained due to the special situation in the purple region. It is a vaulted cellar that is $1.7\,\mathrm{m}$ lower than the first floor via a narrow staircase. The adjacent rooms are separated by solid stone walls. The attenuation properties of these walls, which are over 2 meters thick, are so enormous that signals from other than the two Wi-Fi beacons installed on the first floor can hardly be received. On the other hand, RSSI measurements can be received from beacons that are placed a floor above and, more importantly, vice versa. That means, that the recordings at the reference points above the vaulted basement include the beacons installed there because the ceiling is much thinner then the walls. For the global optimization model, which takes into account all available measurements, this is meaningful information, since the references from the floors above are also taken into account to optimize the parameters of the beacons in the vaulted cellar. On the other hand, local optimization only considers references from the same floor where the beacon to be optimized is located.

This may sound like a disadvantage for the local model, but as the absolute total error above indicates, that this could actually be an advantage. An example showing this, is illustrated by



(a) (b)

Figure 7.9: Short section of walk 3 inside the ROT building (cf. Figure B.1). The ILS using the global optimization model has a very high error between 170 s and 200 s. The reason for this is that the local model is able to approximate the attenuation factor β better than the global one. The figure is taken from [Fet18].

the trajectory of a localization result from walk 3 (cf. Figure B.1) in Figure 7.9. Here, a short section of the otherwise 310 m long track with a running time of approximately 10 min can be seen on the left side. The corresponding approximation error over time is on the right. For position estimation an ILS is applied as described in Section 2.5. The same parameters are used as in the experiments of Section 7.2.2, i.e., 5000 particles and for the movement model $\sigma^2_{\text{step}} = 0.1$ and $\sigma^2_{\text{turn}} = 0.1$. For resampling we set $\hbar = 0.85$ and for the evaluation step we use the sensor model for Wi-Fi as described in 3.1.4 having an uncertainty of $\sigma_{\text{wifi}} = 4.0$. The particles of the CONDENSATION filter are initialized uniformly distributed over the entire building (random position and heading), i.e. we assume no known starting position. The filter update rate is set to 500 ms and the state estimation uses weighted average.

What is immediately noticeable in Figure 7.9b is the very high error of the global model in the range between 170 s and 200 s. The pedestrian comes from the left side of the building via the horizontal corridor and enters the vertical corridor on the right side at 175 s. From here on, the ceiling is made of wood and above the rooms on the right side, there is a wooden mezzanine floor with a height of only 2 m above the first floor. Looking at Figure 7.9a, the trajectory of the global model (blue) makes a big jump into the vertical corridor and produces exactly this high error. The problem is that beacons from this mezzanine are very well received due to the low attenuation of the wooden walls. As mentioned in the discussion of the behavior in the vaulted basement (purple area in Figure 7.8), the local model on the first floor knows about these beacons only because of the reference measurements on that floor, while the global model uses all the references in the building. For each beacon, an attenuation factor β is calculated, which

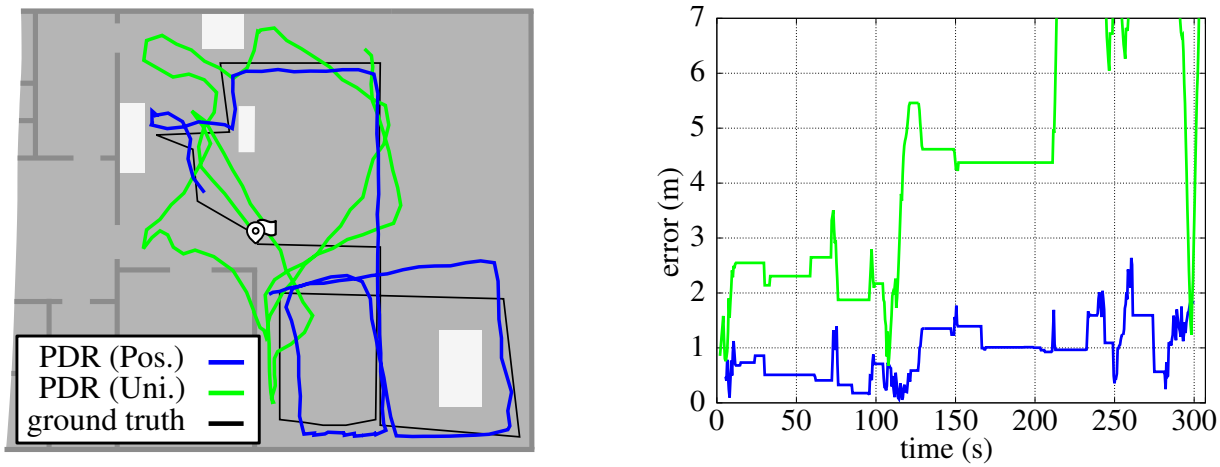
has a significant impact on the estimation of the signal strength by the CAF model in (3.3). If the references have multiple measurements from a beacon transmitting through different materials, e.g. stone, wood or concrete, then β is more of an average value and therefore does not estimate the signal strength optimally. This is the reason for the high error in the described situation. In contrast, the local model sees the beacons from the mezzanine only at isolated locations, making the attenuation factor more accurate from a local perspective.

However, this advantage of the local model changes between $200\,\mathrm{s}$ and $220\,\mathrm{s}$ when the pedestrian descends into the basement vault discussed earlier. Since only a few transmitters from the same floor are measurable there, those from the floor above become very relevant for triangulation (there are only two Wi-Fi beacons in the vaulted basement). The global model is able to optimize this using the many reference measurements available, while the local model had only a few and highly attenuated measurements. The situation could be improved with more Wi-Fi beacons, more fingerprints, or a larger number of reference measurements for the local model inside the basement.

As we could already show in our publications like [Ebn17], [Fet18] or [Pot17b], the basic use of an optimization model instead of manually determining the parameters of each transmitter not only has a massive time advantage, but also leads to a more accurate position estimation for the proposed sensor model. The reason for this can be found in the CAF model. This is used to avoid the massive computational overhead that would result from the WAF intersection tests. For both the optimization and the sensor model, the RSSI are estimated using CAF. This means that the optimized parameters are much better fitted to this (unrealistic) model than the real ones and thus have a much smaller error between measured and estimated RSSI. Since walls are ignored by CAF, the optimization of the transmitter position can compensate for the effects not considered by it. Therefore, the optimized position of a transmitter differs from the actual position, as we have already shown in Figure 7.8). However, since the same propagation model is used in the live phase, the error is minimal. If a more realistic propagation model such as WAF were used, it is expected that the optimized and real parameters would converge with increasing realism.

7.2.4 Sensor Fusion

The following experiments will investigate the fusion process in our ILS. In this context, the question of the advantages and disadvantages of the combination of the respective models will be answered. The ILS used is the same as in the previous evaluations. We use a navigation mesh with a debase strategy as proposed in chapter Section 4.3.2. The number of particles is 5000 and for the movement model we again choose $\sigma^2_{\text{step}} = 0.1$ and $\sigma^2_{\text{turn}} = 0.1$. For the resampling we



(a) (b)

Figure 7.10: Walk 1 of the IPA building with a PDR-based ILS, having only movement model and continuous activity recognition incorporated. The blue trajectory has a correct initial position, while the green does not. The ground truth path is only $72\,\mathrm{m}$ long, however a recording took $5\,\mathrm{min}$ due to 10 checkpoints where the pedestrian had to simulated real work [Fet22]

set $\hbar = 0.85$ and for the evaluation step we use the sensor model for BLE as described in 3.1.4 with an uncertainty of $\sigma_{\text{wifi}} = 5.0$, globally optimized. The filter update rate is set to $500\,\mathrm{ms}$ and the state estimation uses weighted average.

The fastest realizable ILS using the approaches proposed in this work is a pure PDR system. No external infrastructure is needed, only the pedestrian's smartphone and the already known floorplan. From an economic point of view, this is a very attractive concept. However, as described in Section 3.3, inertial sensors are subject to drift that can only be temporarily corrected, but not eliminated. For this reason, absolute sensor sources such as BLE are often used. Nevertheless, an application e.g. for short distances can be quite interesting, as can be seen in Figure 7.10 for the blue trajectory. Here you can see the movement model in combination with the continues activity recognition from 3.5.4 in the IPA building. The dataset is from walk 1, which was recorded with a Pixel 3a. Although the path is only $72\,\mathrm{m}$ long, a recording took an average of $5\,\mathrm{min}$ due to 10 checkpoints where the subject stood in front of a workstation in the workshop and simulated real work (cf. Figure B.2 for ground truth). The accuracy over all available recordings is very good, with an average localization error of $\bar{x} = 0.93\,\mathrm{m}$, a standard deviation of $\bar{\sigma} = 0.47\,\mathrm{m}$ and a .75-quantile of $\tilde{x}_{75} = 1.35\,\mathrm{m}$.

This is where activity recognition comes into its own. Even though subjects could carry the smartphone in their hand, in their pockets, on their chest, or in a hip bag during the recordings, activity recognition achieved $87.5\,\%$ accuracy across all three test tracks in the IPA building. This is all the more surprising given the small movements, such as pacing, that occurred during

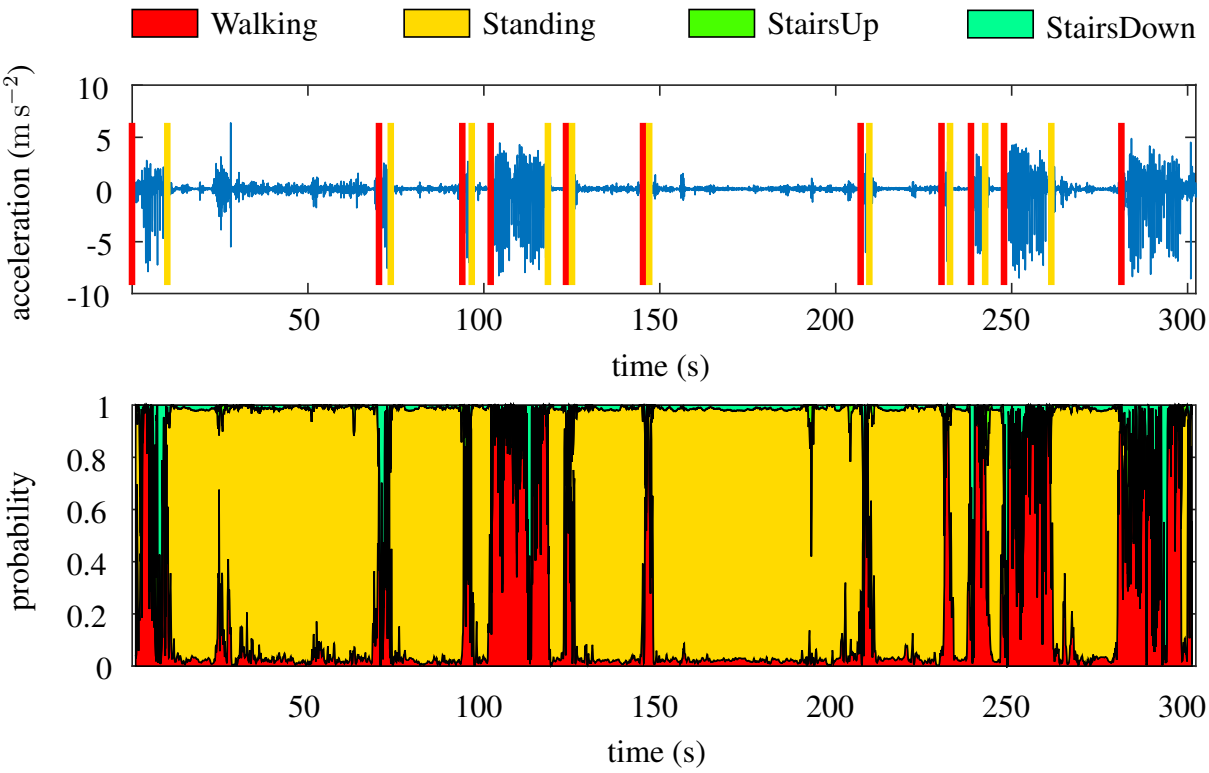


Figure 7.11: Activity detection of a data set of walk 1 from the IPA building, recorded with Pixel 3a in [Fet22]. Top: Accelerometer measurement data with the beginning of an activity plotted. Bottom: Probabilities of the respective activities.

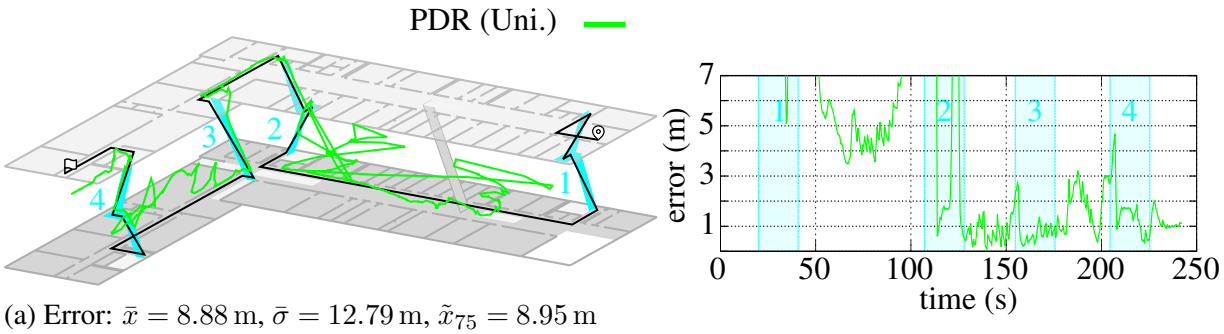
the work simulation. It was also observed that the test subjects briefly put down their smartphones to operate a cordless screwdriver. In this case, a "descending" activity was erroneously detected. Figure 7.11 gives an idea of how well the activity recognition works within the blue trajectory from Figure 7.10. At the top are the accelerometer readings with activity onset plotted. At the bottom are the probabilities of each activity. Prolonged periods of standing can be seen very well by the red areas on the time axis. These coincide very well with the work simulation phases. The activity recognition was trained with a disjoint data set from the SHL building. In order to achieve an even better detection rate, it would be worth considering to train it explicitly for the present scenario with workstations inside the IPA building.

It is important to note that the very good results of the blue trajectory in Figure 7.10 can only be achieved with an exact start position. This means that the a priori distribution $q(\boldsymbol{q}_1)$ results in a single vector, i.e. each particle was initialized with the same position and the same head. This is in contrast to the green trajectory. Here a wrong start position as well as a slightly shifted heading was chosen. You can roughly see the shape of the ground truth, but wrongly rotated and finally distorted by the contact with the lower wall. Looking back at the blue trajectory and

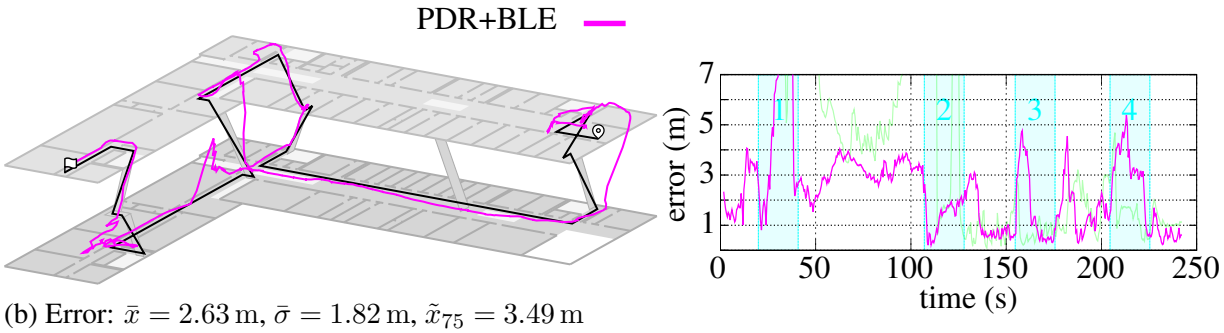
its error over time, one can see that the error becomes larger towards the end. If the distance of the test walk is extended and the subject further moves around the open workshop area, the error would increase further due to gyroscope drift.

This points out one of the main drawbacks of the proposed movement model. In the presence of large variance, i.e., uncertainty due to inaccurate sensors such as Wi-Fi RSSI, the particles will also scatter greatly because they are weighted by the sensor model as being properly realistic. Therefore, for open spaces, the particles will spread slowly over the whole area, and even if they have the right orientation and travel the right distance, the movement model will not be able to converge. On the other hand, in office buildings with small rooms and little open space, illegal movements of the particles, such as running through walls, occur more frequently when there is a high variance. For each illegal movement, the debase strategy halves the weight and the particles that are too low in weight are removed during resampling. Over time, this inevitably leads to only realistic movement being allowed and the model converging to the correct position.

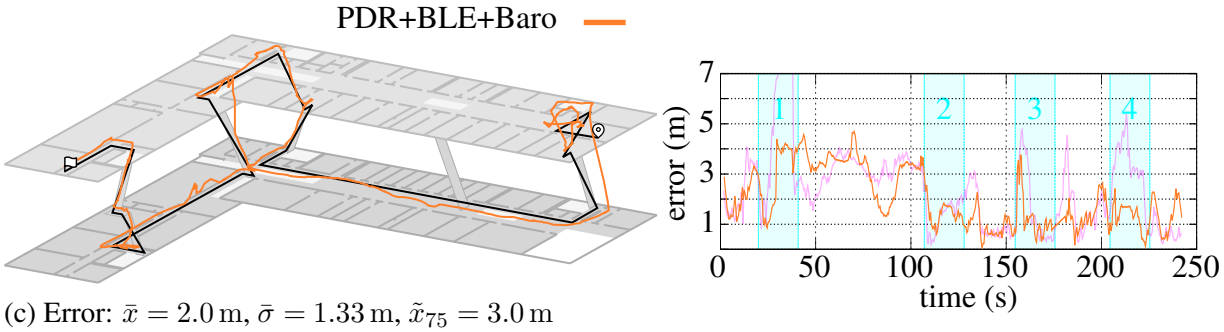
Figure 7.12a shows such a scenario for walk 7 of SHL (cf. Figure B.4) on the second and third floor of the faculty building. In the first $100\,\mathrm{s}$ of the recording, the error is still very high. This can also be seen in the trajectory of the position estimate, which is initially in the center of the building due to the uniform distribution of all particles. The first staircase is descended right at the beginning of the run. Here the movement model with the debase strategy is not yet able to converge the set of particles to the correct position, because the path taken so far is so short that it can be taken in other areas without hitting a wall. When entering the staircase, as described in chapter 3.3, particles that have a constant z-coordinate between two time steps are weighted lower than particles going down the stair. This results in a multimodal density at all four staircases, i.e. the particle set splits into four groups, one per staircase. After the right turn into the corridor below stair 1, the system slowly starts to converge because the corridor is very long and only those particles remain that do not collide with the walls on this long distance. However, the resulting unimodal density is still several meters behind the ground truth, so the error increases even more at stair 2, as most of the particles get stuck in the space in front of it instead of entering the stair. At the exit of stair 2, now back on the third floor, the left turn finally scores the erroneous particles low enough to be removed during resampling. The PDR converges very well in the phase between stair 2 and stair 3. However, when descending stair 3, due to inaccurate activity detection, not all particles change floors, but some run past stair 3 into the back corridor. This can be seen well in the trajectory between stairs 3 and 4, which is between the two floors, because the particle set is multimodal, i.e. a large part of the set is on the second floor and a smaller part on the third floor. Since the path to be covered is identical, the weights of the particles in each mode are also identical, and thus the weighted



(a) Error: $\bar{x} = 8.88\,\mathrm{m}$, $\bar{\sigma} = 12.79\,\mathrm{m}$, $\tilde{x}_{75} = 8.95\,\mathrm{m}$



(b) Error: $\bar{x} = 2.63\,\mathrm{m}$, $\bar{\sigma} = 1.82\,\mathrm{m}$, $\tilde{x}_{75} = 3.49\,\mathrm{m}$



(c) Error: $\bar{x} = 2.0\,\mathrm{m}$, $\bar{\sigma} = 1.33\,\mathrm{m}$, $\tilde{x}_{75} = 3.0\,\mathrm{m}$

Figure 7.12: Comparison of three different sensor combinations on the second and third floor of the SHL faculty building. In (a) our ILS using only the movement model (cf. Section 4.3) and the activity recognition (cf. Section 3.5.4). This results in a PDR filter without absolute source of information. In (b) we add the sensor model for BLE (cf. 3.1.4) and in (c) the one for the barometer (cf. 3.4.1). Errors are the average over 50 MC simulations for each recording.

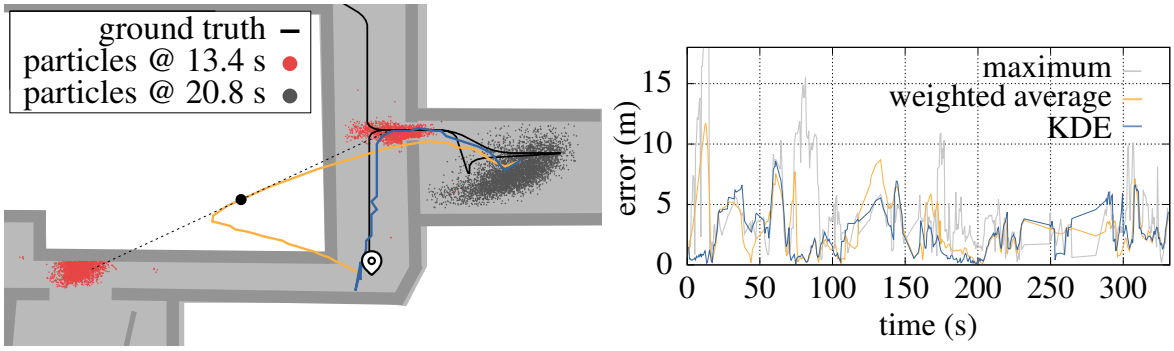
average estimation produces exactly this trajectory. After another staircase and thus detected activity, the system converges again for the last section.

Of course, this may not be a valid choice for an ILS solution, but it shows the potential of the pure PDR approach. Moreover, as we have seen in Figure 7.10, the addition of an accurate initial position might justify its use in a particular scenario. For most other applications however, adding some absolute source of information is necessary to achieve satisfying results even without any a priori knowledge about the pedestrians starting point and heading. In Fig-

ure 7.12b we are extending the PDR with the sensor model for BLE (cf. 3.1.4), optimized in a global context. Looking at the trajectory alone, the improvement is immediately apparent. However, if we also look at the error over time, we get a more differentiated picture, especially for each floor change. The stairwell of stair 1 is heavily shielded for the BLE measurements, and looking at the ground truth in Figure B.4, there are no beacons installed there either. The same is true for stair 4, so the error due to the uncertainty introduced into the posterior density by the BLE sensor model is even larger here than for the PDR alone. Stair 3, on the other hand, shows a similar behavior as the PDR-only approach, except that this effect is amplified by the open design of this staircase. This is because it is not located in a stairwell, but connects the two floors directly through an opening in the ceiling. The RSSI values of the beacons located on the third floor can therefore be received directly at the foot of stair 3 on the second floor without passing through the ceiling (LOS). The propagation model, on the other hand, includes the ceiling in the equation, which creates a discrepancy and gives a higher weight to the particles that are still on the third floor (NLOS). It is only when the subject moves away from the stairs that the multimodality can be resolved. Nevertheless, the error is reduced from $\bar{x} = 8.88\,\mathrm{m}$ ($\bar{\sigma} = 12.79\,\mathrm{m}$) for PDR alone to an average of $\bar{x} = 2.63\,\mathrm{m}$ ($\bar{\sigma} = 1.82\,\mathrm{m}$) when BLE is included.

One way to further reduce the error would be to install beacons in the respective stairwells [Ebn15]. However, at least in the SHL building, this would only be possible with battery-powered beacons, as power plugs are not allowed in stairwells of public buildings for fire safety reasons. A less expensive option for improving such locations is the barometer that is now included in many smartphones, as discussed in Section 3.4.1. Figure 7.12c shows the localization results using the barometric sensor model proposed in (3.27) with the conversion of height change (distance in z-direction) to hectopascals simplified in formula (3.28). With the exception of Staircase 3, the floor change error continued to decrease. At stair 3, the LOS vs. NLOS effect is still present, but becomes less pronounced as more particles successfully change floors. The overall trajectory is also better, especially at the stair entrances and exits, where the barometer has the greatest effect, preventing particles from leaving or entering the stair too early. The mean error continues to decrease to $\bar{x} = 2.0\,\mathrm{m}$ ($\bar{\sigma} = 1.33\,\mathrm{m}$), a value that allows for very stable room detection.

It is important to note that the above error values are calculated using only successful walks, i.e. walks that reached the destination of the path within a radius of $5\,\mathrm{m}$. For this we use a history of the last five valid position estimates of the trajectory. This is a rather generous value, since for most buildings it includes positions on the floor above or below. For example, in all MC simulations of walk 7 records with the ILS in Figure 7.12c, $14\,\%$ did not successfully complete the walk and got stuck. Without the barometer, the number was $21\,\%$. Thus, there is large potential for the advanced methods proposed in this work, as even a small failure rate ($\leq$



(a) (b)

Figure 7.13: Start section of walk 1 inside the ROT building (see Figure B.1) [Bul18]. Due to uncertain measurements, a bimodal posterior distribution occurs, reaching its maximum mode separation at 13.4 s. At 20.8 s the posterior becomes unimodal. The trajectory shown in (a) using weighted average state estimation has a high error as it takes a position between the two modes. In contrast, the KDE approach is able to find the "correct" mode and thus performs better. Nevertheless, the overall localization error between the two is quite similar, as can be seen by comparing the error over time in (b).

5 %) would be unacceptable for a real-world scenarios and would require a manual restart of the application or at least the particle filter.

7.2.5 Estimation Strategy

The primary task of an ILS is to determine the current position of the user. However, particle filters and other SMC methods first approximate a target distribution. Therefore, in Section 2.3.3, various state estimation strategies were presented as point estimators based on the weighted particle set. While the simple selection of the maximum weighted particle and the mode-search algorithm based on the KDE principle have the same goal, they are very different from the weighted-average particle. Therefore, both approaches will be compared and evaluated in the following.

An obvious scenario where the two approaches behave very differently is a multimodal posterior. An example of a bimodal distribution at the beginning of walk 1 in the ROT building is shown in Figure 7.13a. The 5000 particles are again uniformly distributed throughout the building with randomly chosen headings and the filter updates every 500 ms. Since we have no information about the absolute orientation of the pedestrian, the particles are equally likely to be moving up or to the left in the corridor. The Wi-Fi RSSI measurements at this location are very imprecise due to the strong attenuation by the walls and the small number of beacons in this area (see Figure 7.8), and therefore do not allow a clear distinction between the modes. At 13.4 s from the start of the recording, the two posterior modes reach their maximum distance

from each other, represented by the black dashed line. At $20.8\,\mathrm{s}$, the subject enters the room. Due to the movement to the right and the two beacons in the room, the particles of the lower left model are weighted low by both the debase strategy and the Wi-Fi sensor model and are finally removed during resampling. The posterior is now unimodal. Looking at the two trajectories of the estimation strategies, they behave exactly as expected. The weighted mean (orange) always takes a position between the two modes, eventually leading to a position estimate in the courtyard of the building and only resolving between $13.4\,\mathrm{s}$ and $20.8\,\mathrm{s}$. The KDE approach (blue), on the other hand, is able to find the "correct" mode because the particles in this area are weighted slightly more than the particles in the other corridor due to the Wi-Fi RSSI measurements. The trajectory, which is very similar to the ground truth (black), confirms the very low error in this part of the walk.

It is in such multimodal scenarios that a point estimator based on a KDE approach can show its strengths. However, looking at the error over time in Figure 7.13b, we see that the overall error for the entire walk 1 is just slightly better with KDE than with the weighted average. When the error is further averaged over 100 MC simulations per data set, there is little difference between the errors of the two methods. This is due to the high uncertainty of the system. For the experiments in the ROT building, this uncertainty is particularly high when entering and leaving rooms on the ground floor, while in the SHL building the uncertainty increases significantly in the heavily shielded stairwells. The resulting weaker and attenuated Wi-Fi signals behave contrary to the movement model. The latter moves the particles out of the appropriate areas at approximately the speed of pedestrian movement, while the often old, due to the small number of visible beacons, and erroneous RSSI measurements keep the majority of particles in that area. Only when the pedestrian has completely left that area and new measurements from beacons in other areas arrive, the situation is resolved. The KDE approach still shows the "correct" position (mode with the highest density) according to the RSSI data, but it actually belongs to the wrong mode of the posterior, which does not represent the true position. Therefore, the mean between the modes of the distribution is often closer to the ground truth, making the weighted average approach better in such situations.

The averaging behavior of the weighted average ultimately leads to a smoother representation of the trajectory. Therefore, the position estimation is more stable, i.e. imagine a user interface similar to OpenStreetMap, the position will not jump as much between filter updates as it would with a KDE approach. An example of this is demonstrated in Figure 7.14. The trajectories for walk 2 in the ROT building are shown for both estimation methods. The floor outlined in green is the gallery wing of the museum, which contains many small rooms in a row. The subject comes from the stairs, enters the room at the top right, and returns to leave the gallery wing at the bottom left. The ground truth in the two middle rooms forms a zigzag shape

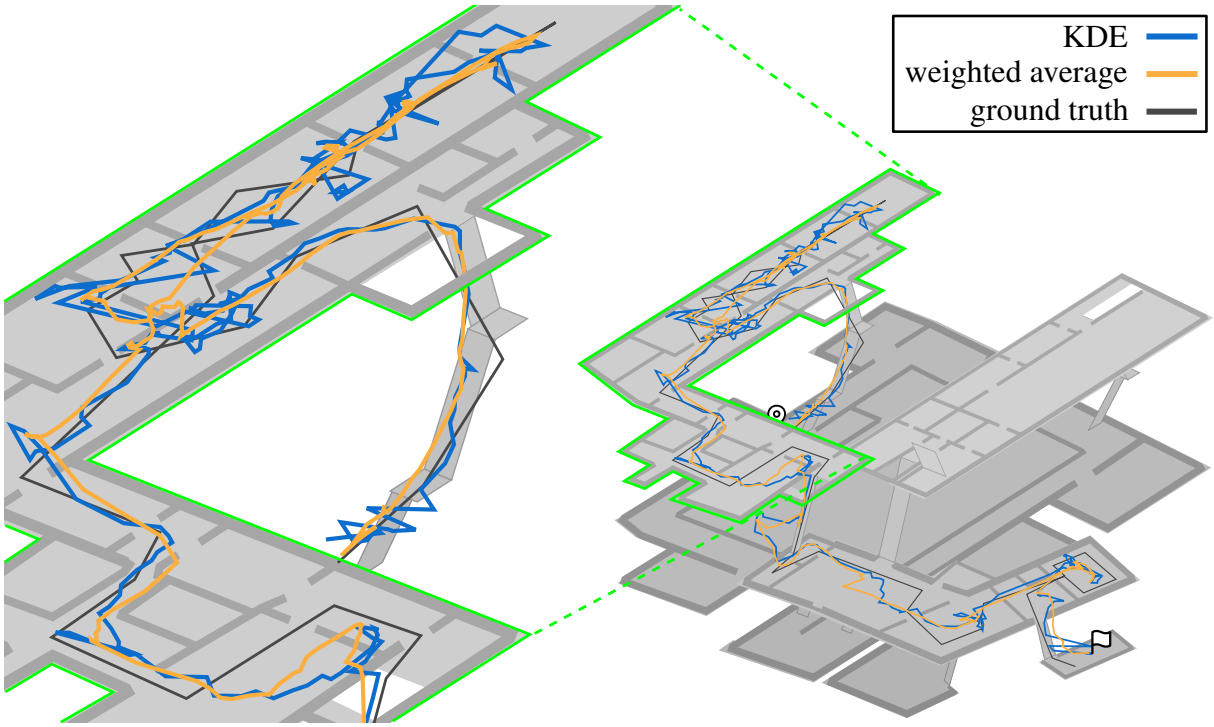


Figure 7.14: Trajectories of the localization results for walk 2 of the ROT building (see Figure B.1) using the KDE approach (blue) and the weighted average particle (orange) for state estimation. The green bordered floor is zoomed in to show the main difference between the resulting estimated positions of the two approaches. The weighted average gives a smoother trajectory, while the KDE gives a better idea of how well the most appropriate mode of the posterior fits the ground truth.

(see also Figure B.1 red path). At the vertices of the zigzag, the pedestrian stops seven times to look at paintings between 10 s to 20 s each. It can be clearly seen that the weighted average (orange) produces a very smooth trajectory in the center of the room, while the KDE approach (blue) is subject to more variability. This can be explained by the many small rooms through which pedestrians pass. The doors act as bottlenecks, so many particles run into the walls and are therefore either very low weighted or run along the wall to the door. This leads to more uncertainty and diversity in the posterior, which is better reflected by the KDE method than by the weighted average. However, even though the two estimated paths look very different, they lead to similar localization errors.

In the experiments above, we always chose the mode with the highest density for the KDE approach to obtain a position from. If one were to use a method such as boxKDE [Bul18] to represent the particle set as an analytic density, this would give rise to another variety of ways to perform a point estimate, since it is possible to sample at any point in the state space. This could allow a hybrid solution between weighted average and mode-seeking KDE. Nevertheless,

weighted average provides a reasonable result for a wide range of applications, especially since the trajectory has a smooth representation. Another important advantage is the low computational cost due to the simple linear complexity. A comparison of the runtime of both methods can be found in [Bul18].

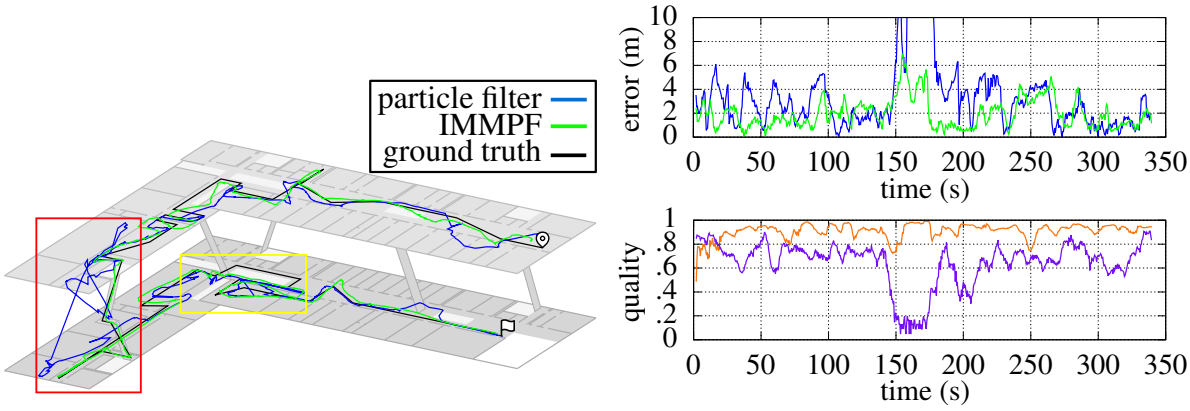
7.3 Advanced Methods

As already discussed in the introductions to Chapter 5 and Chapter 6, classical particle filter methods suffer from some general problems. These include dependencies due to the rather weak assumption of statistical independence of the respective sensor models within the state evaluation as well as the problem of sample impoverishment. For this reason, we have introduced advanced methods with the IMMPF as well as methods of PDO and adapted them to the use case of indoor localization. These methods are evaluated in the following by comparing the ILS solution of the above experiments with the extended version.

7.3.1 Multi Model Fusion

In the following experiments, we will test the hypothesis that a mixing of parallel and independent running particle filters in an IMMPF provide more accurate and robust position estimation than a conventional particle filter. As explained in 5.1.2, the mixing process of the IMMPF plays the most important role, which is why our novel non-trivial Markov switching process using dynamic quality metrics as introduce in 5.2 is of special interest. In particular, for the SHL and IPA data sets, we compare a particle filter as shown in Figure 7.12b, i.e. a combination of movement model (PDR) and radio technology for absolute position information, with an IMMPF ($M = 2$) that uses a PDR filter $m_t = 1$ (transition only) and a pure radio-based filter $m_t = 2$ (evaluation only) separately. In the SHL building, we use BLE and choose the Markov transition matrix as in (5.24). For the IPA building, we choose the transition matrix from (5.23) because here a Wi-Fi FTM infrastructure is provided by 22 ESP32-S1 microcontrollers.

We again run 50 MC simulations using 5000 uniformly distributed particles with random heading per filter and mode of the IMMPF. The sensor noise is modeled according to $\sigma^2_{\text{turn}} = 0.1$, $\sigma^2_{\text{step}} = 0.1$, $\sigma^2_{\text{ble}} = 9$ and $\sigma^2_{\text{ftm}} = 5$. For Wi-Fi FTM we made a regression analysis as described in (3.10), resulting in $a = -42.82$ and $b = 16.15$. For the N_{eff} we set $\hbar = 0.3$ for the particle filter (see Algorithm 5) and $\bar{\hbar} = 0.85$, $\hbar_1 = 0.3$ and $\hbar_2 = 0.8$ was chosen for the mixing step and the two modes of the IMMPF (see Algorithm 7). All filter are again using an update rate of $500\,\text{ms}$, meaning that $\boldsymbol{o}_t$ contains all measurements of that particular period. Finally, as point



(a) (b)

Figure 7.15: Results of particle filter (blue) and IMMPF (green) for a recording of walk 6 in the faculty building of the SHL campus (see Figure B.4). The trajectories are shown in (a) and the error over time in (b). Below, the quality metrics q_{pdr} (orange) and q_{ble} (purple) are plotted over time. The red rectangle shows an interesting scenario discussed in the text between $140\,\text{s}$ and $200\,\text{s}$ and the yellow between $235\,\text{s}$ and $260\,\text{s}$. Figure is based on [Fet23a].

estimator for providing a position and drawing the trajectories we use the weighted average estimation.

A representative trajectory for walk 6, which runs along the second and third floors of the faculty building on the SHL campus, is shown in Figure 7.15a. The walk begins on the third floor on the right, traverses the entire floor with a few branches to the stairwell on the far left. From there it goes to the second floor and back along a similarly complex path. Looking at the results for the IMMPF (green) and the particle filter (blue), especially the trajectory of the former is very close to the ground truth (black). An exception is the area marked with a yellow square. Here, neither the IMMPF nor the particle filter succeeds in mapping the actual ground truth. The reason for this discrepancy is an opening in the floor that divides the corridor and thus represents an inaccessible area. As a result, a significant portion of the particles are redirected into this inaccessible area. The debase strategy used by the movement model reduces the weight of particles attempting to enter this area. The resampling step then removes this large fraction of particles, causing the remaining particles to move in a straight line to the right out of space, instead of correctly upward. This eventually leads to the trajectories shown in the yellow square.

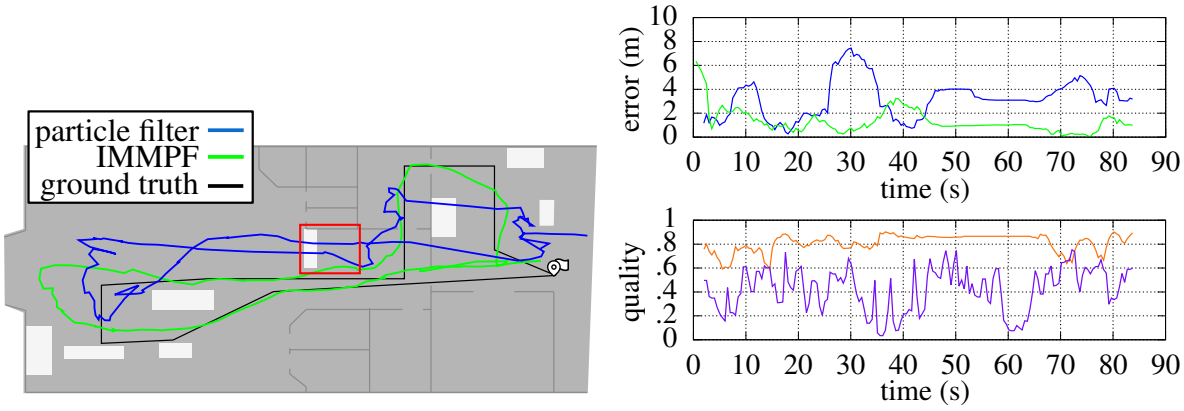
Another very interesting scenario, which takes place between $140\,\text{s}$ and $200\,\text{s}$, can be seen in the red square. Here, the path goes through a stairwell that is heavily shielded by a metal-coated door, a service shaft lined with aluminum panels, and a reinforced concrete structure. In addition, no BLE beacon has been installed at this location. All this leads to a very high measurement error and associated uncertainty of the BLE sensor model. Nevertheless, both the trajectory and the error over time are visibly better for the IMMPF than for the conventional

particle filter. The reason for this can be seen very well in the quality of the respective sensor models over time (cf. Figure 7.15b below). This is because it is precisely in the range between $140\,\mathrm{s}$ and $200\,\mathrm{s}$ that the quality metric q_{ble} (purple) of the BLE drops massively and, due to the transition matrix in (5.24), ensures that the PDR-based filter of the IMMPF is given a correspondingly higher priority in the mixing. This ultimately compensates for the high error of the BLE. The particle filter, on the other hand, is forced to treat both sensor models equally and thus fully incorporates the measurement uncertainty of the BLE.

Overall, the plot of the quality metric q_{pdr} (orange) is very interesting. Over the entire recording, its mean is about 0.9, with downward exceptions especially in the first $20\,\mathrm{s}$. Remembering the experiments from Figure 7.12 in Section 7.2.4, this is due to the initial uncertainty of the filter based on the a priori uniform distribution of all particles in the building. Due to the mixing in the IMMPF, the particles in the BLE filter are immediately prioritized higher, thus flowing into the PDR filter and shortening the time to convergence of the PDR. Overall, the movement model benefits greatly from the SHL building structure. The many small rooms mean that particles that do not move in the central corridor hit a wall and are quickly removed. As a result, the particle set is quite compact, as the debase strategy allows little variance (cf. 5.2.6). This is reflected in the high mean of q_{pdr} (orange). Comparing this to q_{ble} (purple), we see that IMMPF basically trusts the PDR filter more than the BLE filter. Thus, the joint posterior distribution is more influenced by the more accurate PDR filter than the more uncertain BLE filter. Looking at the error over time, the IMMPF benefits greatly, especially up to $200\,\mathrm{s}$. Overall, the localization error over all recordings is $\bar{x} = 2.1\,\mathrm{m}$ ($\bar{\sigma} = 1.32\,\mathrm{m}$) for the IMMPF, while the particle filter achieves only $\bar{x} = 3.27\,\mathrm{m}$ ($\bar{\sigma} = 2.53\,\mathrm{m}$). Thus, the improvements with the IMMPF are significant, even though the sensor models are basically the same as with the particle filter.

A very similar behavior of the IMMPF can be observed on walk 2 in the IPA building. The walk starts on the far right and leads past workbenches to a small room with a kitchen. Passing through a central meeting room, one enters the open exhibition space. The user stops at the lower left position for $20\,\mathrm{s}$, as can also be seen by the constant quality of q_{pdr} (orange) in Figure 7.16b. After a $90°$ turn to the left, we come back to the starting point. In addition to the PDR filter, we use a filter based solely on a Wi-Fi FTM sensor model.

As we saw in Section 3.2, Wi-Fi FTM distances should provide better location accuracy than RSSI measurements. However, as shown in our papers [Bul20b] and [Fet23a], FTM measurements are prone to multipath effects, which result in a nonlinear, non-Gaussian, and often non-unimodal error distribution. Looking at the values of the quality metric q_{ftm} (purple) over time in Figure 7.16b, and considering that the IPA building consists of highly reflective glass facades and corrugated metal, this suggests such a degradation effect. The mean for q_{ftm} is about 0.4, while q_{pdr} (orange) is about 0.8. Thus, similar to the previous walk in SHL, the IMMPF fa-



(a) (b)

Figure 7.16: Results of particle filter (blue) and IMMPF (green) for a recording of walk 2 in the IPA building (see Figure B.2) taken from [Fet23a]. The trajectories are shown in (a) and the error over time in (b). Below, the quality metrics q_{pdr} (orange) and q_{ftm} (purple) are plotted over time. The red rectangle shows an interesting scenario discussed in the text around $30\,\text{s}$.

vors the PDR-based filter over the FTM filter in the mixing step. Looking at the error over time, there is again a strong improvement using the IMMPF (green). The trajectory in Figure 7.16a is also much closer to reality compared to the CONDENSATION particle filter. This results in a mean error over all recordings of $\bar{x} = 3.18\,\text{m}$ ($\bar{\sigma} = 1.36\,\text{m}$) for the particle filter and $\bar{x} = 1.95\,\text{m}$ ($\bar{\sigma} = 1.30\,\text{m}$) for the IMMPF. Again, a very significant improvement.

The very high error of the particle filter (blue) between $25\,\text{s}$ and $35\,\text{s}$ occurs in the area marked in red in Figure 7.16a. Due to the inaccurate FTM measurements, most of the particles are drawn into the area between the white box and the adjacent wall. The white box represents an obstacle (e.g. table) that is constrained by the movement model to be non-walkable. This state lasts for a few seconds as the FTM measurements confirm this position. Only new measurements in combination with the debase strategy will resolve the situation and the particles that are already on the far left side of the walk will be evaluated accordingly high. This leads to a ”jump“ of the position estimated by the weighted average from the white box to the far left. On the other hand, the IMMPF avoids getting stuck because the PDR filter has already a stronger influence from the beginning of the walk. As we have seen in Figure 7.10 the PDR filter is able to provide good results even for open spaces, at least for short distances.

Looking at the last two experiments, it seems reasonable to assume that the use of a PDR filter alone should be sufficient to obtain good localization results. However, as explained in the discussion to Figure 7.12, using the PDR filter alone is a reasonable decision only in exceptional cases, e.g. known starting position and short distances. This leads to the conclusion that even the use of a very inaccurate sensor model with absolute position information is meaningful, if

SHL							IPA			ROT			
1	2	3	4	5	6	7	1	2	3	1	2	3	4
8 %	44 %	0 %	8 %	11 %	8 %	18 %	0 %	0 %	0 %	20 %	6 %	8 %	7 %

Table 7.2: Percentage of 50 simulation runs that got stuck and did not reach the destination within a radius of 10 m for all conducted walks of the data sets. The ILS is the same as in Section 7.2.4. It uses 5000 particles, activity recognition, barometer and either Wi-Fi (ROT, SHL walk 1 to 5), BLE (SHL walk 6 and 7) or Wi-Fi FTM (IPA) as absolute source of information. No impoverishment strategy or multi modal fusion is involved.

the resulting increased uncertainty is appropriately considered and compensated in the sensor fusion process. The IMMPF is a tool that meets exactly these requirements.

7.3.2 Impoverishment Prevention

Throughout this thesis, especially in Chapter 2, we have encountered the general problem of sample impoverishment. In the context of sensor fusion, the problem can be so severe that the estimator completely loses track, gets stuck, and never recovers. For this reason, we introduced several PDO methods in Chapter 6 that claim to mitigate or even eliminate the problem.

Table 7.2 gives an overview of the frequency of impoverishment. It shows for each walk from all datasets how often the ILS got stuck and did not reach the target within a radius of at least 10 m. The particle filter, in the following called None as it does not use any advanced methods, used is a CONDENSATION filter with 5000 particles, activity detection, barometer, and either Wi-Fi, BLE, or Wi-Fi FTM. The problem could not be detected in the IPA building. This is due to the small size of the building and the large open spaces, where the particles have a high variance, i.e. are widely scattered, as discussed above. Across all walks, the probability of missing is 13.86 % for the SHL campus and 10.25 % for ROT. In a commercial ILS, these values would likely be extremely harmful to user acceptance and are therefore unacceptable.

Walk 2 and walk 7 of SHL are particularly noteworthy; both have a very similar start sequence, starting only a few meters from one of the isolated stairwells (reinforced concrete, no beacons) and leading directly into it after a few seconds. At this point the uncertainty in the whole system is still very high due to the unknown starting position, which often leads to the fact that the following floor change is not recognized and the particle set wanders around on the wrong floor. Floor changes are only possible on stairs, i.e. a free "jump" to another floor is not represented by the movement model. This leads to getting stuck 18 % of the time for walk 7 and even 44 % for walk 2. The latter is particularly challenging. Here, two floors are directly changed (see Figure B.3) and after a short phase on the ground floor, the walk leads once across the courtyard from the faculty to the lecture hall building. Except for the two short sections on

	SHL walk 2				ROT walk 1			
	$\bar{x}$	$\bar{\sigma}$	$\tilde{x}_{75}$	stuck	$\bar{x}$	$\bar{\sigma}$	$\tilde{x}_{75}$	stuck
None	19.09 m	19.23 m	31.77 m	44 %	9.30 m	9.76 m	13.41 m	20 %
KLD 5k	18.88 m	16.18 m	29.43 m	40 %	10.12 m	9.57 m	15.39 m	18 %
KLD 25k	16.86 m	21.57 m	26.44 m	32 %	6.26 m	7.38 m	5.72 m	8 %
KDR	27.63 m	20.82 m	47.26 m	100 %	14.30 m	10.68 m	22.70 m	40 %
Random	18.81 m	19.07 m	30.89 m	20 %	2.88 m	2.34 m	3.40 m	0 %
SFA	5.27 m	3.65 m	7.25 m	2 %	2.76 m	2.53 m	3.31 m	0 %
UDE	13.94 m	15.87 m	22.29 m	0 %	2.80 m	2.30 m	3.31 m	0 %

Table 7.3: Overall localization results using the advanced methods presented in Chapter 6 for walk 2 of SHL and walk 1 of ROT. The ILS uses 5000 particles, activity detection, barometer, and Wi-Fi as an absolute information source. We ran 50 simulation runs. A run got stuck if it did not reach the target within a radius of at least 10 m

the ground floor of each building, the Wi-Fi signals are highly intermittent. This is partly due to the heavily isolated stairwell, and even more so because the only radio signals coming from the buildings are through the metalized windows. This leads to strong attenuation and multipath effects. Finally, the outdoor area is a very large open space, which increases the degrees of freedom of the movement model and thus the uncertainty in the PDR.

In the ROT building, walk 1 is particularly affected by severe impoverishment. In 20 % of the cases, the walk gets completely stuck. But even if a walk is successful, the impoverishment due to incorrect Wi-Fi measurements between 10 s and 20 s leads to a very high error in the first 60 s (see Figure 7.18, red line). In the wild, this error could cause the user to stop in the area of incorrect measurements and check the functionality of the application. From the user's point of view, it is not obvious that the path does not converge for another 40 s.

In the following experiments, we will focus on walk 2 in SHL, since it has the highest percentage of "getting stuck", and walk 1 in ROT, as impoverishment causes also a very high intermediate error. The results for all PDO approaches given in Chapter 6 are presented in Table 7.3. We will discuss them in a top down manner, starting with the KLD approaches present in Section 6.1.1. The main idea behind this approach was to allow an adaptive particle size over time, with the hope that in situations of higher uncertainty, the increasing number of particles would be able to prevent impoverishment. Two parameter variations were investigated. KLD 5k starts with $N = 5000$ particles and can adaptively use a maximum of $N_{\text{max}} = 10000$ particles. KLD 25k starts with $N = 25000$ particles and can reach a maximum of $N_{\text{max}} = 50000$ particles. For both we also set the error bound values to $\delta = 0.01$ and $\epsilon = 0.05$ as suggested by [Fox01] and [TBF05]. Comparing the two with None (ILS as in Section 7.2.4), only KLD 25k can significantly improve the results. This can be explained by the larger number of particles, as formulated at the beginning of Section 2.3. However, as has already been investigated in

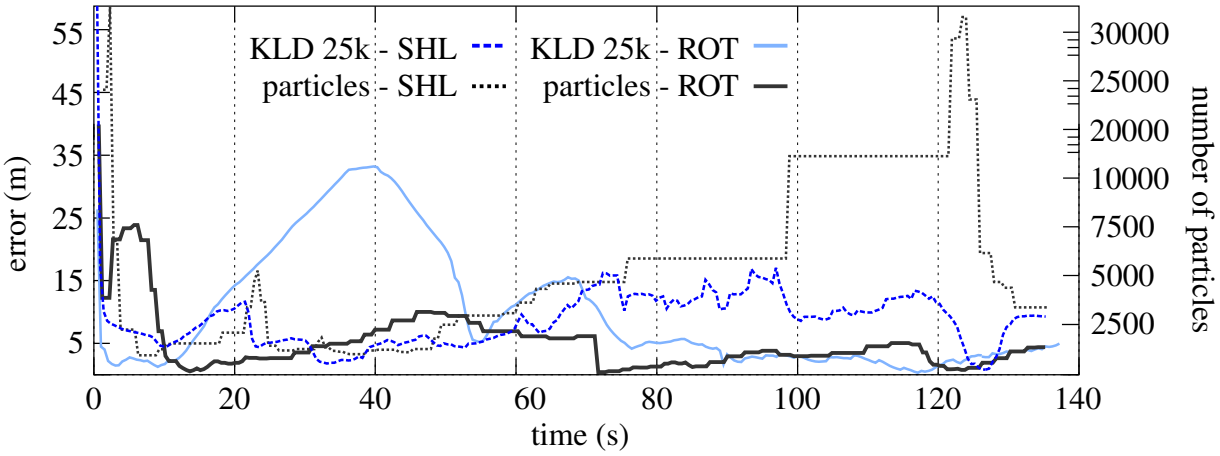


Figure 7.17: Error (blue) and number of particles (black) over time using KLD-Resampling as shown in Algorithm 9 for walk 2 in SHL (dotted) and walk 1 in ROT (solid). Only a very weak correlation between both entities can be observed. For SHL the number of particles increases significantly when going into the large outside open space (courtyard) between 80 s and 120 s.

several fundamental publications, e.g. by [TBF05] or [Che03], there is no linear relationship between increasing particle number and decreasing error. Rather, our experiments show that the maximum number of particles required depends strongly on the particular scenario.

Figure 7.17 shows the number of particles determined by KLD resampling for a recording of walk 2 in SHL and walk 1 in ROT. At first glance, there is a very weak correlation between the error (solid line) and the number of particles (dashed line). The error between 20 s and 60 s, caused by faulty Wi-Fi measurements, is very high for walk 1 in ROT, but the number of particles does not increase significantly. On the other hand, they do increase significantly when walking in the courtyard of SHL, despite the relatively smaller error. The reason for this behavior can be found in the assumption made for the KLD approach that the true posterior $\tau(\boldsymbol{q}_{1:t})$ is based on a discrete piecewise constant distribution, such as a histogram or a grid, overlaid on the multidimensional state space. We were able to show that the number of particles required is first order linear in the number k of bins with support. In simple terms, this means that if a set of particles scatters more as the uncertainty increases, then the number k must increase. In the case of walk 2 in the SHL, this is exactly what happens in the large open space of the courtyard. Since there are no obstacles such as walls, the particles have a high degree of freedom and can quickly move anywhere in the area. In contrast, the high error in ROT occurs in an area with many small rooms and a short staircase (see Figure B.1, yellow). The particles are thus distributed very slowly due to the debase strategy of the restrictive movement model, which leads to a low k and thus to a low adaptive particle number.

To extract more particles in such scenarios, which is necessary to contain the impoverishment problem, the bin size of the histogram can be further reduced. However, for our experi-

ments, we have already chosen a very small value of $20\,\mathrm{cm}$ per dimension. When we reduced the value to $10\,\mathrm{cm}$ per dimension, the average number of particles required per run increased from 1617 to 45554. The purpose of the KLD approach must be questioned here, since the advantage is actually to be able to reduce the number of particles to a minimum at locations with a very high degree of certainty in order to save computational resources. Perhaps an extension of the movement model or the KLD approach for mutual consideration could add real value at this point.

The only approach worsening the results of None is *KDE resampling* (KDR) as presented in 6.1.2. The reason for this is the structure of the spatial model, as described in Chapter 4. We restrict the particles to this model. This means that movement is only possible on the navigation mesh or grid graph, i.e. only parallel to a floor or staircase. Any three-dimensional position, e.g. "in the air" between two floors or outside the building, is not possible. Although not quite correct, this is often referred to as a two and a half dimensional representation. Now, if you imagine a set of particles moving toward a staircase to change floors, a bottleneck is created at that point. The set of particles becomes very compact at the threshold to the staircase because the degrees of freedom in the z-direction are simply limited by the spatial model. Around the stairs there are still many particles left on the surface of the floor. In our case, KDR now approximates a KDE over this set using boxKDE [Bul18]. The resulting probability density is well defined in the plane of the floor (xy-plane), but hardly defined along the staircase (z-axis). The state space in the z-direction is simply too narrow and restrictive, so the approximation of KDE can only map this area poorly. This leads to the fact that hardly any particles are drawn on the staircase during resampling, which prevents a floor change in $100\,\%$ of the cases, as it is the case with an existing impoverishment in the staircase of the SHL building. Similar to the KLD approach, further research is needed to adapt KDR for two and a half dimensional spatial models.

The situation is different for two-dimensional use cases such as the IPA building, walk 1 and walk 5 in SHL, or walk 2 in ROT. The impoverishment that occurs in SHL and ROT was successfully eliminated, i.e., the rate of stuck runs shown in Table 7.2 was reduced to $0\,\%$. However, there was no significant improvement in the localization error for successful runs. As already discussed in Section 7.2.5, the approach shown here does not yet offer any real added value for a real application, but has great potential due to the possibility of an analytical representation of the posterior. For example, we currently draw the particles for resampling (see Section 2.3.2) multinomially over all bins of the KDE approximation. Other strategies are conceivable, e.g. with a stronger focus on floor changes.

Looking again at the overall results in Table 7.3 the last three approaches, namely Random, SFA and UDE, start to significantly improve the results. In the ROT building in particular, the

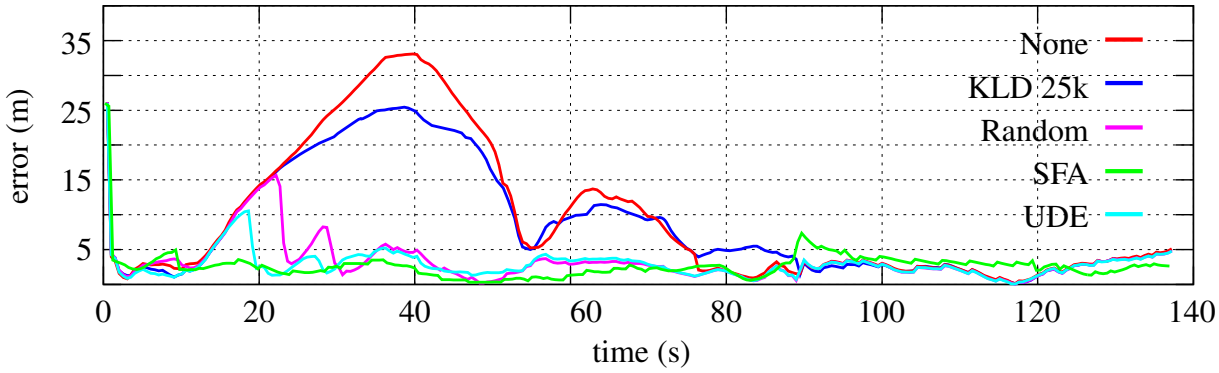


Figure 7.18: Error over time using the different approaches presented in Chapter 6 for a single recording of walk 1 of ROT using the same random seed. KDE and KLD 5k (maximum 5000 particles) do not improve None (ILS as presented in Section 7.2.4) and are therefore omitted. Between $10\,\mathrm{s}$ and $20\,\mathrm{s}$ the Wi-Fi signal was highly attenuated, causing an impoverishment and thus a high error. UDE and Random are able to reduce this effect to a minimum, while SFA is able to eliminate it completely.

results are almost identical. Figure 7.18 shows the error over time for a recording of the data set. The same random seed was chosen for each approach to ensure comparability. In the case of None (red), the high error caused by the impoverishment can be clearly seen. After $80\,\mathrm{s}$ at the latest, all methods converge to an error of less than $5\,\mathrm{m}$. With the exception of SFA, which differs fundamentally from the others due to the IMMPF, the error curve is almost identical. Looking at the period between $10\,\mathrm{s}$ and $40\,\mathrm{s}$, both Random (pink) and UDE (cyan) are briefly affected by the erroneous Wi-Fi measurement, but are able to recover after a few seconds. Random is slightly slower to recover than UDE in this recording, which is simply due to the random factor involved in whether one of the randomly drawn particles ($0.5\,\%$ of the total set) is in the right place at the right time, i.e. where the sensor models suggest the true position of the pedestrian should be.

The larger the building, the more particles need to be randomly drawn to counteract inertia. However, this can lead to uncontrolled behavior, e.g. due to large measurement errors. Such a situation can be observed in Figure 7.19 of walk 2 in the SHL. In the stairwell and in the courtyard, the Wi-Fi quality is poor due to the lack of access points, as only highly attenuated signals are received from the buildings, which are distorted by multipath effects. Random (pink) can complete the floor changes up to about $60\,\mathrm{s}$, but after that the trajectory is pulled upward by a randomly placed particle on the first floor and the increasing measurement error of the Wi-Fi at that point. In the further course of the recording, no position estimation is achieved in the inner courtyard anymore, because the few randomly placed particles are weighted too low and therefore quickly removed during resampling. Only when entering the lecture hall the weighting is sufficient to eliminate the impoverishment.

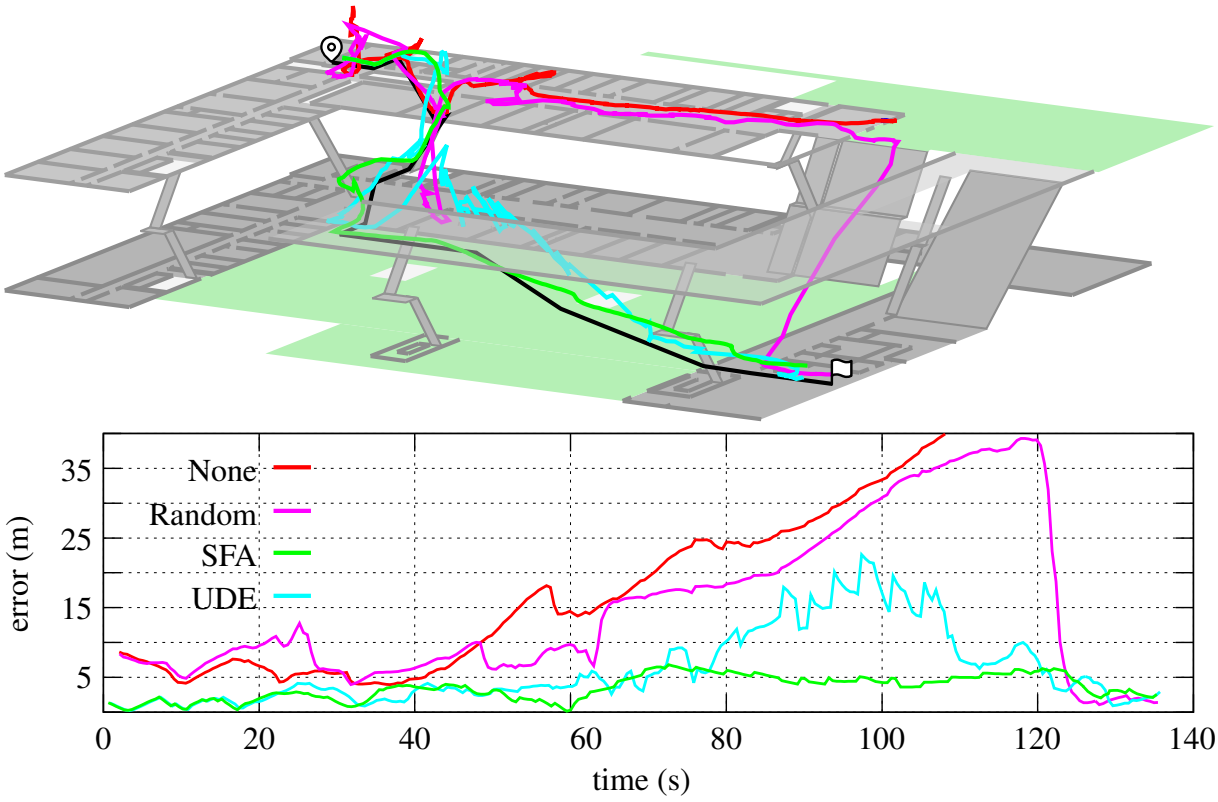


Figure 7.19: Trajectory (top) and error over time (bottom) using the different approaches presented in Chapter 6 for a single recording of walk 2 of SHL (see Figure B.4) using the same random seed. KDE and KLD also get stuck in the first stairwell, similar to None (ILS as presented in Section 7.2.4), and are therefore omitted. In the large open space (courtyard) between $80\,\mathrm{s}$ and $120\,\mathrm{s}$ the error increases significantly because Wi-Fi access points are only available inside the buildings, so the signals are highly attenuated.

For UDE (cyan) the situation is similar, but there are enough static particles Υ_{static}, 10000 over the complete spatial model, in the courtyard to use for resampling (cf. 6.2.3). The error is still high, but at least the trajectory is outside, even if it tends to zigzag in the upper left half of the courtyard directly on the building facade for a while. The closer it gets to the lecture hall building, the more Wi-Fi signals are received from there, so the position estimate becomes a bit more accurate after about $110\,\mathrm{s}$. In principle, both random and UDE depend very much on the quality of the sensor models of the evaluation step, i.e. the Wi-Fi RSSI model. According to the stuck criterion we defined to show the impoverishment, both were able to significantly improve the results compared to None, UDE even to $0\,\%$ over all walks. However, if the end of walk 2 in SHL had been performed directly in the courtyard, this rate would probably have been different. Nevertheless, the experiments show that both approaches are very competent methods to limit impoverishment effects early on in purely indoor scenarios with good radio

coverage. Both have a manageable complexity, with Random having virtually no overhead in computational resources.

Finally, the SFA approach based on the multi modal fusion of the IMMPF has a very good performance in all aspects. The reason for this, as already shown in the experiments of Section 7.3.1, lies in the quality metrics used. The dominant filter is a PDR filter similar to the one used before and the support filter uses a movement model of the form (4.9), where only the outer walls of the building are considered in the mesh, i.e. rooms are ignored and the particles can move freely. The other parameters are identical to the previous experiments.

Figure 7.18 clearly shows how the SFA (green) prevents the initial high error from occurring in the first place, since the Wi-Fi quality matrix q_{wifi} is very low in the period between $10\,\text{s}$ and $20\,\text{s}$, i.e. hardly any particles are drawn from the support filter. Therefore, the position estimation is mainly based on the PDR and not on the uncertain Wi-Fi sensor model. Looking at the error over time, the SFA shows an increased error at 90 seconds and just before $120\,\text{s}$ compared to the other approaches. At both points, a floor change occurs via an open staircase (no surrounding walls, see Figure B.1). At these locations, a particularly large number of Wi-Fi measurements are received from the access point of the two adjacent floors, so the Wi-Fi quality increases according to (5.11). However, no fingerprints were recorded on the stairs to train the global optimization model, which inevitably leads to a significantly higher uncertainty of the radio propagation model at these locations.

Another very interesting observation can be made in the courtyard of the SHL building starting at about $70\,\text{s}$ when looking at Figure 7.19. Although SFA (green) performs significantly better than the other approaches, the average localization error increases in this section. Again, the cause can be found in the quality metrics. As already discussed, the Wi-Fi quality is poor here, also due to the large open area and the large distance from building to building, the PDR also drifts strongly and leads to bimodality due to two obstacles (see Figure 7.19, gray boxes in the green courtyard). As a result, its quality metric q_{pdr} also decreases, but remains slightly more influential than q_{wifi} up to about $120\,\text{s}$, resulting in significantly lower overall errors compared to UDA. An even longer walk outside the building would probably have led to very similar results for SFA and UDE in the long run. However, the usefulness of such a scenario should be questioned and additional sensor information such as GNSS or outdoor access points might be considered instead in a real-world application.

Chapter 8

Summary

Indoor localization is a lively and very active field of research due to its high complexity (nonlinear dynamics and non-Gaussian uncertainty). As we have seen in 1.2.1, there are many different approaches to solve the problem of localizing a person or an object in a building. Hybrid methods based on the basic principle of sensor fusion are one of the most promising approaches, since smartphone-based self-localization makes a large number of different sensors available for inference. The basic idea is that the more sensors are combined with each other, the more accurate and robust statements can be made. In addition, possible failures of individual sensors can be better compensated.

In short, we use sensor fusion to combine a set of sensors to make inferences about a pedestrian's position that would not be possible with a single sensor alone. The knowledge for this is obtained indirectly from the associated (noisy) measurements that yield an observation $\boldsymbol{o}$. Obviously, such a process suffers from a variety of uncertainties due to the imperfection and incompleteness of the sensors and the methods used. Using the principles of statistical inference, we are able to describe these uncertainties in a mathematical way. In particular, we focus on a Bayesian interpretation of probability, as it allows to add a priori knowledge to the estimation process. Since indoor localization, as well as most sensor fusion applications, takes place in a time-sequential setting, this leads to the broad class of Bayes filters as introduced in Section 2.2.

With the marginal of the joint posterior, also called the filtering distribution, one of the most important notations of the whole work is given in (2.35). It clearly shows the necessary models to be developed; evaluation, transition, and prior model. Since a closed-form solution for (2.35) is not to be expected in practice, we use the principles of MC simulation for a numerical solution, ultimately leading to the very popular class of particle filter in Section 2.4. Here, the two widely used methods CONDENSATION (see Section 2.4.1) and APF (see Section 2.4.2) are derived. Although the latter is more advanced by adding the current observation $\boldsymbol{o}_t$ to the

importance distribution, the CONDENSATION approach is extremely easy to implement since it does not require any further information or parameterization except for the models mentioned above. By weakening the Markov property, we introduce an extension to CONDENSATION that includes the previous observation $\boldsymbol{o}_{t-1}$ in the transition and the previous state $\boldsymbol{q}_{t-1}$ in the evaluation. This is of course the same as the APF adding $\boldsymbol{o}_t$, but provides a very sophisticated method that is in-between the simplicity of CONDENSATION and the more advanced APF. Based on these principles, a generic filter framework is presented in Section 2.5.1, which allows a general formulation of an ILS for all types of Bayse filters. The advantage lies in the separation of the ILS from its concrete implementation, which allows a comparability of different approaches and leads to a real modularity of sensor models (see e.g. (3.1)).

Sensor models, whether for physical or virtual sensors, are presented in Chapter 3. Their purpose is to provide inferences about the possible position of the pedestrian, which are then used in the evaluation step of the particle filter for sensor fusion. In order to obtain absolute position information, i.e. independent of the pedestrian's movement, we have presented powerful models for Wi-Fi and Bluetooth RSSI in Section 3.1 and Wi-Fi FTM in Chapter Section 3.2. Instead of the widely used fingerprinting, we apply radio propagation models for the RSSI measurements in order to obtain a position estimate by means of multilateration. Instead of a complex geometric calculation, a probabilistic model is used (see (3.5)), as with all sensor models in this work. Here, a probability density is formulated for each measurement, which can be easily combined with each other under the assumption of statistical independence. The resulting joint density function describes the most probable position of the pedestrian. Finally, in order to achieve a similar accuracy to fingerprinting, but without a high calibration effort, we have proposed optimization procedures for the respective model parameters in chapter 3.1.5. Despite high susceptibility to errors (background noise, attenuation, multipath effects), BLE RSSI measurements in particular are the de facto standard sensor for smartphone-based indoor localization.

With Wi-Fi FTM, we proposed a promising alternative in Section 3.2, which was explicitly developed with the idea of a protocol designed for localization and not for data transmission like RSSI. However, as discussed in detail in Section Section 3.2.2, there is little agreement in the literature on which effects mainly influence the error/noise of the sensor, so we proposed a data-driven model in Section Section 3.2.3. For this purpose, we model a skewed normal distribution in $1\,\mathrm{m}$ wide bins of a histogram. We optimize the respective parameters per bin from data sets collected in a calibration process, and then fit continuous functions for the respective parameters to smooth the transition between bins. This results in the probabilistic sensor model in (3.17) and (3.15), which can again be thought of as a kind of multilateration in the form of a joint

probability density. One of the major limitations of FTM is its poor hardware and software support, which makes it currently not a real alternative to RSSI.

In addition to the absolute position data measured by external transmitters on the smartphone, the internal sensors in the form of the IMU and the barometer play an important role in incorporating the pedestrian's motion into the evaluation. Due to the high drift that occurs when integrating acceleration, we first propose a step detection based on classical signal processing in chapter Section 3.3.1. In order to get more than just a binary decision whether a step was detected or not, we present an extension in (3.20) that provides a probability based on a gamma distribution. This has the advantage that particles that have not moved the same distance as a step in the transition are not completely excluded during the evaluation, but are weighted according to this probability.

The heading, i.e. the direction in which the pedestrian is moving, is determined by integrating the gyroscope data rotated in the ENU frame (angular change). We compute the necessary transformation using the Madgwick filter, which in turn is a sensor fusion of gyroscope and accelerometer using gradient descent. To avoid misinterpreting every change in orientation of the smartphone as a change in heading, e.g. turning the smartphone from portrait to landscape, the motion vector is also determined. The motion vector always points in the walking direction of the pedestrian, i.e. it rotates with the smartphone, so that a change in the orientation of the device can be detected as described in (3.23). The sensor model in (3.22) can then map the evaluation step using a dynamic van Mises distribution as described in (3.25). Another extension, which we did not use, would be to include the magnetometer to obtain an absolute orientation via the earth's magnetic field.

Especially when entering three-dimensional environments such as a multi-story building, accurate detection of floor changes becomes a critical factor. Therefore, in Section 3.4, we propose a sensor model based on a barometer. However, as can be seen in the model in (3.27), we again use a relative solution by using the difference between the measured atmospheric pressure and an initial reference measurement. This is due to the fact that we basically assume that we have no prior knowledge about the pedestrian's initial position, i.e. which floor he is on, secondly we have no information about the height above sea level of the building, and finally we were able to show that absolute air pressure measurements are subject to extreme fluctuations in short time intervals. Considering relative values also has the advantage that the barometric formula, used to convert atmospheric pressure to altitude, can be greatly simplified. In addition, due to the similar climatic conditions within a building and the rather small height differences between the individual floors, further assumptions can be made in the form of constant parameters (see (3.28)). With all this, very good results are obtained for continuous and discrete floor changes on stairs; only elevators remain difficult to model.

In addition to movement, i.e. in which direction and how far the pedestrian is moving, the activity being performed, e.g. climbing stairs, is useful additional information. This allows conclusions to be drawn about the current position in the building, e.g. whether a person is on a staircase, and allows the parameters of the other sensor models and the movement model to be adjusted depending on the activity. Section 3.5 therefore presents two different models for detecting the typical activities of standing, walking and climbing stairs. The very simple threshold-based activity recognition in Section 3.5.2 can be quickly and easily integrated with a handful of parameters, although this is reflected in the rather moderate recognition rate, especially for standing. This is in contrast to the adequate analytical transformation, which uses a classic windowed pattern recognition pipeline with PCA for dimension reduction (see Figure 3.11). Using a multi-class SVM classification, the approach achieves very good recognition rates of well over $90\,\%$, especially for walking and standing. The result is a virtual sensor that, for each activity, provides a probability that the activity is currently being performed. A corresponding probabilistic sensor model that can be used for fusion is then proposed in Section 3.5.4. As can be seen in (3.31), each possible activity is compared with a movement of the particle from time $t-1$ to t, e.g. a change in z-direction when climbing stairs, and evaluated according to the probability received from the recognition. In the ILS, this ultimately leads to particles that have a similar movement pattern to the current activity gaining importance in the posterior. It should always be noted that due to the window functions used, there is a difference (lag) between the execution and the actual recognition of the activity.

One of the biggest challenges in pedestrian localization is the modeling of human movements. Due to the high degrees of freedom such as walking direction, speed, locomotion, etc., this is of non-linear complexity. When moving indoors, additional factors are added, such as changing floors by climbing stairs or riding an elevator, scenario-related activities such as standing at a workstation or looking at an exhibit, or simply restrictions due to unpassable obstacles such as walls. As we have seen, the latter in particular offers opportunities because it restricts movement to walkable surfaces, i.e., places where the pedestrian can realistically move. In order to identify such areas or walls, a spatial representation model based on a floor plan is required. Considering the service-oriented and efficiency-related requirements for a spatial representation from [ARC12b], we have chosen two different spatial models as the basis for movement modeling in Chapter 4, namely a grid graph and a navigation mesh.

The usage of either grid or mesh depends heavily on the scenario and the sensor models used. The regular tessellation of the grid makes it very easy to store metadata directly on the it. For example, this is a great advantage for fingerprinting methods (Wi-Fi fingerprinting or magnetic matching), as a corresponding fingerprint can be located directly at each vertex $\boldsymbol{v}_i$ of the graph G, including its position. In addition, methods that use a grid or bins to approximate

the state space, such as KLD, KDE or UDA, also benefit from an existing grid structure. The accuracy then corresponds exactly to the granularity g_s of the graph. Disadvantages are the increased memory requirements mentioned in Section 4.2.4 and the restriction of the movement model proposed in Section 4.2.3 to movements that can only represent a multiple of $45°$.

In contrast to this, the navigation mesh shown in Section 4.3 with its irregularity reduces the memory requirement considerably, since large areas can be represented with only a few triangles. The movement model in Section 4.3.2 shows how continuous movement can be implemented efficiently and without complex intersection tests. It allows for additional strategies to deal with particles that have performed an illegal action, such as running into a wall. These strategies allow a pure PDR approach without information about the starting position of the pedestrian in buildings with many small rooms, such as office buildings, to converge after some time and achieve a good position estimate. This is particularly valuable when other sensors, such as Wi-Fi or BLE, have poor or no coverage in certain areas of the building. If the network is also to be used for navigation (route guidance) or to store meta-information, more advanced methods are required, such as the use of barycentric coordinates or an additional grid.

The idea of combining individual sensors to achieve greater accuracy and robustness is well founded, and the particle filter is a good tool for doing so. However, the high degree of modularity achieved by simply combining the individual models in the evaluation is not only covered by the assumption of statistical independence of the respective models, but also by the (naive) assumption that a combination yields a higher accuracy than one sensor alone. However, this is not the case if individual observations have a very high uncertainty due to measurement errors. This inevitably leads to a higher uncertainty in the whole system. One way to deal with this was presented in Chapter 5 with the IMMPF. Instead of combining all sensors in one evaluation step, several independent filters can be used. This allows any combination of sensor models and movement models. For example, a Wi-Fi and BLE fingerprinting filter can be combined with a grid-based movement model, and a pure PDR filter can be combined with a navigation mesh-based filter. The advantage is that even if the RSSI measurements in the example above were heavily distorted by attenuation, the PDR filter is still able to deliver reasonable results, as we have seen in several examples throughout the experiments in 7.

The combination, or mixing, of the individual filters is primarily controlled by the Markov transition matrix. It gives the probability of moving from m_t to m_{t+1} in one time step, thus denoting a Markov switching process on the underlying Markov chain. Using quality metrics for each presented sensor model, we present a novel dynamic Markov switching process in Section 5.1.2. This is one of the most important results of this work and allows for the consideration of faulty measurements, approximation errors and other phenomena that occur during execution and can directly influence the impact of a filter or sensor on the overall position estimation.

Another important contribution of this work has been the attempt to control and, in the best case, eliminate the general problem of sample impoverishment. For this purpose, PDO approaches were adapted to the problem of indoor localization or newly proposed in Chapter 6. The KLD (see Section 6.1.1) and KDR (see Section 6.1.2) approaches promise general applicability, i.e. independent of the sensor models used in the ILS. In KLD, the number of particles available to a filter for the posterior approximation is determined dynamically using a statistical bound based on the Kullback-Leibler divergence between the current posterior and the true target distribution. The idea is that the greater the uncertainty in the system, the more particles are available for the approximation. In theory, this should also result in a larger part of the state space being covered by particles, making impoverishment more unrealistic. However, as shown in the experiments in Section 7.3.2, in practice this often has only a small effect.

The situation with KDR is similar; again, the theoretical properties have not been able to significantly improve position estimation in practice. KDR relies on resampling, which uses a continuous representation of the posterior using KDE. Instead of extracting particles from the fixed particle set, particles can be sampled at arbitrary positions. KDR thus covers a larger region of the state space than the classical resampling methods described in Section 2.3.2. In the two-dimensional domain (one floor), KDR is therefore able to successfully eliminate impoverishment. For multi-floor buildings, however, the approach does not work due to the few degrees of freedom in the state space (only walking on stairs), as we discussed in Section 7.3.2.

These general PDO methods are contrasted with data-driven methods that use some degree of context and/or sensor knowledge to solve the problem. The simplest approach is to randomly sample some particles (e.g. $1\,\%$) over the entire state space rather than following the resampling strategy used (see Section 6.2.1). Although this can lead to unpredictable behaviour due to the random factor involved, especially if measurement errors occur, this method is particularly convincing for smaller buildings.

The best, but also the most complex, method for eliminating impoverishment is SFA (see Section 6.2.2). It uses the special properties of the IMMPF proposed in Chapter 6. The transition matrix in (6.16) is used to continuously compare a dominant filter, which is a highly accurate ILS, and a support filter, which is particularly robust to impoverishment, using Kullback-Leibler divergence. If the two probability densities of the respective filters differ too much, it is likely that the dominant filter will be subject to impoverishment and the support filter will become more important in the mixture. As the quality of the sensor models used for the support filter deteriorates, the influence of the support filter automatically decreases. This is a huge advantage, especially in situations such as the isolated stairwells in the SHL building.

UDE in Section 6.2.3 concludes the contributions to this thesis. The approach is intended to be a compromise between the high complexity of SFA due to the use of two filters and the

simplicity of random sampling. Instead of a support filter, a second static particle set is used for resampling. This is weighted based on the evaluation to draw a small number of particles multinomially. UDE is simple, fast to implement and the computational complexity is low. It was able to avoid impoverishment in all test datasets, but is inferior to SFA in the accuracy of position estimation.

All the methods proposed in this thesis have been evaluated in one way or another in the experiments described in Chapter 7, compared as far as possible and the results discussed in detail. For this purpose, a technically complex and time-consuming experimental setup with software tools, process sequences, data structures and finally a very large number of data sets was set up. To ensure good reproducibility, the tools and the resulting experimental design are available as free software. The data sets were recorded in three very different building complexes and contain realistic walking routes under real-world conditions (see Appendix B).

In Section 7.2 the respective sensor and movement models were first compared. This confirmed many of the hypotheses put forward in Section 3 and Section 4. The results from Section 7.2.4 are of particular interest, as here we combine all models step by step to obtain the best possible ILS for our purposes. Due to the higher accuracy resulting from the continuous properties (see Section 7.2.2), we chose the navigation mesh as the movement model and the global optimization model (see Section 7.2.3) in order to always be able to optimize with the maximum number of fingerprints.

This ILS, which is already very powerful and works well in a large part of the data sets, was now used as the basis for the advanced methods in Section 5 and Section 6. Again, the IMMPF approach is characterized by its high robustness against impoverishment combined with very high accuracy. The interaction of independently operating filters, which agree on a joint posterior based on their own quality, delivers very good results despite extremely difficult conditions. Perhaps the most exciting thing is that we currently believe that the full potential of this method for indoor localization research has not yet been exhausted. We therefore hope that this work will pave the way for a new perspective away from the classical all-combining filter towards true modularity of arbitrary complexity.

Chapter 9

Future Work

Looking at the results in Table 7.3 of the experiments in Section 7.3.2, SFA promises good accuracy despite difficult conditions and an infrastructure that is simply not designed for localization. Our victories in the 2016 and 2023 IPIN competitions, in which we chose UDA, have once again confirmed that our approach is de facto the state-of-the-art for localization with commercial smartphone technologies (BLE, IMU, barometer, and map) [Eva]. Nevertheless, or perhaps because of this, we still see room for improvement. Looking first at the localization error, which averages about $2\,\mathrm{m}$ across all data sets, it becomes clear that the use cases are limited to navigation and spatially accurate positioning. Other problems such as exhibit accuracy or spaghetti diagrams for work process analysis still require special sensor technologies such as UWB or LiDAR [Fet22]. The sensor fusion proposed here is well suited to integrate these sensors, in particular SFA can handle the discrepancy between the different accuracy resolutions between them. Nevertheless, there is still a need to get the maximum accuracy out of the sensor and movement models proposed here. Other factors such as stability of position estimation, computational complexity, current limitations of the proposed models and flexibility for other use cases such as multi-target tracking also offer potential for improvement. Therefore, the following section provides an outlook on future research projects with some concrete recommendations.

A very obvious point at the moment is the limitation of the movement models used. A very critical point is the lack of step length estimation. Currently this is approximated by a single value as in (4.3) or by activity recognition as in (4.5). However, this is only an approximation of the average distance traveled by a person per step. People with walking disabilities or activities such as running are difficult to represent. For this reason, a relatively large σ between $0.1\,\mathrm{m}$ and $0.3\,\mathrm{m}$ must be set as noise. This increases the uncertainty and inevitably worsens the position estimation. As discussed in Section 3.3, implementing step length estimation without

additional hardware is extremely complex, especially when the smartphone is held in the hand. Deep learning approaches like [KA20] and [Her22] offer potential. These could train a parameterization per user and possibly even per time step for the model proposed here. It would also be conceivable to extend activity recognition to other activities such as running, strolling, or walking on a stick.

One sensor that was not used at all in this work is the magnetometer. However, it offers great potential for use in more or less static environments because of the possibility of obtaining other absolute position information in addition to radio data. A first step would be to integrate a magnetic matching sensor model into the proposed ILS. Especially the approaches of [Sol16] are very promising and have been proven in practice. In addition to position estimation based on the fingerprinting principle, an absolute heading of the pedestrian can be determined. This information is completely missing in our proposed system and can at best be approximated by determining a heading averaged over all particles. To determine the absolute heading, the anomalies of the earth's magnetic field caused by the building must be subtracted to obtain results similar to those of a compass in a free field. In addition to improved visualization for the user in an user interface, this would also support the PDR, as the relative change in heading θ can then be related to the absolute value in the PDR, and not to a random value per particle as before [LCH20]. As a result, the system will converge faster and have less noise overall.

The optimization from Section 3.1.5 currently trains a static parameter set per radio transmitter for CAF in (3.3). We make two assumptions: first, that both path loss and attenuation do not change dynamically, and second, that the influence of walls can be ignored. Although we try to compensate for these factors to some extent by optimizing the position of the access point, this is always done in a static context. Ray tracing is a technique known from GNSS and computer graphics that allows a more accurate physical modeling of radio propagation. Representing a radio signal emanating from the transmitter as a ray that passes through various walls and is attenuated or reflected by other surfaces until it reaches the antenna of the receiving smartphone is an approximation with asymptotic properties, similar to MC simulation. The implementation is often similar to fingerprinting; reference measurements are taken in the building. Based on their positions, a set of beams is sent around the building and tracked based on the physical conditions of the building and some predefined criteria, such as the maximum distance to be covered, until they hit a transmitter. From the resulting trajectory, a signal strength for that beam can be determined using WAF, which is then compared in the live phase with the actual measurement as in (3.5). Such a model can be completely pre-calculated for each building or used live with the appropriate computing capacity, e.g. a graphics card cluster. However, the big problem here is the need for a precise building model that provides accurate information about the building structures, such as construction materials, wall thicknesses, or

window positions. A possible approach to avoid the very time-consuming creation of such a model is again an optimization procedure, as we proposed in Section 3.1.5, not only for the parameters of the access point alone, but also, for example, to determine the attenuation factors of the individual materials used in the building. Research projects such as [Tay09], [Hyu19] and [SMS23] show the potential of ray tracing, but have yet to gain traction. This may change as computing power of handheld devices further increases.

By using the IMMPF as proposed in Chapter 5 for the first time in the field of indoor localization and adapting it to perform dynamic mixing, we have opened a big door for future work. Besides the introduction of further quality metrics, the combination of different filter variants, or the investigation of the relationship between different movement models, there are still some general questions (see the discussion in 6.1.2). One challenge is the search for general quality metrics that can make reliable statements independent of the sensors used. This would make it possible to easily integrate arbitrary filters into the IMMPF without any adjustments. Another limitation is the requirement of an identical update rate for all modes, since the mixing process must be performed synchronously (see Algorithm 7). Especially when high frequency sensors like LiDAR are combined with low frequency sensors like BLE RSSI, this often leads to time discrepancies between the filters. Again, an individual update rate per filter would probably not only improve the position estimation, but also further simplify the combination of existing filters.

The last methods to be mentioned for future work are those that rely on a fast approximation of the KDE, such as KDR or the mean shift algorithm (see Section 6.1.2 and Section 2.1). The potential of both methods is evident in the experiments, but is limited to finding the maximum posterior mode. However, the availability of a continuous approximation of the probability density function allows the use of statistical inference methods in the field of particle filters, which have been little used or not used at all. This is because the presence of multimodalities in the posterior is still a key factor for inaccurate state estimations. The interesting thing is that with a reasonably trained eye, the ”true“ mode can often be recognized when plotting the particle set. We therefore believe that appropriate solutions can be found.

In conclusion, we encourage everyone to test, adapt and adopt the methods proposed in this work. Thank you for reading and thank you for caring.

List of Figures

List of Algorithms

List of Tables

Listings

Bibliography

[Aba14] M. J. Abadi, L. Luceri, M. Hassan, C. T. Chou, and M. Nicoli. "A collaborative approach to heading estimation for smartphone-based PDR indoor localisation." In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Oct. 2014, pp. 554–563.

[AC99] A. Azzalini and A. Capitanio. "Statistical applications of the multivariate skew normal distribution." en. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00194, pp. 579–602.

[AHZ21] M. Aleksandrov, D. J. Heslop, and S. Zlatanova. "3D Indoor Environment Abstraction for Crowd Simulations in Complex Buildings." en. In: *Buildings* 11.10 (Oct. 2021). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 445.

[Ala16] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa. "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances." en. In: *Sensors* 16.5 (May 2016). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 707.

[Ald97] J. Aldrich et al. "RA Fisher and the Making of Maximum Likelihood 1912-1922." In: *Statistical Science* 12.3 (1997), pp. 162–176.

[And03] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. "An Introduction to MCMC for Machine Learning." en. In: *Machine Learning* 50.1 (Jan. 2003), pp. 5–43.

[Ang97] M. V. Anglada. "An improved incremental algorithm for constructing restricted Delaunay triangulations." en. In: *Computers & Graphics*. Graphics Hardware 21.2 (Mar. 1997), pp. 215–223.

[AP90] D. K. Arrowsmith and C. M. Place. *An Introduction to Dynamical Systems*. en. Google-Books-ID: HOU6q6wHCncC. Cambridge University Press, July 1990.

[ARC12a] I. Afyouni, C. Ray, and C. Christophe. "Spatial models for context-aware indoor navigation systems: A survey." In: *Journal of Spatial Information Science* 1.4 (May 2012), pp. 85–123.

[ARC12b] I. Afyouni, C. Ray, and C. Christophe. "Spatial models for context-aware indoor navigation systems: A survey." In: *Journal of Spatial Information Science* 1.4 (May 2012), pp. 85–123.

[Arn01] Arnaud Doucet and Nando De Freitas Neil Gordon. "An Introduction to Sequential Monte Carlo Methods." In: *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.

[Au16] E. Au. "The Latest Progress on IEEE 802.11mc and IEEE 802.11ai [Standards]." In: *IEEE Vehicular Technology Magazine* 11.3 (2016), pp. 19–21.

[Ban18] L. Banin, O. Bar-Shalom, N. Dvorecki, and Y. Amizur. "Scalable Wi-Fi client self-positioning using cooperative FTM-sensors." In: *IEEE Transactions on Instrumentation and Measurement* 68.10 (2018), pp. 3686–3698.

[Bar22] V. Barral, O. Campos, T. Domínguez-Bolaño, C. J. Escudero, and J. A. García-Naya. "Fine Time Measurement for the Internet of Things: A Practical Approach Using ESP32." In: *IEEE Internet of Things Journal* (2022). Conference Name: IEEE Internet of Things Journal, pp. 1–1.

[BCX19] C. Briso, C. Calvo, and Y. Xu. "UWB Propagation Measurements and Modelling in Large Indoor Environments." In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 41913–41920.

[BD03] Y Boers and J. N. Driessen. "Interacting multiple model particle filter." In: *October* 150.5 (2003), pp. 344–349.

[BD19] R. Bassett and J. Deride. "Maximum a Posteriori Estimators as a Limit of Bayes Estimators." In: *Mathematical Programming* 174.1-2 (Mar. 2019). arXiv:1611.05917 [math, stat], pp. 129–144.

[BDW06] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II." In: *IEEE Robotics & Automation Magazine* 13.3 (Sept. 2006). Conference Name: IEEE Robotics & Automation Magazine, pp. 108–117.

[Bea] Beaconstac. *Buy Long-Range Outdoor Beacon | 300m range | Works with Eddystone & iBeacon*. en.

[Ber13] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. en. Google-Books-ID: 1CDaBwAAQBAJ. Springer Science & Business Media, Mar. 2013.

[BH13] A. Brajdic and R. Harle. "Walk detection and step counting on unconstrained smartphones." In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. UbiComp '13. New York, NY, USA: Association for Computing Machinery, Sept. 2013, pp. 225–234.

[BHE01] N. Bulusu, J. Heidemann, and D. Estrin. "Adaptive beacon placement." In: *Proceedings 21st International Conference on Distributed Computing Systems*. Apr. 2001, pp. 489–498.

[BHM97] B. Bracio, W. Horn, and D. Moller. "Sensor fusion in biomedical systems." In: *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 'Magnificent Milestones and Emerging Opportunities in Medical Engineering' (Cat. No.97CH36136)*. Vol. 3. ISSN: 1094-687X. Oct. 1997, 1387–1390 vol.3.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Vol. 1. Springer, 2006.

[Bla09] J.-L. Blanco. "Contributions to Localization, Mapping and Navigation in Mobile Robotics." PhD thesis. PhD. in Electrical Engineering, University of Malaga, 2009.

[Bos07] A. v. d. Bos. *Parameter Estimation for Scientists and Engineers*. en. John Wiley & Sons, Aug. 2007.

[Bot13] A. Botea, B. Bouzy, M. Buro, C. Bauckhage, and D. Nau. "Pathfinding in Games." en. In: (2013). Artwork Size: 11 pages Medium: application/pdf Publisher: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany Version Number: 1.0, 11 pages.

[Bow04] K. Bowyer. "Face recognition technology: security versus privacy." In: *IEEE Technology and Society Magazine* 23.1 (2004). Conference Name: IEEE Technology and Society Magazine, pp. 9–19.

[Box76] G. E. P. Box. "Science and Statistics." In: *Journal of the American Statistical Association* 71.356 (Dec. 1976). Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1976.10480949, pp. 791–799.

[BP00] P. Bahl and V. N. Padmanabhan. “RADAR: An In-Building RF-based User Location and Tracking System.” In: *Proc. of the 19th Annu. Joint Conf. of the IEEE Computer and Communications Societies*. Vol. 2. 2000, pp. 775–784.

[BP11] A. Bernacchia and S. Pigolotti. “Self-Consistent Method for Density Estimation.” In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 73.3 (June 2011), pp. 407–422.

[Bre17] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer. “Evolution of Indoor Positioning Technologies: A Survey.” en. In: *Journal of Sensors* 2017 (Mar. 2017). Publisher: Hindawi, e2630413.

[BS09] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. en. John Wiley & Sons, Sept. 2009.

[BSA16] L. Banin, U. Schatzberg, and Y. Amizur. “WiFi FTM and map information fusion for accurate positioning.” In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[BSC86] Y. Bar-Shalom and L. Campo. “The Effect of the Common Process Noise on the Two-Sensor Fused-Track Covariance.” In: *IEEE Transactions on Aerospace and Electronic Systems* AES-22.6 (Nov. 1986). Conference Name: IEEE Transactions on Aerospace and Electronic Systems, pp. 803–805.

[BSLK01] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. en. John Wiley & Sons, June 2001.

[BSWT11] Y. Bar-Shalom, P. K. Willett, and X. Tian. *Tracking and data fusion: a handbook of algorithms*. en. Storrs, CT: YBS Publishing, 2011.

[Bul18] M. Bullmann, T. Fetzer, F. Ebner, F. Deinzer, and M. Grzegorzek. “Fast Kernel Density Estimation Using Gaussian Filter Approximation.” In: *International Conference on Information Fusion, FUSION 2018*. Cambridge, United Kingdom, 2018, pp. 1233–1240.

[Bul20a] M. Bullmann, F. Ebner, M. Ebner, T. Fetzer, and L. Köping. *SensorReadoutApp*. `https://github.com/simpleLoc/SensorReadoutApp`. (Visited: 19.07.2022). 2020.

[Bul20b] M. Bullmann, T. Fetzer, F. Ebner, M. Ebner, F. Deinzer, and M. Grzegorzek. “Comparison of 2.4 GHz WiFi FTM- and RSSI-Based Indoor Positioning Methods in Realistic Scenarios.” In: *Sensors* 20.16 (2020).

[Bul22] M. Bullmann, T. Fetzer, M. Ebner, S. Kastner, F. Deinzer, and M. Grzegorzek. "Data Driven Sensor Model for Wi-Fi Fine Timing Measurement." In: *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Sept. 2022, pp. 1–8.

[Bul23] M. Bullmann, F. Ebner, M. Ebner, T. Fetzer, and L. Köping. *SensorFingerprintApp*. `https://github.com/simpleLoc/SensorFingerprintApp`. (Visited: 19.07.2022). 2023.

[BW84] A. Bundy and L. Wallen. "Breadth-First Search." en. In: *Catalogue of Artificial Intelligence Tools*. Ed. by A. Bundy and L. Wallen. Symbolic Computation. Berlin, Heidelberg: Springer, 1984, pp. 13–13.

[BYB07] E. R. Bachmann, X. Yun, and A. Brumfield. "Limitations of Attitude Estimnation Algorithms for Inertial/Magnetic Sensor Modules." In: *IEEE Robotics & Automation Magazine* 14.3 (Sept. 2007). Conference Name: IEEE Robotics & Automation Magazine, pp. 76–87.

[CB21] G. Casella and R. L. Berger. *Statistical Inference*. en. Cengage Learning, Jan. 2021.

[CB95] H. Choset and J. Burdick. "Sensor based planning. I. The generalized Voronoi graph." In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 2. ISSN: 1050-4729. May 1995, 1649–1655 vol.2.

[CCF99] J. Carpenter, P. Clifford, and P. Fearnhead. "Improved particle filter for nonlinear problems." en. In: *IEE Proceedings - Radar, Sonar and Navigation* 146.1 (Feb. 1999). Publisher: IET Digital Library, pp. 2–7.

[CFY76] A. Charnes, E. L. Frome, and P. L. Yu. "The Equivalence of Generalized Least Squares and Maximum Likelihood Estimates in the Exponential Family." In: *Journal of the American Statistical Association* 71.353 (Mar. 1976), pp. 169–171.

[Che03] Z. Chen. "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond." In: *Statistics* 182.1 (2003), pp. 1–69.

[Che87] L. P. Chew. "Constrained Delaunay triangulations." In: *Proceedings of the third annual symposium on Computational geometry*. SCG '87. New York, NY, USA: Association for Computing Machinery, Oct. 1987, pp. 215–222.

[Che95] Y. Cheng. "Mean shift, mode seeking, and clustering." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (Aug. 1995). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 790–799.

[Cie10] B. Ciepłuch, R. Jacob, P. Mooney, and A. C. Winstanley. "Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps." en. In: *Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resuorces and Enviromental Sciences 20-23rd July 2010* (2010). Publisher: University of Leicester, p. 337.

[CO67] E. Condon and H. Odishaw. *Handbook of physics*. McGraw-Hill handbooks. McGraw-Hill, 1967.

[CP20] N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. en. Springer Series in Statistics. Cham: Springer International Publishing, 2020.

[CP17] V. Cantón Paterna, A. Calveras Augé, J. Paradells Aspas, and M. A. Pérez Bullones. "A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering." en. In: *Sensors* 17.12 (Dec. 2017). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 2927.

[Cri17] C. Cristodaro, F. Dovis, G. Falco, and M. Pini. "GNSS receiver performance in urban environment: Challenges and test approaches for automotive applications." In: *2017 International Conference of Electrical and Electronic Technologies for Automotive*. June 2017, pp. 1–6.

[CT18] Z.-C. Chen and C.-W. Tang. "Robust Pedestrian Tracking Using Interactive Multiple Model Particle Filter and Feature Matching." In: *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*. July 2018, pp. 480–485.

[CW98] F. Chin and C. A. Wang. "Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time." In: *SIAM Journal on Computing* 28.2 (Jan. 1998). Publisher: Society for Industrial and Applied Mathematics, pp. 471–486.

[Das01] B. V. Dasarathy. "Information Fusion – what, where, why, when, and how?" en. In: *Information Fusion* 2.2 (June 2001), pp. 75–76.

[DB05] H. Driessen and Y. Boers. "Efficient Particle Filter for Jump Markov Nonlinear Systems." In: *Radar, Sonar and Navigation, IEE Proceedings* - 152.5 (2005), pp. 323–326.

[DDN01] F. Deinzer, J. Denzler, and H. Niemann. "On Fusion of Multiple Views for Active Object Recognition." In: *Pattern Recognition – Proc. of the 23rd DAGM Symp.* Ed. by B. Radig. Vol. 2191. LNCS. Heidelberg: Springer, 2001, pp. 239–245.

[Dei05] F. Deinzer. *Optimale Ansichtenauswahl in der aktiven Objekterkennung*. Vol. 20. Studien zur Mustererkennung. Logos Verlag Berlin, 2005.

[Del09] D. Delling. "Engineering and Augmenting Route Planning Algorithms." Dissertation. Universität Fridericiana zu Karlsruhe, Oct. 2009.

[Del34] B. Delaunay. "Sur la sphère vide. A la mémoire de Georges Vorono." In: *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles* 6 (1934), p. 793.

[Dev98] O. Devillers. "Improved incremental randomized Delaunay triangulation." en. In: *Proceedings of the fourteenth annual symposium on Computational geometry - SCG '98*. Minneapolis, Minnesota, United States: ACM Press, 1998, pp. 106–115.

[DH10] H. Dodel and D. Häupler. "Die Satellitennavigation." de. In: *Satellitennavigation*. Ed. by H. Dodel and D. Häupler. Berlin, Heidelberg: Springer, 2010, pp. 1–43.

[DH15] S. Dimitrov and H. Haas. *Principles of LED Light Communications: Towards Networked Li-Fi*. Cambridge University Press, 2015.

[Dic16] P. Dickinson, G. Cielniak, O. Szymanezyk, and M. Mannion. "Indoor positioning of shoppers using a network of Bluetooth Low Energy beacons." In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. 2016, pp. 1–8.

[DJ09] A. Doucet and A. Johansen. "A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later." In: *Handbook of Nonlinear Filtering* 12 (Jan. 2009).

[DK05] K.-B. Duan and S. S. Keerthi. "Which Is the Best Multiclass SVM Method? An Empirical Study." en. In: *Multiple Classifier Systems*. Ed. by N. C. Oza, R. Polikar, J. Kittler, and F. Roli. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 278–285.

[DMAZC20] M. E. Diago-Mosquera, A. Aragón-Zavala, and G. Castañón. "Bringing It Indoors: A Review of Narrowband Radio Propagation Modeling for Enclosed Spaces." In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 103875–103899.

[DMB98] L. Devroye, E. P. Mücke, and Binhai Zhu. “A Note on Point Location in Delaunay Triangulations of Random Points.” en. In: *Algorithmica* 22.4 (Dec. 1998), pp. 477–482.

[DMDJ06] P. Del Moral, A. Doucet, and A. Jasra. “Sequential Monte Carlo samplers.” en. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2006.00553.x, pp. 411–436.

[DP17] P. Davidson and R. Piché. “A Survey of Selected Indoor Positioning Methods for Smartphones.” In: *IEEE Communications Surveys and Tutorials* 19.2 (Apr. 2017). Publisher: Institute of Electrical and Electronics Engineers Inc., pp. 1347–1370.

[DR07] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[Dvo19] N. Dvorecki, O. Bar-Shalom, L. Banin, and Y. Amizur. “A Machine Learning Approach for Wi-Fi RTT Ranging.” In: Jan. 2019.

[DW04] Q. Du and D. Wang. “Constrained boundary recovery for three dimensional Delaunay triangulations.” en. In: *International Journal for Numerical Methods in Engineering* 61.9 (2004). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1120, pp. 1471–1500.

[DWB06] H. Durrant-Whyte and T. Bailey. “Simultaneous Localization and Mapping: Part I.” In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[DWH16] H. Durrant-Whyte and T. C. Henderson. “Multisensor Data Fusion.” en. In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer Handbooks. Cham: Springer International Publishing, 2016, pp. 867–896.

[Ebn13] F. Ebner. “Statistische Selbstlokalisation mittels WiFi SLAM.” MA thesis. Würzburg: University of Applied Sciences Würzburg - Schweinfurt, 2013.

[Ebn19] M. Ebner. “Sensor-Based Activity Classification.” University of Applied Sciences Wuerzburg-Schweinfurt, 2019.

[Ebn21] F. Ebner. *Smartphone-Based 3D Indoor Localization and Navigation*. en. Google-Books-ID: iKsSEAAAQBAJ. Logos Verlag Berlin GmbH, Jan. 2021.

[Ebn15] F. Ebner, T. Fetzer, L. Köping, M. Grzegorzek, and F. Deinzer. “Multi Sensor 3D Indoor Localisation.” In: *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. Banff, Canada: IEEE, 2015.

[Ebn16] F. Ebner, T. Fetzer, M. Grzegorzek, and F. Deinzer. "On Prior Navigation Knowledge in Multi Sensor Indoor Localisation." In: *19th Int. Conf. on Information Fusion (FUSION)*. 2016, pp. 1–8.

[Ebn17] F. Ebner, T. Fetzer, F. Deinzer, and M. Grzegorzek. "On Wi-Fi Model Optimizations for Smartphone-Based Indoor Localization." In: *ISPRS International Journal of Geo-Information* 6.8 (2017).

[Ebn20] M. Ebner, T. Fetzer, M. Bullmann, F. Deinzer, and M. Grzegorzek. "Recognition of Typical Locomotion Activities Based on the Sensor Data of a Smartphone in Pocket or Hand." In: *Sensors* 20.22 (2020).

[Ebn22] M. Ebner, T. Fetzer, M. Bullmann, S. Kastner, F. Deinzer, and M. Grzegorzek. "PIPF: Proposal-Interpolating Particle Filter." In: *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Sept. 2022, pp. 1–8.

[EK10] K. El-Kafrawy, M. Youssef, A. El-Keyi, and A. Naguib. "Propagation Modeling for Accurate Indoor WLAN RSS-Based Localization." In: *2010 IEEE 72nd Vehicular Technology Conference - Fall*. ISSN: 1090-3038. Sept. 2010, pp. 1–5.

[Elm02] W. Elmenreich. "Sensor Fusion in Time-Triggered Systems." PhD thesis. Jan. 2002.

[Env] Environmental Systems Research Institute. *Smart Building Management | Indoor Routing, Map & App Making*. en-us. `https://www.esri.com/en-us/arcgis/products/arcgis-indoors/license-levels/maps`, Accessed: 2023-02-22.

[Eva] *Evaluating AAL Systems through Competitive Benchmarking*. `https://evaal.aaloa.org/`, Accessed: 2023-03-12.

[Fet14] T. Fetzer, F. Deinzer, L. Köping, and M. Grzegorzek. "Statistical Indoor Localization Using Fusion of Depth-Images and Step Detection." In: *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. 2014, pp. 1–9.

[Fet16] T. Fetzer, F. Ebner, F. Deinzer, L. Köping, and M. Grzegorzek. "On Monte Carlo smoothing in multi sensor indoor localisation." In: *2016 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016*. 2016.

[Fet17] T. Fetzer, F. Ebner, F. Deinzer, and M. Grzegorzek. "Recovering from sample impoverishment in context of indoor localisation." In: *2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*. Vol. 2017-Janua. 2017, pp. 1–8.

[Fet18] T. Fetzer, F. Ebner, M. Bullmann, F. Deinzer, and M. Grzegorzek. “Smartphone-Based Indoor Localization within a 13th Century Historic Building.” In: *Sensors (Basel, Switzerland)* 18.12 (2018), pp. 1–32.

[Fet22] T. Fetzer, J. Maier, M. Ebner, M. Bullmann, and F. Deinzer. “Digitales Spaghetti-Diagramm zur Laufweganalyse.” de. In: *wt Werkstattstechnik online* 112.10/2022 (Oct. 2022), pp. 727–731.

[Fet23a] T. Fetzer, M. Bullmann, M. Ebner, S. Kastner, F. Deinzer, and M. Grzegorzek. “Interacting Multiple Model Particle Filter for Indoor Positioning Applications.” en. In: *Proceedings of the 2023 International Technical Meeting of The Institute of Navigation*. Long Beach, California, 2023, pp. 1089 –1100.

[Fet23b] T. Fetzer, F. Ebner, F. Deinzer, and M. Grzegorzek. “Using Barometer for Floor Assignation within Statistical Indoor Localization.” en. In: *Sensors* 23.1 (Jan. 2023). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 80.

[FH14] R. Faragher and R. Harle. “An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications.” en. In: ISSN: 2331-5954. Sept. 2014, pp. 201–210.

[FH15] R. Faragher and R. Harle. “Location Fingerprinting With Bluetooth Low Energy Beacons.” In: *IEEE Journal on Selected Areas in Communications* 33.11 (Nov. 2015). Conference Name: IEEE Journal on Selected Areas in Communications, pp. 2418–2428.

[Fig10] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. P. Cardoso. “Preprocessing techniques for context recognition from accelerometer data.” en. In: *Personal and Ubiquitous Computing* 14.7 (Oct. 2010), pp. 645–662.

[Fin14] G. A. Fink. *Markov Models for Pattern Recognition: From Theory to Applications*. en. Springer Science & Business Media, Jan. 2014.

[Fin97] D. Fink. “A Compendium of Conjugate Priors.” en. In: (1997).

[FM94] J. Fan and J. S. Marron. “Fast Implementations of Nonparametric Curve Estimators.” In: *Journal of Computational and Graphical Statistics* 3.1 (Mar. 1994). Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/10618600.1994.10474629, pp. 35–56.

[For86] S Fortune. “A sweepline algorithm for Voronoi diagrams.” In: *Proceedings of the second annual symposium on Computational geometry*. SCG ’86. New York, NY, USA: Association for Computing Machinery, Aug. 1986, pp. 313–322.

[Fox01] D. Fox. “KLD-Sampling: Adaptive Particle Filters.” In: *Advances in Neural Information Processing Systems*. Vol. 14. MIT Press, 2001.

[Fox05] E. Foxlin. “Pedestrian Tracking with Shoe-Mounted Inertial Sensors.” en. In: *IEEE Computer Graphics and Applications* 25.6 (Nov. 2005), pp. 38–46.

[Fox99] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots.” en. In: *AAAI* (1999), pp. 343–349.

[Fri71] H. T. Friis. “Introduction to radio and radio antennas.” In: *IEEE Spectrum* 8.4 (1971), pp. 55–61.

[Fu14] J. Fu, T. Liu, M. Jones, G. Qian, and Y.-K. Jan. “Characterization of wheelchair maneuvers based on noisy inertial sensor data: A preliminary study.” In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. ISSN: 1558-4615. Aug. 2014, pp. 1731–1734.

[GA10] M. S. Grewal and A. P. Andrews. “Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives].” In: *IEEE Control Systems Magazine* 30.3 (June 2010). Conference Name: IEEE Control Systems Magazine, pp. 69–78.

[Gag17] P. A. Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. en. John Wiley & Sons, July 2017.

[Gar17] M. T. Garip, P. H. Kim, P. Reiher, and M. Gerla. “INTERLOC: An interference-aware RSSI-based localization and sybil attack detection mechanism for vehicular ad hoc networks.” In: *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. ISSN: 2331-9860. Jan. 2017, pp. 1–6.

[GC11] C. Gao and W. Chen. “Ground Moving Target Tracking with VS-IMM Using Mean Shift Unscented Particle Filter.” en. In: *Chinese Journal of Aeronautics* 24.5 (Oct. 2011), pp. 622–630.

[Gel13] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis, Third Edition*. en. CRC Press, Nov. 2013.

[Gen20] C. Gentner, M. Ulmschneider, I. Kuehner, and A. Dammann. "WiFi-RTT Indoor Positioning." In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. ISSN: 2153-3598. Apr. 2020, pp. 1029–1035.

[Geo] Geofabrik GmbH Karlsruhe. *Geofabrik Download Server*. `http://download.geofabrik.de/europe.html`, Accessed: 2023-02-22.

[Ge04] F.-X. Ge, D. Shen, Y. Peng, and V. Li. "Super-resolution time delay estimation in multipath environments." In: *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*. Vol. 2. ISSN: 1525-3511. Mar. 2004, 1121–1126 Vol.2.

[GM03] A. G. Gray and A. W. Moore. "Nonparametric Density Estimation: Toward Computational Tractability." In: *Proceedings of the 2003 SIAM International Conference on Data Mining (SDM)*. Proceedings. Society for Industrial and Applied Mathematics, May 2003, pp. 203–211.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[Goo] Google. *Android KitKat*. en.

[Goo19] Google and Open Handset Alliance. *Wi-Fi Location: ranging with RTT*. `https://developer.android.com/guide/topics/connectivity/wifi-rtt`. (Visited: 27.03.2022). 2019.

[GOP12] C. Gomez, J. Oller, and J. Paradells. "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology." In: *Sensors (Basel, Switzerland)* 12.9 (Aug. 2012), pp. 11734–11753.

[Gra17] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino. "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges." en. In: *Information Fusion* 35 (May 2017), pp. 68–80.

[GS10] F. Gustafsson and S. Saha. "Particle filtering with dependent noise." In: *2010 13th International Conference on Information Fusion*. July 2010, pp. 1–4.

[GS85] L. Guibas and J. Stolfi. "Primitives for the manipulation of general subdivisions and the computation of Voronoi." In: *ACM Transactions on Graphics* 4.2 (Apr. 1985), pp. 74–123.

[GS91] L. Greengard and J. Strain. "The Fast Gauss Transform." In: *SIAM Journal on Scientific and Statistical Computing* 12.1 (Jan. 1991). Publisher: Society for Industrial and Applied Mathematics, pp. 79–94.

[GSG06] Z. Guan, C. Song, and Y. Gu. “The boundary recovery and sliver elimination algorithms of three-dimensional constrained Delaunay triangulation.” en. In: *International Journal for Numerical Methods in Engineering* 68.2 (2006). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1707, pp. 192–209.

[Guo16] X. Guo, W. Shao, F. Zhao, Q. Wang, D. Li, and H. Luo. “WiMag: Multimode Fusion Localization System based on Magnetic/WiFi/PDR.” In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Oct. 2016, pp. 1–8.

[Guo20] X. Guo, N. Ansari, F. Hu, Y. Shao, N. R. Elikplim, and L. Li. “A Survey on Fusion-Based Indoor Positioning.” In: *IEEE Communications Surveys & Tutorials* 22.1 (2020). Conference Name: IEEE Communications Surveys & Tutorials, pp. 566–594.

[Gup15] M. Gupta, C. Holloway, B. M. Heravi, and S. Hailes. “A comparison between smartphone sensors and bespoke sensor devices for wheelchair accessibility studies.” In: *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. Apr. 2015, pp. 1–6.

[Gu19] F. Gu, X. Hu, M. Ramezani, D. Acharya, K. Khoshelham, S. Valaee, and J. Shang. “Indoor Localization Improved by Spatial Context;A Survey.” In: *ACM Computing Surveys* 52.3 (July 2019), 64:1–64:35.

[Har13] R. Harle. “A Survey of Indoor Inertial Positioning Systems for Pedestrians.” In: *IEEE Communications Surveys & Tutorials* 15.3 (2013). Conference Name: IEEE Communications Surveys & Tutorials, pp. 1281–1293.

[Har12] M. Hardegger, D. Roggen, S. Mazilu, and G. Tröster. “ActionSLAM: Using location-related actions as landmarks in pedestrian SLAM.” In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Nov. 2012, pp. 1–10.

[Hat21] E. Hatem, S. Fortes, E. Colin, S. Abou-Chakra, J.-M. Laheurte, and B. El-Hassan. “Accurate and Low-Complexity Auto-Fingerprinting for Enhanced Reliability of Indoor Localization Systems.” en. In: *Sensors* 21.16 (Jan. 2021). Number: 16 Publisher: Multidisciplinary Digital Publishing Institute, p. 5346.

[HB22] X. Hou and J. Bergmann. “HeadSLAM: Pedestrian SLAM with Head-Mounted Sensors.” en. In: *Sensors* 22.4 (Jan. 2022). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 1593.

[HC16] S. He and S.-H. G. Chan. "Wi-Fi Fingerprint-based Indoor Positioning: Recent Advances and Comparisons." In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 466–490.

[Hel13] H. Hellmers, A. Norrdine, J. Blankenbach, and A. Eichhorn. "An IMU/magnetometer-based Indoor positioning system using Kalman filtering." In: *International Conference on Indoor Positioning and Indoor Navigation*. 2013, pp. 1–9.

[Her] Here Technologies. *Data Specification - HERE Indoor Map Data Specification*. `https://developer.here.com/documentation/indoor-map-data-spec/data_spec_guide/index.html`, Accessed: 2023-02-22.

[Her21] S. Herath, S. Irandoust, B. Chen, Y. Qian, P. Kim, and Y. Furukawa. *Fusion-DHL: WiFi, IMU, and Floorplan Fusion for Dense History of Locations in Indoor Environments*. arXiv:2105.08837 [cs]. May 2021.

[Her22] S. Herath, D. Caruso, C. Liu, Y. Chen, and Y. Furukawa. "Neural Inertial Localization." en. In: 2022, pp. 6604–6613.

[HH09] D. Humphrey and M. Hedley. "Prior Models for Indoor Super-Resolution Time of Arrival Estimation." In: *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*. ISSN: 1550-2252. Apr. 2009, pp. 1–5.

[Hil14] S. Hilsenbeck, D. Bobkov, G. Schroth, R. Huitl, and E. Steinbach. "Graph-based Data Fusion of Pedometer and WiFi Measurements for Mobile Indoor Positioning." In: *Proc. of the 2014 ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*. 2014, pp. 147–158.

[HL02] C.-W. Hsu and C.-J. Lin. "A comparison of methods for multiclass support vector machines." In: *IEEE Transactions on Neural Networks* 13.2 (Mar. 2002). Conference Name: IEEE Transactions on Neural Networks, pp. 415–425.

[HM04] D. L. Hall and S. A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion*. en. Google-Books-ID: HxjDMcJWLYwC. Artech House, 2004.

[Hon09a] H. Hongwei, S. Wei, X. Youzhi, and Z. Hongke. "The effect of human activites on 2.4 GHz radio propagation at home environment." In: *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*. Oct. 2009, pp. 95–99.

[Hon09b] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piche. "A comparative survey of WLAN location fingerprinting methods." In: *Navigation and Communication 2009 6th Workshop on Positioning*. Mar. 2009, pp. 243–251.

[Hor20a] B. K. P. Horn. "Observation Model for Indoor Positioning." en. In: *Sensors* 20.14 (Jan. 2020). Number: 14 Publisher: Multidisciplinary Digital Publishing Institute, p. 4027.

[Hor20b] B. K. Horn. "Doubling the Accuracy of Indoor Positioning: Frequency Diversity." In: *Sensors* 20.5 (2020), p. 1489.

[Hor22] B. Horn. "Indoor Localization Using Uncooperative Wi-Fi Access Points." In: *Sensors* 22 (Apr. 2022), p. 3091.

[Hos14] S. A. Hoseinitabatabaei, A. Gluhak, R. Tafazolli, and W. Headley. "Design, Realization, and Evaluation of uDirect-An Approach for Pervasive Observation of User Facing Direction on Mobile Phones." In: *IEEE Transactions on Mobile Computing* 13.9 (Sept. 2014). Conference Name: IEEE Transactions on Mobile Computing, pp. 1981–1994.

[HR19] J. H. Huggins and D. M. Roy. "Sequential Monte Carlo as approximate sampling: bounds, adaptive resampling via ∞-ESS, and an application to particle Gibbs." In: *Bernoulli* 25.1 (Feb. 2019). Publisher: Bernoulli Society for Mathematical Statistics and Probability, pp. 584–622.

[HS99] J. Hershberger and S. Suri. "An Optimal Algorithm for Euclidean Shortest Paths in the Plane." In: *SIAM Journal on Computing* 28.6 (Jan. 1999). Publisher: Society for Industrial and Applied Mathematics, pp. 2215–2256.

[HSG06] J. D. Hol, T. B. Schon, and F. Gustafsson. "On Resampling Algorithms for Particle Filters." In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. Sept. 2006, pp. 79–82.

[HT97] T. Hastie and R. Tibshirani. "Classification by Pairwise Coupling." In: *Advances in Neural Information Processing Systems*. Vol. 10. MIT Press, 1997.

[Hua16] Q. Huang, Y. Zhang, Z. Ge, and C. Lu. *Refining Wi-Fi Based Indoor Localization with Li-Fi Assisted Model Calibration in Smart Buildings*. arXiv:1602.07399 [cs]. Feb. 2016.

[Hua20] L. Huang, B. Yu, H. Li, H. Zhang, S. Li, R. Zhu, and Y. Li. "HPIPS: A High-Precision Indoor Pedestrian Positioning System Fusing WiFi-RTT, MEMS, and Map Information." en. In: *Sensors* 20.23 (Jan. 2020). Number: 23 Publisher: Multidisciplinary Digital Publishing Institute, p. 6795.

[Hur02] M. Hurley. “An information theoretic justification for covariance intersection and its generalization.” In: *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*. Vol. 1. July 2002, 505–511 vol.1.

[HW96] P. Hall and M. P. Wand. “On the Accuracy of Binned Kernel Density Estimators.” In: *Journal of Multivariate Analysis* 56.2 (Feb. 1996), pp. 165–184.

[HWB00] J. Hightower, R. Want, and G. Borriello. “SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength.” en. In: (2000).

[HYH20] O. Hashem, M. Youssef, and K. A. Harras. “WiNar: RTT-based Sub-meter Indoor Localization using Commercial Devices.” In: *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. ISSN: 2474-249X. Mar. 2020, pp. 1–10.

[Hyu19] J. Hyun, T. Oh, H. Lim, and H. Myung. “UWB-based Indoor Localization Using Ray-tracing Algorithm.” In: *2019 16th International Conference on Ubiquitous Robots (UR)*. ISSN: 2325-033X. June 2019, pp. 98–101.

[HZ03] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. en. Cambridge University Press, 2003.

[IB98] M. Isard and A. Blake. “CONDENSATION - Conditional Density Propagation for Visual Tracking.” In: *Int. Journal of Computer Vision* 29.1 (1998), pp. 5–28.

[Ibr18] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, and F. Bai. “Verification: Accuracy evaluation of WiFi fine time measurements on an open platform.” In: *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM. 2018, pp. 417–427.

[Ich15] R. Ichikari, L. C. M. Ruiz, M. Kourogi, T. Kurata, T. Kitagawa, and S. Yoshii. “Indoor floor-level detection by collectively decomposing factors of atmospheric pressure.” In: *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Oct. 2015, pp. 1–11.

[Inc] A. Inc. *iBeacon*. en.

[Ish21] S. Ishii, A. Yokokubo, M. Luimula, and G. Lopez. “ExerSense: Physical Exercise Recognition and Counting Algorithm from Wearables Robust to Positioning.” en. In: *Sensors* 21.1 (Jan. 2021). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 91.

[IT07] K. Ishikawa-Takata. "Exercise and physical activity reference for health promotion 2006 (EPAR2006)." en. In: *Journal of Epidemiology* 17.5 (2007), pp. 177–177.

[Jah10] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka, and J. Gutiérrez Boronat. "Comparison and evaluation of acceleration based step length estimators for handheld devices." In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. Sept. 2010, pp. 1–6.

[Jan08] b.-j. Jang, S.-H Wi, J.-G. Yook, and M.-Q. Lee. "Wireless bio-radar sensor for heartbeat and respiration detection." In: *Prog Electromagn Res (PIER) C* 5 (Jan. 2008).

[Jaw17] W. Jaworski, P. Wilk, P. Zborowski, W. Chmielowiec, A. Y. G. Lee, and A. Kumar. "Real-time 3D indoor localization." In: *2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1–8.

[Jay68] E. T. Jaynes. "Prior Probabilities." In: *IEEE Transactions on Systems Science and Cybernetics* 4.3 (Sept. 1968). Conference Name: IEEE Transactions on Systems Science and Cybernetics, pp. 227–241.

[JD08] A. M. Johansen and A. Doucet. "A note on auxiliary particle filters." en. In: *Statistics & Probability Letters* 78.12 (Sept. 2008), pp. 1498–1504.

[Jef46] H. Jeffreys. "An Invariant Form for the Prior Probability in Estimation Problems." In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 186.1007 (1946), pp. 453–461.

[Jia16] Q. Jiang, Y. Ma, K. Liu, and Z. Dou. "A Probabilistic Radio Map Construction Scheme for Crowdsourcing-Based Fingerprinting Localization." In: *IEEE Sensors Journal* 16.10 (May 2016). Conference Name: IEEE Sensors Journal, pp. 3764–3774.

[Jio20] K. Jiokeng, G. Jakllari, A. Tchana, and A.-L. Beylot. "When FTM Discovered MUSIC: Accurate WiFi-based Ranging in the Presence of Multipath." In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. ISSN: 2641-9874. July 2020, pp. 1857–1866.

[JK17] P. Jain and P. Kar. "Non-convex Optimization for Machine Learning." In: *Foundations and Trends® in Machine Learning* 10.3-4 (2017). arXiv:1712.07897 [cs, math, stat], pp. 142–336.

[Jor19] A. Jordao, A. C. Nazare Jr., J. Sena, and W. R. Schwartz. *Human Activity Recognition Based on Wearable Sensor Data: A Standardization of the State-of-the-Art*. en. arXiv:1806.05226 [cs]. Feb. 2019.

[JU04] S. Julier and J. Uhlmann. "Unscented filtering and nonlinear estimation." In: *Proceedings of the IEEE* 92.3 (Mar. 2004). Conference Name: Proceedings of the IEEE, pp. 401–422.

[JU97] S. Julier and J. Uhlmann. "A non-divergent estimation algorithm in the presence of unknown correlations." In: *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*. Vol. 4. ISSN: 0743-1619. June 1997, 2369–2373 vol.4.

[KA20] I. Klein and O. Asraf. "StepNet—Deep Learning Approaches for Step Length Estimation." In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 85706–85713.

[Kal10] M. Kallmann. "Shortest Paths with Arbitrary Clearance from Navigation Meshes." en. In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2010). Artwork Size: 10 pages ISBN: 9783905674279 Publisher: The Eurographics Association, 10 pages.

[Kal60] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45.

[Kan20] Z. Kang, J. Yang, Z. Yang, and S. Cheng. "A Review of Techniques for 3D Reconstruction of Indoor Environments." en. In: *ISPRS International Journal of Geo-Information* 9.5 (May 2020). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 330.

[KBT04] M. Kallmann, H. Bieri, and D. Thalmann. "Fully Dynamic Constrained Delaunay Triangulations." en. In: *Geometric Modeling for Scientific Visualization*. Ed. by G. Farin, H.-C. Hege, D. Hoffman, C. R. Johnson, K. Polthier, G. Brunnett, B. Hamann, H. Müller, and L. Linsen. Series Title: Mathematics and Visualization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 241–257.

[KF98] D. Koller and R. Fratkina. "Using learning for approximation in stochastic processes." en. In: *International Conference on Machine Learning* (1998).

[KGA17] A. Khalajmehrabadi, N. Gatsis, and D. Akopian. "Modern WLAN Fingerprinting Indoor Positioning Methods and Deployment Challenges." In: *IEEE Communications Surveys & Tutorials* 19.3 (2017). Conference Name: IEEE Communications Surveys & Tutorials, pp. 1974–2002.

[KGD14] L. Koeping, M. Grzegorzek, and F. Deinzer. "Probabilistic Step and Turn Detection in Indoor Localization." In: *Conf. on Data Fusion and Target Tracking 2014: Algorithms and Applications (DFTT 2014)*. Liverpool, UK, 2014, pp. 1–7.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing." In: *SCIENCE* 220.4598 (1983), pp. 671–680.

[Kha10] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim. "Human Activity Recognition via an Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis." In: *2010 5th International Conference on Future Information Technology*. ISSN: 2159-7014. May 2010, pp. 1–6.

[Kha13a] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. "Multisensor data fusion: A review of the state-of-the-art." en. In: *Information Fusion* 14.1 (Jan. 2013), pp. 28–44.

[Kha13b] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. "Multisensor data fusion: A review of the state-of-the-art." In: *Information Fusion* 14.1 (2013), pp. 28–44.

[Kie] M. Kiers, E. Krajnc, M. Dornhofer, and W. Bischof. "Evaluation and Improvements of an RFID Based Indoor Navigation System for Visually Impaired and Blind People." en. In: ().

[Kit96] G. Kitagawa. "Monte Carlo Filter and Smoother for non-Gaussian nonlinear State Space Models." In: *Journal of Computational and Graphical Statistics* 5.1 (1996), pp. 1–25.

[Kjæ10] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk. "Indoor positioning using GPS revisited." In: *Proceedings of the 8th international conference on Pervasive Computing*. Pervasive'10. Berlin, Heidelberg: Springer-Verlag, May 2010, pp. 38–56.

[KK12] K. Kaemarungsi and P. Krishnamurthy. "Analysis of WLAN's received signal strength indication for indoor location fingerprinting." en. In: *Pervasive and Mobile Computing*. Special Issue: Wide-Scale Vehicular Sensor Networks and Mobile Sensing 8.2 (Apr. 2012), pp. 292–316.

[KL16] S. Kaiser and C. Lang. "Detecting elevators and escalators in 3D pedestrian indoor navigation." In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Oct. 2016, pp. 1–6.

[KMK16] A. Keshavarz-Mohammadiyan and H. Khaloozadeh. “Adaptive IMMPF for bearing-only maneuvering target tracking in Wireless Sensor networks.” In: *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*. Jan. 2016, pp. 6–11.

[Kna17] S. Knauth. “Smartphone PDR positioning in large environments employing WiFi, particle filter, and backward optimization.” In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Sept. 2017, pp. 1–6.

[Köp14] L. Köping, F. Ebner, M. Grzegorzek, and F. Deinzer. “Indoor Localization Using Step and Turn Detection Together with Floor Map Information.” In: *FHWS Science Journal* (2014).

[Kün05] H. R. Künsch. “Recursive Monte Carlo Filters: Algorithms and Theoretical Analysis.” In: *Ann. of Statistics* (2005), pp. 1983–2021.

[Kuo14a] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta. “Luxapose: indoor positioning with mobile phones and visible light.” In: *Proceedings of the 20th annual international conference on Mobile computing and networking*. MobiCom ’14. New York, NY, USA: Association for Computing Machinery, Sept. 2014, pp. 447–458.

[Kuo14b] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta. “Luxapose: Indoor Positioning with Mobile Phones and Visible Light.” In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. ACM. 2014, pp. 447–458.

[Lad05] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach. “Robotics-Based Location Sensing Using Wireless Ethernet.” en. In: *Wireless Networks* 11.1 (Jan. 2005), pp. 189–204.

[Lai19] D. Laidig, D. Lehmann, M.-A. Bégin, and T. Seel. “Magnetometer-free real-time inertial motion tracking by exploitation of kinematic constraints in 2-dof joints.” In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2019, pp. 1233–1238.

[LBD15] T. Li, M. Bolic, and P. M. Djuric. “Resampling Methods for Particle Filtering: Classification, implementation, and strategies.” en. In: *IEEE Signal Processing Magazine* 32.3 (May 2015), pp. 70–86.

[LCH20] S. Lee, S. Chae, and D. Han. "ILoA: Indoor Localization Using Augmented Vector of Geomagnetic Field." In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 184242–184255.

[LCL01] J. S. Liu, R. Chen, and T. Logvinenko. "A Theoretical Framework for Sequential Importance Sampling with Resampling." en. In: *Sequential Monte Carlo Methods in Practice*. Ed. by A. Doucet, N. Freitas, and N. Gordon. New York, NY: Springer New York, 2001, pp. 225–246.

[LCR10] X. Li, C. Claramunt, and C. Ray. "A grid graph-based model for the analysis of 2D indoor spaces." In: *Computers, Environment and Urban Systems* 34.6 (2010), pp. 532–540.

[Leh20] D. Lehmann, D. Laidig, R. Deimel, and T. Seel. "Magnetometer-free inertial motion tracking of arbitrary joints with range of motion constraints." In: *IFAC-PapersOnLine* 53.2 (2020), pp. 16016–16022.

[LHG13] B. Li, B. Harvey, and T. Gallagher. "Using Barometers to Determine the Height for Indoor Positioning." In: *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. 2013, pp. 1–7.

[Lia16] J. Liang, D. Wang, L. Su, B. Chen, H. Chen, and H. C. So. "Robust MIMO radar target localization via nonconvex optimization." en. In: *Signal Processing* 122 (May 2016), pp. 33–38.

[Li14] T. Li, S. Sun, T. P. Sattar, and J. M. Corchado. "Fight Sample Degeneracy and Impoverishment in Particle Filters: A Review of Intelligent Approaches." In: *Expert Systems with Applications* 41.8 (2014), pp. 3944–3954.

[Li20] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari. "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments." In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020). Conference Name: IEEE Robotics and Automation Letters, pp. 6583–6590.

[Liu07] H. Liu, H. Darabi, P. Banerjee, and J. Liu. "Survey of Wireless Indoor Positioning Techniques and Systems." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (Nov. 2007). Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), pp. 1067–1080.

[LL17a] D. Lymberopoulos and J. Liu. "Microsoft Indoor Localization Competition: Experiences and Lessons Learned." In: *IEEE Signal Processing Magazine* 34.5 (2017), pp. 125–140.

[LL17b] D. Lymberopoulos and J. Liu. “The Microsoft Indoor Localization Competition: Experiences and Lessons Learned.” In: *IEEE Signal Processing Magazine* 34.5 (Sept. 2017). Conference Name: IEEE Signal Processing Magazine, pp. 125–140.

[LLF19] E. Lewandowicz, P. Lisowski, and P. Flisek. “A Modified Methodology for Generating Indoor Navigation Models.” en. In: *ISPRS International Journal of Geo-Information* 8.2 (Feb. 2019). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 60.

[LP04] X. Li and K. Pahlavan. “Super-resolution TOA estimation with diversity for indoor geolocation.” In: *IEEE Transactions on Wireless Communications* 3.1 (Jan. 2004). Conference Name: IEEE Transactions on Wireless Communications, pp. 224–234.

[LSS12] T. Li, T. P. Sattar, and S. Sun. “Deterministic resampling: Unbiased sampling to avoid sample impoverishment in particle filters.” en. In: *Signal Processing* 92.7 (July 2012), pp. 1637–1645.

[LSS13] T. Li, S. Sun, and T. P. Sattar. “Adapting sample size in particle filters through KLD-resampling.” en. In: *Electronics Letters* 49.12 (June 2013). Publisher: IET Digital Library, pp. 740–742.

[LY13] M. A. Labrador and O. D. L. Yejas. *Human Activity Recognition: Using Wearable Sensors and Smartphones*. en. Google-Books-ID: xyMtAgAAQBAJ. CRC Press, Dec. 2013.

[Mad10] S. Madgwick et al. “An efficient orientation filter for inertial and inertial/magnetic sensor arrays.” In: *Report x-io and University of Bristol (UK)* 25 (2010), pp. 113–118.

[Mar14] A. Marcaletti, M. Rea, D. Giustiniano, V. Lenders, and A. Fakhreddine. “Filtering Noisy 802.11 Time-of-Flight Ranging Measurements.” In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. CoNEXT ’14. New York, NY, USA: Association for Computing Machinery, Dec. 2014, pp. 13–20.

[Mar18] A. Martinelli, H. Gao, P. D. Groves, and S. Morosi. “Probabilistic Context-Aware Step Length Estimation for Pedestrian Dead Reckoning.” In: *IEEE Sensors Journal* 18.4 (Feb. 2018). Conference Name: IEEE Sensors Journal, pp. 1600–1611.

[Mas17] H. Masuyama. *Binary sampling from discrete distributions*. en. arXiv:1704.07012 [math, stat]. June 2017.

[Mas20] S. Mascetti, G. Civitarese, O. El Malak, and C. Bettini. "SmartWheels: Detecting urban features for wheelchair users' navigation." en. In: *Pervasive and Mobile Computing* 62 (Feb. 2020), p. 101115.

[Ma22] C. Ma, B. Wu, S. Poslad, and D. R. Selviah. "Wi-Fi RTT Ranging Performance Characterization and Positioning System Design." In: *IEEE Transactions on Mobile Computing* 21.02 (2022), pp. 740–756.

[Mau12] R. Mautz. "Indoor Positioning Technologies." PhD thesis. Habilitationsschrift ETH Zürich, 2012, 2012.

[MBG13] M. Mofarreh-Bonab and S. A. Ghorashi. "A low complexity and high speed gradient descent based secure localization in Wireless Sensor Networks." In: *3th Int. eConf. on Computer and Knowledge Engineering (ICCKE)*. 2013, pp. 300–303.

[MBH17] A. Moretto and I. Borras Hernandez. "Indoor positioning through fingerprinting technics: How many beacons should be deployed and where?" In: *2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*. ISSN: 1882-5621. Dec. 2017, pp. 522–528.

[MC13] M. Mononen and Contributor. *Recast Navigation*. `https://github.com/recastnavigation/recastnavigation`. (Visited: 19.07.2022). 2013.

[Meh17] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. "VNect: real-time 3D human pose estimation with a single RGB camera." In: *ACM Transactions on Graphics* 36.4 (July 2017), 44:1–44:14.

[Meh20] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt. "XNect: real-time multi-person 3D motion capture with a single RGB camera." In: *ACM Transactions on Graphics* 39.4 (Aug. 2020), 82:82:1–82:82:17.

[MEK05] S. McLaughlin, R. Evans, and V. Krishnamurthy. "A graph theoretic approach to data incest management in network centric warfare." In: *2005 7th International Conference on Information Fusion*. Vol. 2. July 2005, 8 pp.–.

[Meu92] G. Meurant. *Data Fusion in Robotics & Machine Intelligence*. en. Google-Books-ID: 47kOwU1xvMMC. Academic Press, Oct. 1992.

[MHV11] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. “Estimation of IMU and MARG orientation using a gradient descent algorithm.” In: *2011 IEEE International Conference on Rehabilitation Robotics*. ISSN: 1945-7901. June 2011, pp. 1–7.

[Mik99] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers. “Fisher discriminant analysis with kernels.” en. In: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*. Madison, WI, USA: IEEE, 1999, pp. 41–48.

[Mir14] P. Mirowski, D. Milioris, P. Whiting, and T. K. Ho. “Probabilistic radio-frequency fingerprinting and localization on the run.” In: *Bell Labs Technical Journal* 18.4 (Mar. 2014). Conference Name: Bell Labs Technical Journal, pp. 111–133.

[Mit93] J. S. B. Mitchell. “Shortest paths among obstacles in the plane.” In: *Proceedings of the ninth annual symposium on Computational geometry*. SCG ’93. New York, NY, USA: Association for Computing Machinery, July 1993, pp. 308–317.

[MK01] A. Martinez and A. Kak. “PCA versus LDA.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.2 (Feb. 2001). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 228–233.

[MKS] D. Martin, N. Kühl, and G. Satzger. “Virtual Sensors.” In: *Business & Information Systems Engineering* 63 ().

[Möh] H. Möhring. *RothenburgMuseum*. `https://rothenburgmuseum.de`, Accessed: 2023-09-22. Rothenburg, Germany.

[Mos13] A. Moschevikin, A. Galov, A. Soloviev, A. Mikov, A. Volkov, and S. Reginya. “Realtrac Technology Overview.” In: *International Competition on Evaluating AAL Systems through Competitive Benchmarking*. Springer. 2013, pp. 60–71.

[MPS14] L. Mainetti, L. Patrono, and I. Sergi. “A Survey on Indoor Positioning Systems.” In: *Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on*. IEEE. 2014, pp. 111–120.

[MSZ99] E. P. Mücke, I. Saias, and B. Zhu. “Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations.” en. In: *Computational Geometry* (1999).

[MT11] R. Mautz and S. Tilch. “Survey of Optical Indoor Positioning Systems.” In: *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. IEEE. 2011, pp. 1–7.

[MUH04] S. Müller, M. Uchanski, and K. Hedrick. “Estimation of the Maximum Tire-Road Friction Coefficient.” In: *Journal of Dynamic Systems, Measurement, and Control* 125.4 (Jan. 2004), pp. 607–617.

[Mur12] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. English. Illustrated edition. Cambridge, MA: The MIT Press, Aug. 2012.

[Mur14] K. Muralidharan, A. J. Khan, A. Misra, R. K. Balan, and S. Agarwal. “Barometric Phone Sensors: More Hype Than Hope!” In: *Proc. of the 15th Workshop on Mobile Computing Systems and Applications*. Santa Barbara, California, 2014, 12:1–12:6.

[Mus97] O. R. Musin. “Properties of the Delaunay triangulation.” en. In: *Proceedings of the thirteenth annual symposium on Computational geometry - SCG ’97*. Nice, France: ACM Press, 1997, pp. 424–426.

[MYNK09] F. Mohd-Yasin, D. J. Nagel, and C. E. Korman. “Noise in MEMS.” en. In: *Measurement Science and Technology* 21.1 (Nov. 2009), p. 012001.

[Ni03] L. Ni, Y. Liu, Y. C. Lau, and A. Patil. “LANDMARC: indoor location sensing using active RFID.” In: *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).* Mar. 2003, pp. 407–415.

[Noaa] *Eddystone*. original-date: 2015-06-26T10:58:02Z. Aug. 2022.

[Noab] *IEEE802*.

[Noac] *Product Finder Results | Wi-Fi Alliance*.

[Not18] M. Noto, F. Shang, S. Kidera, and T. Kirimoto. “Super-Resolution Time of Arrival Estimation Using Random Resampling in Compressed Sensing.” In: *IEICE Transactions on Communications* E101.B.6 (2018), pp. 1513–1520.

[Nur13] H. Nurminen, A. Ristimaki, S. Ali-Loytty, and R. Piche. “Particle Filter and Smoother for Indoor Localization.” In: *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. 2013, pp. 1–10.

[Nur14] H. Nurminen, M. Koivisto, S. Ali-Loytty, and R. Piche. “Motion Model for Positioning with Graph-based Indoor Map.” In: *Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014, pp. 646–655.

[OAM22] G. Ouyang and K. Abed-Meraim. “A Survey of Magnetic-Field-Based Indoor Localization.” en. In: *Electronics* 11.6 (Jan. 2022). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 864.

[O’B16] T. A. O’Brien, K. Kashinath, N. R. Cavanaugh, W. D. Collins, and J. P. O’Brien. “A fast and objective multidimensional kernel density estimation method: fastKDE.” In: *Computational Statistics and Data Analysis* 101 (Sept. 2016), pp. 148–160.

[Oeh09] V. Oehler, J. M. Krueger, T. Beck, M. Kirchner, H. L. Trautenberg, J. Hahn, and D. Blonski. “Galileo System Performance Status Report.” en. In: Sept. 2009, pp. 2956–2966.

[OL76] A. O’HAGAN and T. LEONARD. “Bayes estimation subject to uncertainty about parameter constraints.” In: *Biometrika* 63.1 (Jan. 1976), pp. 201–203.

[Ope] OpenStreetMap Foundation. *Simple Indoor Tagging - OpenStreetMap Wiki*. `https://wiki.openstreetmap.org/wiki/Simple_Indoor_Tagging`, Accessed: 2023-02-22.

[Orh12] E. Orhan. *Particle Filtering*. `http://www.cns.nyu.edu/~eorhan/` (Visited: 27.11.2015). Technical Report. 2012.

[Oti20] T. Otim, A. Bahillo, L. E. Díez, P. Lopez-Iturri, and F. Falcone. “Towards Sub-Meter Level UWB Indoor Localization Using Body Wearable Sensors.” In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 178886–178899.

[PB84] H. A. P. Blom. “An efficient filter for abruptly changing systems.” In: *The 23rd IEEE Conference on Decision and Control*. Dec. 1984, pp. 656–658.

[Ped] M. Pedley. “High-Precision Calibration of a Three-Axis Accelerometer.” en. In: (), p. 35.

[Pel16] N. Pelechano, J. M. Allbeck, M. Kapadia, and N. I. Badler. *Simulating Heterogeneous Crowds with Interactive Behaviors*. en. Google-Books-ID: 2yMNDgAAQBAJ. CRC Press, Oct. 2016.

[PGH17] J. Powar, C. Gao, and R. Harle. “Assessing the impact of multi-channel BLE beacons on fingerprint-based positioning.” In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Sept. 2017, pp. 1–8.

[Pip19] G. Pipelidis, N. Tsiamitros, C. Gentner, D. B. Ahmed, and C. Prehofer. "A Novel Lightweight Particle Filter for Indoor Localization." In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Sept. 2019, pp. 1–8.

[PKC08] J. Parviainen, J. Kantola, and J. Collin. "Differential barometry in personal navigation." In: *Position, Location and Navigation Symp., 2008 IEEE/ION*. 2008, pp. 148–152.

[PL15] M.-S. Pan and H.-W. Lin. "A Step Counting Algorithm for Smartphone Users: Design and Implementation." In: *IEEE Sensors Journal* 15.4 (Apr. 2015). Conference Name: IEEE Sensors Journal, pp. 2296–2305.

[PLP21] S. Park, J. H. Lee, and C. G. Park. "Robust Pedestrian Dead Reckoning for Multiple Poses in Smartphones." In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 54498–54508.

[Pot17a] F. Potortì, S. Park, A. R. Jiménez Ruiz, P. Barsocchi, M. Girolami, A. Crivello, S. Y. Lee, J. H. Lim, J. Torres-Sospedra, F. Seco, R. Montoliu, G. M. Mendoza-Silva, M. D. C. Pérez Rubio, C. Losada-Gutiérrez, F. Espinosa, and J. Macias-Guarasa. "Comparing the Performance of Indoor Localization Systems through the EvAAL Framework." en. In: *Sensors* 17.10 (Oct. 2017). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 2327.

[Pot17b] F. Potortì, S. Park, A. R. Jiménez Ruiz, P. Barsocchi, M. Girolami, A. Crivello, S. Y. Lee, J. H. Lim, J. Torres-Sospedra, F. Seco, R. Montoliu, G. M. Mendoza-Silva, M. D. C. Pérez Rubio, C. Losada-Gutiérrez, F. Espinosa, and J. Macias-Guarasa. "Comparing the Performance of Indoor Localization Systems through the EvAAL Framework." In: *Sensors* 17.10 (2017).

[Pre09] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard. "A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data." en. In: *IEEE Transactions on Biomedical Engineering* 56.3 (Mar. 2009), pp. 871–879.

[QZL14] T. Qiao, Y. Zhang, and H. Liu. "Nonlinear Expectation Maximization Estimator for TDOA Localization." In: *IEEE Wireless Communications Letters* 3.6 (Dec. 2014). Conference Name: IEEE Wireless Communications Letters, pp. 637–640.

[Rab02] S. Rabin. *AI game programming wisdom*. eng. Hingham, MA : Charles River Media, 2002.

[Rab19] S. Rabin. *Game AI Pro 360: Guide to Movement and Pathfinding*. en. Google-Books-ID: bTX3DwAAQBAJ. CRC Press, Sept. 2019.

[Rac11] S. A. Rackley. *Wireless Networking Technology: From Principles to Successful Implementation*. en. Elsevier, Feb. 2011.

[Rac16] J. Racko, P. Brida, A. Perttula, J. Parviainen, and J. Collin. "Pedestrian Dead Reckoning with Particle Filter for handheld smartphone." In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. 2016, pp. 1–7.

[Raj16] N. Rajagopal, S. Chayapathy, B. Sinopoli, and A. Rowe. "Beacon placement for range-based indoor localization." In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. ISSN: 2471-917X. Oct. 2016, pp. 1–8.

[Rap96] T. S. Rappaport et al. *Wireless communications: principles and practice*. Vol. 2. prentice hall PTR New Jersey, 1996.

[RC13] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. en. Springer Science & Business Media, Mar. 2013.

[RC14] V. Renaudin and C. Combettes. "Magnetic, Acceleration Fields and Gyroscope Quaternion (MAGYQ)-Based Attitude Estimation with Smartphone Sensors for Indoor Pedestrian Navigation." en. In: *Sensors* 14.12 (Dec. 2014). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, pp. 22864–22890.

[Rek04] I. M. Rekleitis. "A Particle Filter Tutorial for Mobile Robot Localization." In: *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02* (2004).

[RM12] S. Robitzsch and L. Murphy. "Empirical analysis of measured 802.11 Receive Signal Strength Values using various Atheros based Mini-PCI cards." In: *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2012, pp. 1–6.

[Rog99] L. Rognant, J. Chassery, S. Goze, and J. Planes. "The Delaunay constrained triangulation: the Delaunay stable algorithms." In: *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*. ISSN: 1093-9547. July 1999, pp. 147–152.

[RWRC14] N. Roy, H. Wang, and R. Roy Choudhury. "I am a smartphone and i can tell my user's walking direction." In: *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. MobiSys '14. New York, NY, USA: Association for Computing Machinery, June 2014, pp. 329–342.

[SB21] C. Sanchez-Belenguer, E. Wolfart, A. Casado-Coscolla, and V. Sequeira. "RISEdb: a Novel Indoor Localization Dataset." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. ISSN: 1051-4651. Jan. 2021, pp. 9514–9521.

[Seb10] M. T. Sebastian. *Dielectric Materials for Wireless Communication*. en. Google-Books-ID: eShDR4_YyM8C. Elsevier, July 2010.

[SG12] S. Saha and F. Gustafsson. "Particle Filtering With Dependent Noise Processes." In: *IEEE Transactions on Signal Processing* 60.9 (Sept. 2012). Conference Name: IEEE Transactions on Signal Processing, pp. 4497–4508.

[SGM13] I. M. Smith, D. V. Griffiths, and L. Margetts. *Programming the Finite Element Method*. en. Google-Books-ID: ZbtiAAAAQBAJ. John Wiley & Sons, Nov. 2013.

[She96] J. R. Shewchuk. "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator." en. In: *Applied Computational Geometry Towards Geometric Engineering*. Ed. by M. C. Lin and D. Manocha. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1996, pp. 203–222.

[Sil82] B. W. Silverman. "Algorithm AS 176: Kernel Density Estimation Using the Fast Fourier Transform." In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.1 (1982). Publisher: [Wiley, Royal Statistical Society], pp. 93–99.

[sim] simpleLoc. *GitHub - simpleLoc/IndoorMapParser: C++ parser for indoor XML map format*. `https://github.com/simpleLoc/IndoorMapParser`, Accessed: 2023-02-22.

[SJ18] F. Seco and A. R. Jiménez. "Smartphone-Based Cooperative Indoor Localization with RFID Technology." In: *Sensors 2018, Vol. 18, Page 266* 18.1 (2018), p. 266.

[SKG16] K. Shirahama, L. Köping, and M. Grzegorzek. “Codebook approach for sensor-based human activity recognition.” en. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. Heidelberg Germany: ACM, Sept. 2016, pp. 197–200.

[SMS23] N. Suga, Y. Maeda, and K. Sato. “Indoor Radio Map Construction via Ray Tracing With RGB-D Sensor-Based 3D Reconstruction: Concept and Experiments in WLAN Systems.” In: *IEEE Access* 11 (2023). Conference Name: IEEE Access, pp. 24863–24874.

[SNH10] I. Skog, J.-O. Nilsson, and P. Händel. “Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems.” In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. Sept. 2010, pp. 1–6.

[Sol16] A. Solin, S. Särkkä, J. Kannala, and E. Rahtu. “Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning.” In: *2016 European Navigation Conference (ENC)*. May 2016, pp. 1–9.

[Sol20] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar. “A Review of Radio Frequency Fingerprinting Techniques.” In: *IEEE Journal of Radio Frequency Identification* 4.3 (Sept. 2020). Conference Name: IEEE Journal of Radio Frequency Identification, pp. 222–233.

[SP20] P. Spachos and K. N. Plataniotis. “BLE Beacons for Indoor Positioning at an Interactive IoT-Based Smart Museum.” In: *IEEE Systems Journal* 14.3 (Sept. 2020). Conference Name: IEEE Systems Journal, pp. 3483–3493.

[SR92] S. Y. Seidel and T. S. Rappaport. “914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings.” In: *IEEE Transactions on Antennas and Propagation* 40.2 (1992), pp. 207–217.

[SS10] U. Steinhoff and B. Schiele. “Dead reckoning from the pocket - An experimental study.” In: *2010 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Mar. 2010, pp. 162–170.

[Ste] Steinbeis Transferzentrum New Media and Data Science. *simpleLoc*.

[STW07] C. Shan, T. Tan, and Y. Wei. “Real-time hand tracking using a mean shift embedded particle filter.” en. In: *Pattern Recognition* 40.7 (July 2007), pp. 1958–1970.

[Sun10] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li. "Activity Recognition on an Accelerometer Embedded Mobile Phone with Varying Positions and Orientations." en. In: *Ubiquitous Intelligence and Computing*. Ed. by Z. Yu, R. Liscano, G. Chen, D. Zhang, and X. Zhou. Vol. 6406. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 548–562.

[SVW22] T. Savić, X. Vilajosana, and T. Watteyne. "Constrained Localization: A Survey." In: *IEEE Access* 10 (2022). Conference Name: IEEE Access, pp. 49297–49321.

[Sze22] R. Szeliski. *Computer Vision: Algorithms and Applications*. en. Google-Books-ID: QptXEAAAQBAJ. Springer Nature, Jan. 2022.

[Sä13] S. Särkkä. *Bayesian Filtering and Smoothing*. en. 1st ed. Cambridge University Press, Sept. 2013.

[Tan11] K. Tan, D. Wu, A. J. Chan, and P. Mohapatra. "Comparing simulation tools and experimental testbeds for wireless mesh networks." en. In: *Pervasive and Mobile Computing* 7.4 (Aug. 2011), pp. 434–448.

[Tay09] A. Tayebi, J. Gomez Perez, F. M. S. d. Adana Herrero, and O. Gutierrez Blanco. "The Application of Ray-Tracing to Mobile Localization using the Direction of Arrival and Received Signal Strength in Multipath Indoor Environments." In: *Progress In Electromagnetics Research* 91 (2009), pp. 1–15.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge, MA, USA: The MIT Press, 2005.

[Thr03] S. Thrun. "Learning Occupancy Grid Maps with Forward Sensor Models." en. In: *Autonomous Robots* 15.2 (2003), pp. 111–127.

[Tor84] D. J. Torrieri. "Statistical Theory of Passive Location Systems." In: *IEEE Transactions on Aerospace and Electronic Systems* AES-20.2 (Mar. 1984). Conference Name: IEEE Transactions on Aerospace and Electronic Systems, pp. 183–198.

[Tri21] H. K. Tripathy, S. Mishra, H. K. Thakkar, and D. Rai. "CARE: A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search." en. In: *Computers & Electrical Engineering* 94 (Sept. 2021), p. 107327.

[TS17] J. Torres-Sospedra, A. R. Jiménez, S. Knauth, A. Moreira, Y. Beer, T. Fetzer, V. C. Ta, R. Montoliu, F. Seco, G. M. Mendoza-Silva, O. Belmonte, A. Koukofikis, M. J. Nicolau, A. Costa, F. Meneses, F. Ebner, F. Deinzer, D. Vaufreydaz, T. K. Dao, and E. Castelli. "The smartphone-based offline indoor location competition at IPIN 2016: Analysis and future work." In: *Sensors (Switzerland)* 17.3 (2017).

[TS21] J. Torres-Sospedra, F. A. Polo, F. Parralejo, V. B. Parent, F. Alvarez, A. Pérez-Navarro, A. R. Jimenez, and F. Seco. *Datasets and Supporting Materials for the IPIN 2021 Competition Track 3 (Smartphone-based, off-site)*. Dec. 2021.

[TUI17] T. Taketomi, H. Uchiyama, and S. Ikeda. "Visual SLAM algorithms: a survey from 2010 to 2016." In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (June 2017), p. 16.

[TV09] P. Teunissen and S. Verhagen. "GNSS Carrier Phase Ambiguity Resolution: Challenges and Open Problems." en. In: *Observing our Changing Earth*. Ed. by M. G. Sideris. International Association of Geodesy Symposia. Berlin, Heidelberg: Springer, 2009, pp. 785–792.

[TZ11] D. Tang and Y.-J. Zhang. "Combining Mean-Shift and Particle Filter for Object Tracking." In: *2011 Sixth International Conference on Image and Graphics*. Aug. 2011, pp. 771–776.

[Vav16] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis. "The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones:" in: *Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health*. Rome, Italy: SCITEPRESS - Science, and Technology Publications, 2016, pp. 143–151.

[VJ19] M. Vezočnik and M. B. Juric. "Average Step Length Estimation Models' Evaluation Using Inertial Sensors: A Review." In: *IEEE Sensors Journal* 19.2 (Jan. 2019). Conference Name: IEEE Sensors Journal, pp. 396–403.

[VN20] D. Van Nguyen, T. Van Pham, A. Van Tran, K. NguyenTuan, H. Duong ThiThuy, H. Hoang The, and T. Tran Duc. "Elevator Motion States Recognition Using Barometer Support Indoor Positioning System." en. In: *7th International Conference on the Development of Biomedical Engineering in Vietnam (BME7)*. Ed. by V. Van Toi, T. Q. Le, H. T. Ngo, and T.-H. Nguyen. Vol. 69. Series Title: IFMBE Proceedings. Singapore: Springer Singapore, 2020, pp. 499–504.

[Wan94] M. P. Wand. "Fast Computation of Multivariate Kernel Estimators." In: *Journal of Computational and Graphical Statistics* 3.4 (Dec. 1994). Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/10618600.1994.10474656, pp. 433–445.

[Wan06] H. Wang, H. Lenz, A. Szabo, U. D. Hanebeck, and J. Bamberger. "Fusion of barometric sensors, WLAN signals and building information for 3-D indoor/-campus localization." In: *Proc. of Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI 2006), S*. 2006, pp. 426–432.

[Wan12] X. Wang, M. Xu, H. Wang, Y. Wu, and H. Shi. "Combination of Interacting Multiple Models with the Particle Filter for Three-Dimensional Target Tracking in Underwater Wireless Sensor Networks." en. In: *Mathematical Problems in Engineering* 2012 (Dec. 2012). Publisher: Hindawi, e829451.

[Wan22] Q. Wang, M. Fu, J. Wang, H. Luo, L. Sun, Z. Ma, W. Li, C. Zhang, R. Huang, X. Li, Z. Jiang, and Q. Liang. "Recent Advances in Pedestrian Inertial Navigation Based on Smartphone: A Review." In: *IEEE Sensors Journal* 22.23 (Dec. 2022). Conference Name: IEEE Sensors Journal, pp. 22319–22343.

[Wil02] R. Wilson. *Propagation Losses Through Common Building Materials 2.4 GHz vs 5 GHz.* last accessed 15.06.2015. 2002.

[Wil11] R. R. Wilcox. *Introduction to Robust Estimation and Hypothesis Testing*. en. Google-Books-ID: 8f8nBb4__EYC. Academic Press, Dec. 2011.

[Wis19] J. Wisanmongkol, L. Klinkusoom, T. Sanpechuda, L.-o. Kovavisaruch, and K. Kaemarungsi. "Multipath Mitigation for RSSI-Based Bluetooth Low Energy Localization." In: *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*. ISSN: 2643-6175. Sept. 2019, pp. 47–51.

[WKB08] Widyawan, M. Klepal, and S. Beauregard. "A Backtracking Particle Filter for fusing building plans with PDR displacement estimates." In: *Navigation and Communication 2008 5th Workshop on Positioning*. Mar. 2008, pp. 207–212.

[WKM11] M. Werner, M. Kessel, and C. Marouane. "Indoor Positioning using Smartphone Camera." In: *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. IEEE. 2011, pp. 1–6.

[WKS15] T. Willemsen, F. Keller, and H. Sternberg. "A Topological Approach with MEMS in Smartphones based on Routing-Graph." In: *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. 2015, pp. 1–6.

[WL90] E. Waltz and J. Llinas. *Multisensor data fusion*. en. Artech House radar library. Boston: Artech House, 1990.

[WLG18] P. Wang, L. Li, and S. J. Godsill. "Particle Filtering and Inference for Limit Order Books in High Frequency Finance." In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. Apr. 2018, pp. 4264–4268.

[Woo] O. J. Woodman. "An introduction to inertial navigation." en. In: (), p. 37.

[WS21] J. Wahlström and I. Skog. "Fifteen Years of Progress at Zero Velocity: A Review." In: *IEEE Sensors Journal* 21.2 (Jan. 2021). Conference Name: IEEE Sensors Journal, pp. 1139–1151.

[Wu12] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman. "Classification Accuracies of Physical Activities Using Smartphone Motion Sensors." en. In: *Journal of Medical Internet Research* 14.5 (Oct. 2012), e130.

[WX14] Z. Wang and X. Xue. "Multi-Class Support Vector Machine." en. In: *Support Vector Machines Applications*. Ed. by Y. Ma and G. Guo. Cham: Springer International Publishing, 2014, pp. 23–48.

[XW19] N. Xia and M. A. Weitnauer. "TDOA-Based Mobile Localization Using Particle Filter With Multiple Motion and Channel Models." In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 21057–21066.

[Yan14] G.-Z. Yang, ed. *Body Sensor Networks*. en. London: Springer, 2014.

[Yan18] Z. Yang, Y. Wang, L. Zhang, and Y. Shen. "Indoor Interference Classification Based on WiFi Channel State Information." en. In: *Security, Privacy, and Anonymity in Computation, Communication, and Storage*. Ed. by G. Wang, J. Chen, and L. T. Yang. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 136–145.

[YC17] Z.-S. Ye and N. Chen. "Closed-Form Estimators for the Gamma Distribution Derived From Likelihood Equations." In: *The American Statistician* 71.2 (Apr. 2017). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00031305.2016.1209129, pp. 177–181.

[Yeh09] L.-W. Yeh, M.-S. Hsu, Y.-F. Lee, and Y.-C. Tseng. "Indoor localization: Automatically constructing today's radio map by iRobot and RFIDs." In: *2009 IEEE SENSORS*. ISSN: 1930-0395. Oct. 2009, pp. 1463–1466.

[Ye16] H. Ye, T. Gu, X. Tao, and J. Lu. "Scalable floor localization using barometer on smartphone: Scalable floor localization using barometer on smartphone." en. In: *Wireless Communications and Mobile Computing* 16.16 (Nov. 2016), pp. 2557–2571.

[YHF19] H. Yan, S. Herath, and Y. Furukawa. *RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods*. arXiv:1905.12853 [cs]. May 2019.

[Yiu17] S. Yiu, M. Dashti, H. Claussen, and F. Perez-Cruz. "Wireless RSSI fingerprinting localization." en. In: *Signal Processing* 131 (Feb. 2017), pp. 235–244.

[YKK13] T. Yang, K. Kaji, and N. Kawaguchi. "Elevator Acceleration Sensing: Design and Estimation Recognition Algorithm Using Crowdsourcing." In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. July 2013, pp. 534–539.

[YLY19] Z. Yuan, W. Li, and S. Yang. "Beacon Node Placement for Minimal Localization Error." In: *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. July 2019, pp. 980–985.

[Yua15] X. Yuan, S. Yu, S. Zhang, G. Wang, and S. Liu. "Quaternion-Based Unscented Kalman Filter for Accurate Indoor Heading Estimation Using Wearable Multi-Sensor System." en. In: *Sensors* 15.5 (May 2015). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, pp. 10872–10890.

[Yur20] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies." In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 58443–58469.

[Yu19] Y. Yu, R. Chen, L. Chen, G. Guo, F. Ye, and Z. Liu. "A Robust Dead Reckoning Algorithm Based on Wi-Fi FTM and Multiple Sensors." en. In: *Remote Sensing* 11.5 (Jan. 2019). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 504.

[YW15] L. Yang and M. Worboys. "Generation of navigation graphs for indoor space." In: *International Journal of Geographical Information Science* 29.10 (Oct. 2015). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13658816.2015.1041141, pp. 1737–1756.

[Zen22] Y. Zeng, J. Liu, J. Xiong, Z. Liu, D. Wu, and D. Zhang. "Exploring Multiple Antennas for Long-range WiFi Sensing." In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.4 (Dec. 2022), 190:1–190:30.

[ZG21] F. Zangenehnejad and Y. Gao. "GNSS smartphones positioning: advances, challenges, opportunities, and future perspectives." en. In: *Satellite Navigation* 2.1 (Nov. 2021), p. 24.

[Zha06] C. Zhang, M. Kuhn, B. Merkl, A. Fathy, and M. Mahfouz. "Accurate UWB indoor localization system utilizing time difference of arrival approach." In: *2006 IEEE Radio and Wireless Symposium*. ISSN: 2164-2974. Oct. 2006, pp. 515–518.

[Zho15] B. Zhou, Q. Li, Q. Mao, W. Tu, and X. Zhang. "Activity Sequence-Based Indoor Pedestrian Localization Using Smartphones." In: *IEEE Transactions on Human-Machine Systems* 45.5 (Oct. 2015). Conference Name: IEEE Transactions on Human-Machine Systems, pp. 562–574.

[ZL08] Y. Zhou and J. Li. "Robust Decentralized Data Fusion Based on Internal Ellipsoid Approximation." en. In: *IFAC Proceedings Volumes*. 17th IFAC World Congress 41.2 (Jan. 2008), pp. 9964–9969.

Appendix A

Graphical Survey of Localization Systems

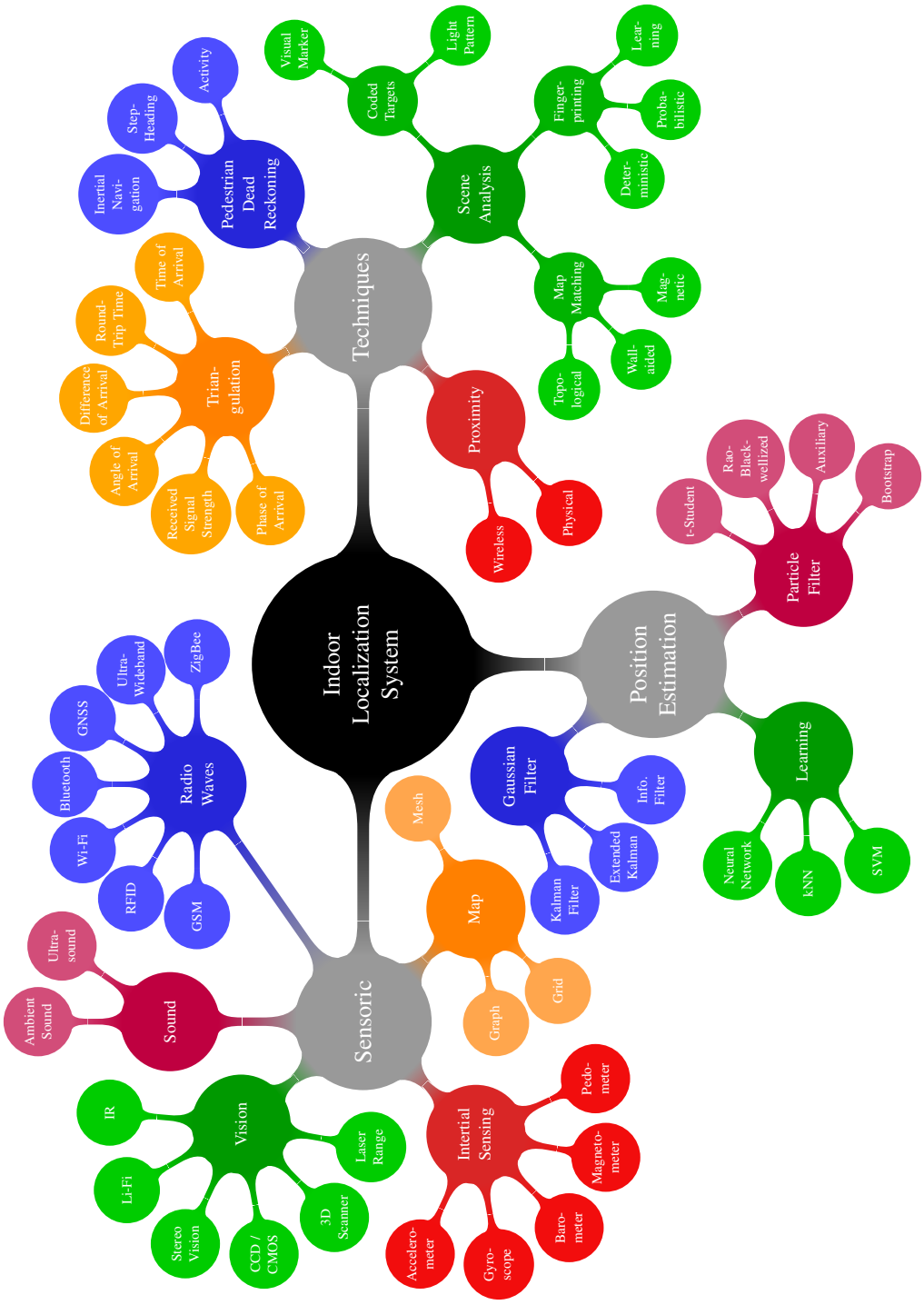
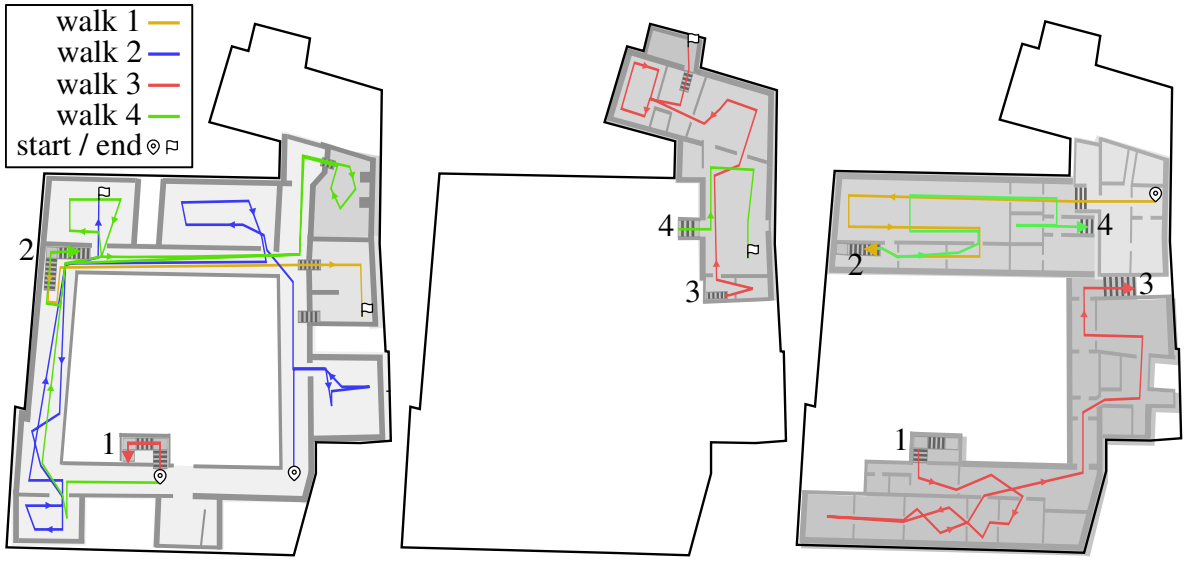


Figure A.1: Comprehensive mindmap of technologies for indoor localization systems. The information has been gathered from different surveys and overviews [Ala16; DP17; Har13; Liu07; Mau12; MPS14; MT11].

Appendix B

Ground Truth of Test Walks

The ground truth trajectories of the experimental setups discussed in this work are shown in the figures below. We call them a *walk* as all test subjects were pedestrians.



(a) Ground floor (b) First floor (c) Second floor

Figure B.1: Ground truth of all four walks conducted within the ROT building ($70\,\mathrm{m} \times 50\,\mathrm{m}$ with $2000\,\mathrm{m}^2$). The arrows indicate the walking direction. The stairs are numbered so that it is easy to see how the floors are connected. Walk 1 is $152\,\mathrm{m}$ long and took about $2\,\mathrm{min}\ 30\,\mathrm{s}$ to walk. Walk 2 has a length of $223\,\mathrm{m}$ and walk 3 a length of $231\,\mathrm{m}$, both required about $6\,\mathrm{min}$ to walk. Finally, walk 4 is $310\,\mathrm{m}$ long and takes $10\,\mathrm{min}$ to walk.

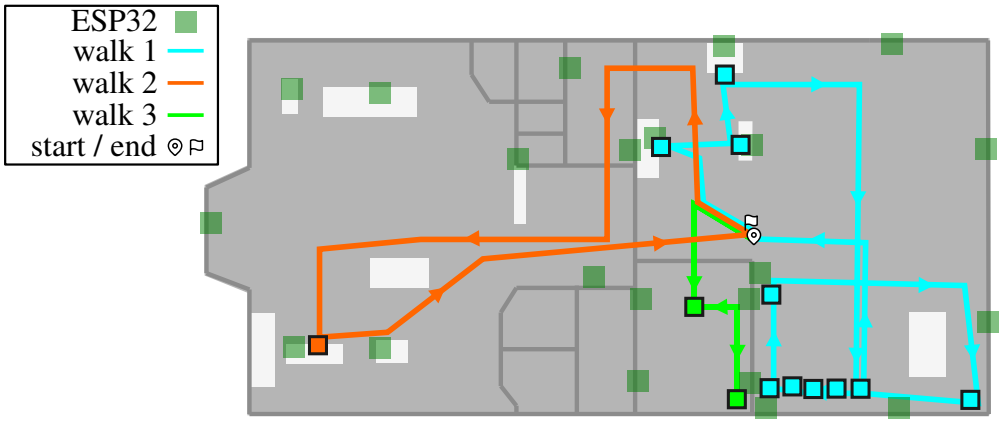


Figure B.2: Ground truth of all four walks conducted within the IPA building (38 m × 15 m). The arrows indicate the walking direction and the black rectangles indicate the checkpoints where the subject had to simulate work at the workstations. Walk 1 is 72 m long and took about 5 min to walk. Walk 2 has a length of 62 m with a duration of about 1 min 30 s. Finally, walk 3 is 27 m and 2 min. All walks have the same start and end. To provide a BLE and Wi-Fi FTM infrastructure, 22 ESP32-S3-DevKitC-1 were installed as beacons.

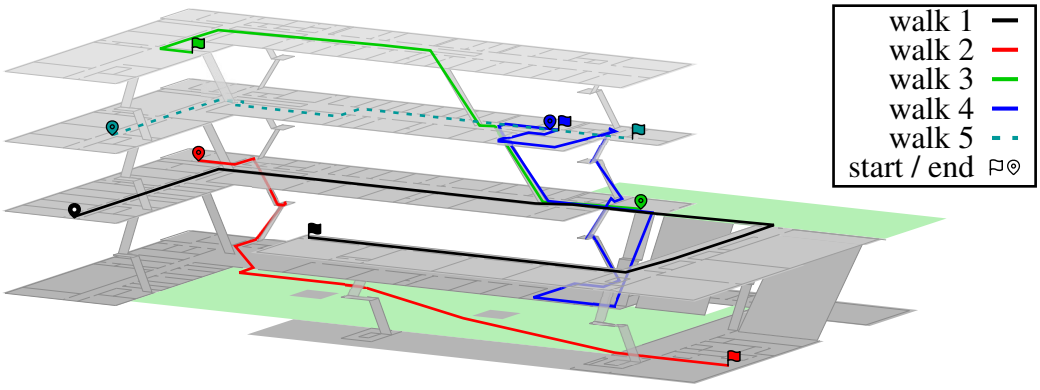


Figure B.3: Ground truth for five walks recorded between 2017 and 2020 on the complete campus of SHL (12 000 m^2). The positions of the Wi-Fi access points is omitted due to legal issues. Walk 1 is 207 m long and took about 3 min to walk. Walk 2 has a length of 138 m with a duration of about 2 min 30 s. Both paths lead across the exterior from the faculty building to the lecture hall building. Walk 3 (86 m, 2 min), walk 4 (140 m, 3 min) and walk 5 (97 m, 2 min) are conducted inside the faculty building only.

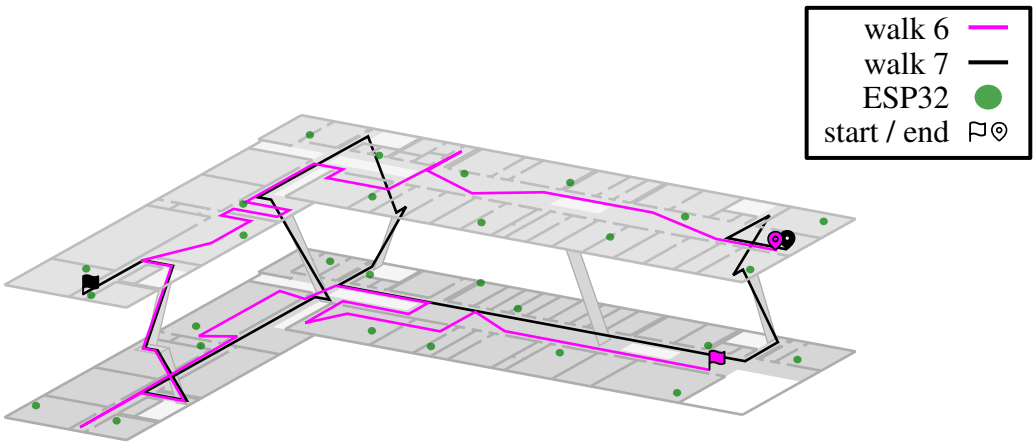


Figure B.4: Ground truth for two walks recording between 2021 and 2023 in floor two and three of the faculty building of SHL ($77\,\mathrm{m} \times 50\,\mathrm{m}$). To provide a BLE and Wi-Fi FTM infrastructure, 28 ESP32-S3-DevKitC-1 were installed as beacons. Walk 6 is $300\,\mathrm{m}$ long and took about $6\,\mathrm{min}$ to walk. Walk 7 has a length of $200\,\mathrm{m}$ with a duration of about $4\,\mathrm{min}\ 20\,\mathrm{s}$.