Human Data Understanding Sensors, Models, Knowledge

Editors: Frank Deinzer & Marcin Grzegorzek

Frank Ebner

Smartphone-Based 3D Indoor Localization and Navigation

λογος

Human Data Understanding
Sensors, Models, Knowledge

Vol. 1

Human Data Understanding
Sensors, Models, Knowledge

Volume 1

Herausgegeben von

Prof. Dr.-Ing. Frank Deinzer
Hochschule für angewandte Wissenschaften
Würzburg-Schweinfurt

Prof. Dr.-Ing. Marcin Grzegorzek
Universität zu Lübeck

Frank Ebner

Smartphone-Based 3D Indoor Localization and Navigation

UNIVERSITÄT ZU LÜBECK

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

"Smartphone-Based 3D Indoor 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

Frank Ebner

from Werneck

Lübeck 2020

First referee: Prof. Dr.-Ing. M. Grzegorzek

Second referee: Prof. Dr.-Ing. A. Schrader

Third referee: Prof. Dr.-Ing. F. Deinzer

Date of oral examination: 21.09.2020

Approved for printing: Lübeck, 30.09.2020

Acknowledgements

To Katharina, who has been a true friend, and will never be forgotten.

Frank Ebner

Abstract

With the steadily increasing need and wish to travel, people often have to reach locations they have never been to before. Modern means of transportation, like cars, ships and planes, thus come equipped with onboard navigation systems, assisting with this task, based on the global positioning system (GPS), or a derivative. However, the navigation task is not solely limited to outdoor environments. Reaching the correct gate within an airport, finding a ward in an unknown hospital, or the auditorium within a new university, represent navigation problems as well. With the GPS requiring a direct line of sight towards the sky, it is unavailable for absolute location estimation indoors. Therefore, the question for suitable indoor navigation techniques arises. Besides localization accuracy, additional factors should be met for such a new system to become a success. It should be easy to set up and maintain, limiting required working hours and costs. Likewise, hardware for the users themselves should be cheap, and readily available. Due to the ubiquity of smartphones, these devices represent a desirable platform for pedestrians, backed by the variety of sensors installed in these devices. Within this work, smartphone-based pedestrian indoor localization and navigation is discussed in detail. This covers examining the suitability of several available sensors: step-detection using readings from the accelerometer, relative turn-detection utilizing the turn rates of the gyroscope, absolute heading estimations based on the magnetometer's indications, and altitude evaluation from the barometer. While all aforementioned sensors do not require any additional infrastructure, thus suitable for all sorts of buildings, they only allow for relative location estimations. Absolute localization can utilize Wi-Fi, as it is supported by all smartphones, and most public buildings already contain the required infrastructure. Due to the behavior of radio signals, the smartphone's current location can be approximated by examining signal strengths of nearby transmitters. This aspect is often utilized by Wi-Fi fingerprinting, which, however, requires a time consuming setup process. Therefore, an alternative is developed that allows for significantly faster setup times. Additionally, the building's 3D floorplan is included, modeling potential pedestrian movements, limiting impossible walks to improve estimation results, and to provide routing towards a desired destination. For this, two spatial floorplan representations are derived and examined. All aforementioned aspects are hereafter combined probabilistically, using recursive density estimation based on the particle filter. This allows for fusioning all sensor observations while respecting their individual uncertainties, and the building's floorplan as additional constraints.

To summarize, the system described within this work covers probabilistic 3D pedestrian indoor localization, using commodity smartphones, contained sensors, a building's existing infrastructure and floorplan, all combined by the particle filter to derive an indoor localization and navigation system that is easy to set up and maintain.

Zusammenfassung

Mit dem stetig zunehmenden Reisewunsch finden sich Menschen immer häufiger vor der Aufgabe, bislang unbekannte Orte zu erreichen. Moderne Transportmittel, wie Autos, Schiffe und Flugzeuge sind deshalb mit GPS-basierten Navigationssystemen ausgestattet, die hierbei unterstützen. Allerdings ist der Navigationsaspekt selten nur auf den Außenbereich beschränkt. Das richtige Gate im Flughafen zu finden, eine Station im Krankenhaus, oder den Hörsaal in der neuen Universität, ist oft ähnlich anspruchsvoll. Da das GPS jedoch eine direkte Sichtverbindung benötigt, steht dieses innerhalb von Gebäuden nicht zur Verfügung. Hier stellt sich deshalb die Frage nach geeigneten Alternativen. Für Neuentwicklungen müssen neben der Positionsgenauigkeit auch andere Aspekte berücksichtigt werden. Das System sollte nicht nur wartbar, sondern auch kostengünstig ausrollbar sein. Auch für die Nutzer sollten die Anschaffungskosten so gering wie möglich ausfallen. Smartphones stellen aufgrund ihrer Allgegenwärtigkeit und Vielzahl von Sensoren deshalb eine ideale Zielplattform dar. In dieser Arbeit werden verfügbare Sensoren auf ihre Tauglichkeit untersucht: Schritterkennung mittels Beschleunigungssensor, Laufrichtungsschätzung via Magnetometer, Laufrichtungsänderungen gemessen durch das Gyroskop, und Höhenbestimmung per Barometer. Diese Sensoren stellen zwar keinerlei Anforderungen an das Zielgebäude, liefern jedoch lediglich relative Informationen bzgl. möglicher Aufenthaltsorte. Eine absolute Positionsbestimmung wird über Wi-Fi ermöglicht, welches von allen Smartphones unterstützt wird und in den meisten öffentlichen Gebäuden verfügbar ist. Basierend auf dem Verhalten von Funksignalen lässt sich der aktuelle Standort des Smartphones aus den Signalstärken der nahegelegenen Access Points ableiten. In der Literatur wird hierfür häufig auf Fingerprinting zurückgegriffen, welches zwar genau, aber aufwending in der Einrichtung ist. Deshalb wird eine Alternative erarbeitet, die die Einrichtungszeit und Kosten stark reduziert. Zusätzlich wird ein 3D Gebäudeplan verwendet, der mögliche und unmögliche Bewegungen von Fußgängern bestimmen, und die kürzeste Route zu einem gewünschten Ziel berechnen kann. Beides dient der Verbesserung der Vorhersagen des Gesamtsystems. Hierfür werden zwei verschiedene Repräsentationen des Gebäudeplans erzeugt und untersucht. Alle vorherigen Komponenten werden schließlich über rekursive Dichte-Schätzung mittels Partikel-Filter zusammengeführt. Mit dieser lassen sich alle Sensor Messungen inklusive ihrer Unsicherheiten kombinieren, und auch der Gebäudeplan kann als zusätzliche Rahmenbedingung integriert werden, um unmögliche Bewegungen auszufiltern.

Zusammenfassend beschreibt diese Arbeit ein auf Wahrscheinlichkeitsrechnung basierendes 3D Lokalisations- und Navigations-System für Fußgänger in Gebäuden, das alle Informationen mittels Partikel Filter kombiniert, einfach einzurichten und zu warten ist. Vorausgesetzt werden lediglich ein Smartphone, eine vorhandene WLAN-Infrastruktur und ein Gebäudeplan.

Contents

Chapter 1

Introduction

Finding a specific place or location one has never been to, or hasn't been to for a long time, is a common task that everybody encounters from time to time. Back in the days, almost everyone had a road map stowed away within the car's glovebox, ready to use, whenever needed. However, without a co-driver, reading the map and providing instructions, this was a quite cumbersome solution for getting from A to B. This situation changed in the year 2000, when the former military-only global positioning system (GPS) became freely available for civilian use. Now it was possible to locate objects anywhere on the earth, with an accuracy down to a few meters, using just a single receiver. Combined with digitized maps, this allowed for both, self-localization and navigation [DH10].

Starting from there, it only took several months for receivers to become both, significantly cheaper and smaller, and companies like *TomTom* or *Garmin* started developing products for motor vehicles, using digital maps from vendors such as *Tele Atlas* or *Navtech*. At first, navigation systems were either installed directly within a vehicle, or required fully featured hardware, like portable computers equipped with an external receiver. Yet, with the advent of *Personal Digital Assistants* (PDAs), containing even smaller GPS receivers and mass storage devices based on flash memory, navigation systems became portable. Today, almost every new smartphone is suitable for GPS-based navigation, using its built-in sensors, as well as a piece of software that includes the necessary maps and navigation algorithms.

Inexpensive receivers for GPS, new similar systems, like GLONASS, and (freely) available maps for almost every place on earth, lead to the ubiquity of navigation systems. Thus, their success was not only based on demand, but also on the availability of relatively affordable components for hardware, software, and low running costs. At least for the *customer*: Consumer hardware can be used for several years, and, depending on the vendor, map updates are either free of charge, part of an annual subscription, or charged per update.

For *providers*, however, the situation is different. Additionally to several billion dollars for the initial development and setup, the infrastructure behind GPS has to be kept up and running, costing additional millions – per day. While licensing fees, e.g. for access to increased accuracy, compensate for some of these costs, the remaining part is paid by the US government. Running costs for the *vendors* of navigation appliances can be expected to be much cheaper. However, due to rapid changes in infrastructure, they have to provide up-to-date mapping data, resulting in many companies charging for updates [Pac+95; DH10]

Due to ever-increasing globalization and transnational business connections, the need and wish to move or travel is constantly increasing. Besides getting to airports, train stations or company compounds, the buildings themselves often represent a navigation problem as well: Finding the correct terminal within an airport, the conference room in a large company, a room within a townhall, or the correct ward within hospitals, isn't always straightforward. With this in mind, localization and navigation *indoors* becomes of increasing importance as well.

However, while working perfectly for most navigation purposes, e.g. for cars, pedestrians and cyclists, currently available systems are unsuited, as both, the sensors and the typical map formats, are intended for *outdoor* use. For good location estimations, GPS relies on a direct line-of-sight between satellites and receiver, and older devices thus had to be installed on top of the car, in order to function properly. Similarly, the format of most digitized maps is focused on outdoor purposes, as the underlying data structures mainly use a two-dimensional representation of roads, lanes, and intersections, unsuited for modeling a building's interior.

Furthermore, when considering indoor environments, completely different use cases, besides typical navigation from A to B, arise as well. Starting from finding a specific product within a large supermarket, to the economy's interest in location-based services, e.g. placing ads for nearby products as well. Also covering cultural aspects, like guided tours through a museum, presenting useful information on exhibits, based on the visitor's current location and viewing direction. Depending on the building and intended use case, requirements can be completely different. This especially concerns the aspect of *localization accuracy*. While a coarse GPS location estimation is sufficient for a car driving along the motorway, it can be too erroneous for a slowly paced pedestrian, walking through an area with many small alleyways. The same holds true for localization indoors, where estimating the current whereabouts on a room-level scale might be sufficient for some intentions, like presenting information on nearby exhibits. For others scenarios, such as navigation, however, estimations should be as accurate as possible, for audible commands and visualizations given to the user, to be helpful instead of misleading.

Therefore, the question arises, how such a multi-purpose indoor localization and navigation system can be developed, and what criteria should be met for it to be valuable.

1.1 Navigation within Buildings

Based on the previous aspects, it becomes clear that the topic of localization indoors is not solely related to sensors and achievable accuracy, but also to costs, for initial setup, maintenance over time, software and hardware required by the consumer, and by the system's operator. In case of localization indoors, the latter is unlikely to be a government, like it is for the GPS, GLONASS or Galileo, but more likely the owner of the building to deploy the system to, like an airport, hospital, supermarket or museum. This gives even more importance to the aspect of costs, as many public buildings that benefit from indoor localization, like townhalls or museums, typically are on a tight budget. Closely coupled with costs is the time required for setup and servicing, as they also arise per building, additionally dependent on its size. As known from other projects, the solution is a tradeoff between quality (accuracy), time and costs.

Similar aspects apply to the required building maps. As it is unlikely for a global company to create maps for every single building, where indoor localization could possibly be used, this data has to be supplied by the operator or a public community, dedicated to this task [Ope]. Furthermore, in contrast to maps for navigation outdoors, indoor maps can be rather eclectic, as they have to support buildings with multiple floors, elevators, escalators, and different types of stairs [EBS16; Elh+14]. Depending on the intended use case, they should also support adding semantic information, like room numbers, points of interest, and access restrictions or limitations. The latter is especially relevant to the disabled, who are unable to take stairs, or require additional audible information when visually impaired. These aspects can also affect the topic of navigation, as the shortest path towards the destination might not be the best solution for all pedestrians, especially not for those being handicapped or injured.

Based on the previously mentioned thoughts, a non-exhaustive list of requirements for indoor localization and navigation thus contains the following aspects:

- *Software and Hardware required by the consumer* should be as cheap as possible, with required components being small and always at hand, if possible.

- The system's *accuracy* must be sufficient for a pedestrian to be localized within the building, and to provide navigation guidance. Hereby, *sufficient* is not quantifiable, strongly depends on the intended use case, and the building's architecture, as narrow corridors with many adjacent rooms require a higher accuracy than e.g. large, open shopping malls.

- Time and costs for the *initial system setup* should be as low as possible. This includes costs for all necessary hardware components, time for their setup, and effort needed to provide a digital map of the building's floorplan.

- Time and costs for *maintenance* after the initial setup should be as low as possible. Ideally, the system is easily adaptable to architectural changes, like new/removed drywalls.

- *Partial failures of the infrastructure* should not completely disable the whole system, only may affect the provided accuracy.

Besides use case-dependent details, the question of suitable hardware components is the most critical. As existing positioning methods like GPS and GLONASS do rarely work indoors, other sensors are required to infer an absolute location. As of today, there is no established solution, and this matter is still open for new suggestions. However, to conform with previous discussions, it should not only be accurate, but also cheap, and easily available. Therefore, most ongoing research is targeted at *smartphones*, as they are ubiquitous, almost always at hand, and contain an increasing number of sensors [Tia+15; Gui+16; Ndz+17; Ye+14; Mou+15; Kir+18].

That is in contrast to outdoor navigation, where new platforms started to develop around the existence of a single sensor. For indoor localization and navigation, a desirable target platform is already available, and the question arises, whether it is suitable for the intended task. This lead to numerous new research topics, analyzing the suitability of certain sensors, that are installed within commodity smartphones. Most of them are adapted from previous research in different fields, where some sensor or component has already been proven helpful.

This e.g. covers velocity and heading, estimated from an accelerometer and a gyroscope, together providing the base for *dead reckoning* [ND97], which allows relative (incremental) location estimations, if initial whereabouts are known. This technique already underwent extensive research to adapt it from vehicles to pedestrians. Yet, the focus was mainly on *multiple* sensors, attached to different parts of the body, picking up leg movement and turning behavior of a pedestrian, well-suited for motion estimations [SD16; TS12; Goy+11]. With the rising interest in indoor localization, it began to be adapted to smartphone-only setups, where the orientation of the device has to be considered, when the pedestrian e.g. holds the smartphone upfront, looking at its screen while navigating through a building [PHP17; Yu+19; Kus+15].

Yet, with dead reckoning providing information on *relative movements*, it is only suitable when initial whereabouts are known, and it is likely to fail over time, due to increasing errors. For actual indoor localization, hints on absolute whereabouts are mandatory. For this, former research on Wi-Fi-based location estimation [BP00] became of interest again. By using signal strength observations from nearby access points, it is possible to roughly estimate the distance towards them, and thus a coarse, absolute location information. This strategy also conforms with most aforementioned requirements: As of today, most public buildings are equipped with Wi-Fi, already containing the required infrastructure, and Wi-Fi is supported by all modern smartphones. However, besides these positive aspects, achievable accuracy is either too coarse,
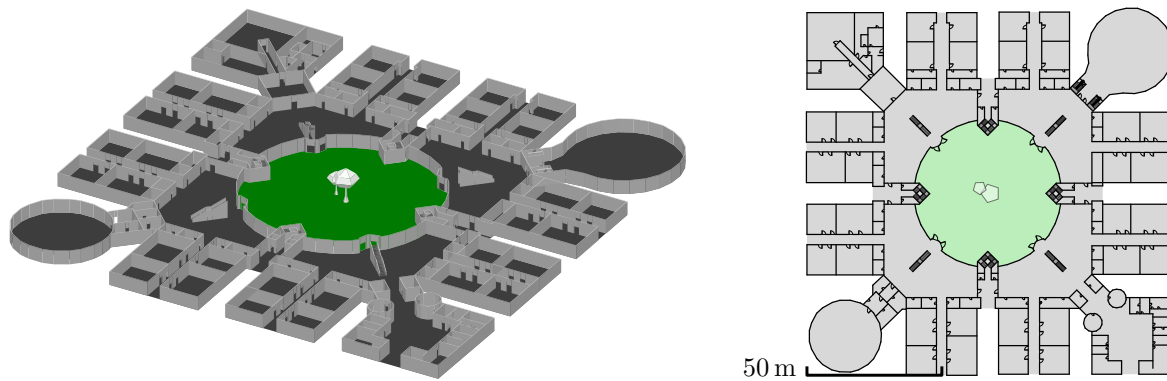
Figure 1.1: Example of a complex single-floor, with large open spaces and small adjacent rooms. First floor of the UAH building of the *University of Alcalá de Henares*, Spain.

or a manual and time-consuming setup is required beforehand. During the latter, accuracy is increased by actually measuring the behavior of the installed infrastructure's radio signals, throughout the whole architecture. Thus, this area is still undergoing extensive research.

In contrast to navigation outdoors, there is not yet a single sensor that solves the problem formulation with sufficient accuracy. Instead, research tends towards employing combination of multiple components, each of which providing a contribution to the overall result. Besides the two mentioned examples for relative and absolute estimations, various other sensors, such as the camera, magnetometer or barometer, which are also found within smartphones, can thus be of interest as well [HB08; Shu+15; Mur+14].

Alongside sensors, where some components already seem established, mapping still requires extensive research. In outdoor navigation, a graph data structure is ideal to model rivers, roads and interconnections, for both, displaying and routing. Considering indoor use cases, however, there is not yet a clear best-candidate among potential data structures [ARC12]. Indoor environments are less restrictive and often inhomogeneous, ranging from narrow hallways with multiple adjacent rooms, to large open spaces, as can be seen in figure 1.1 and 1.2. This scalability must be supported by the chosen model, including minor details where needed, yet without requiring too much memory. Furthermore, the map has to provide all the semantic information that might be required for some sort of sensor component. Additionally, multiple floors and their interconnections, like stairs, escalators or elevators, are also a strong requirement. Not to mention *editability*, as the map has to be generated for each and every building, with support for including future architectural changes. The problem of creating a 3D representation of such a multistory building has already been solved by computer graphics [KSS17]. Yet, determining whether a particular movement is possible, calculating the shortest path towards a room or point of interest, correctly including stairs and elevators, all while being computationally efficient, still is a topic of active research.

Deutsches Hutmuseum Lindenberg, Germany   *RothenburgMuseum*, Germany
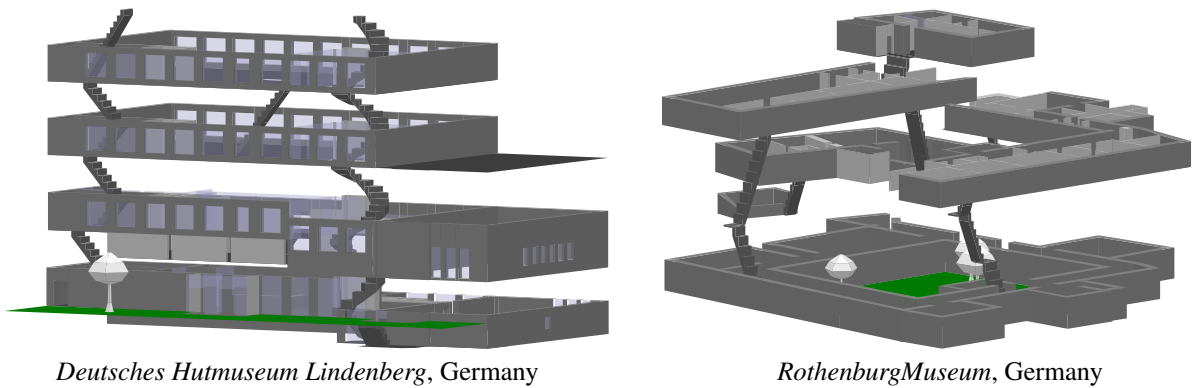
Figure 1.2: Two complex multi-floor buildings. While the left one is stacked almost evenly, the right one is irregular in size, shape, and floor-level. The distance between floors was increased for visualization.

Mentioned earlier, the floorplan not only serves as a visualization to the user, it contributes valuable information as well. The map within car navigation systems is also used to compensate uncertainties of the GPS, e.g. by placing the virtual car onto the nearest road. Additionally, when the car drives through a tunnel, and the GPS signal is lost, the last known velocity and heading can be used to continue predicting the car's whereabouts, based on the underlying mapping information. Similar aspects apply to localization and navigation indoors, where the map is used to denote possible movements, limit impossible movements, and to prevent the impact of sensor uncertainties and errors. For example, assuming two subsequent absolute location observations to be ten meter apart from each other. Such a change in location is likely, when both locations refer to the same floor, and several seconds have passed between the two sensor observations. Similarly, such a change is unlikely, when e.g. only one second has passed, or both locations belong to two different floors, and neither stairs nor elevators nor escalators are nearby. By combining assumptions on *pedestrian walking behavior* and information provided by the floorplan, probabilities for potential location changes can be inferred.

Aforementioned aspects lead to the requirement for a technique, which fuses all available information, to derive the overall result. As every sensor provides its own point of view, there is no straight-forward solution of combining all observations. Especially in case of sensors indicating relative location changes, restrictions of the floorplan should be included to rule out physically impossible movements. Furthermore, every single component is subject to different types of errors that must be considered as well. The overall task thus is to determine the *most likely* whereabouts, based on all sensor observations, assumptions, and the building's floorplan. Depending on the complexity of the latter, and the number of sensors, this task can exceed the capabilities of embedded devices, and represents an extensive research topic on its own [Gus10].

Based on all presented thoughts and requirements, the research objective of this work is formulated within the following.

1.2 Research Objective

In contrast to outdoor navigation, where most devices were developed around a single sensor, with its accuracy sufficient for most use cases, as of today, pedestrian indoor localization relies on multiple sensors, with the smartphone representing a desirable target platform. The goal of this work is to derive a scalable system for pedestrian indoor localization and navigation, targeting this platform. Thus, the focus is solely on smartphones, the sensors available within, and to build a system that is suitable for most use cases, easy to set up and maintain. Neither requiring large amounts of time, nor cost for setup and infrastructure. While considering solely sensors and infrastructure available as of today, the discussed system is intended to be scalable, allowing for easily including new sensors in the future. For the use case of localization and navigation, the smartphone is expected to be held upfront by the pedestrian, e.g. looking at navigational advice, presented on the device's screen. This aspect is relevant to certain sensors and corresponding coordinate systems, discussed throughout the course of this work.

With GPS being unavailable indoors, Wi-Fi is considered the main component for absolute location information, as required infrastructure is available within most buildings where localization or navigation are a benefit, and it is supported by most of today's smartphones [BP00; YA05; Roo+02; Liu+12]. Yet, with the expected accuracy being insufficient for navigation, additional sensors are required. Here, the focus is on well-known dead reckoning techniques that are adapted for use on smartphones. This e.g. covers the smartphone being held upfront by the pedestrian, therefore applying required compensation techniques. Besides, additional sensors, such as the barometer and magnetometer, will also be considered, providing further information to increase the overall accuracy, without affecting setup, costs or maintenance. As discussed, every sensor component is subject to different types of errors that have to be handled accordingly. Therefore, the focus is on *probabilistic* approaches, including all sensor observations based on their likelihood. That is, for every individual component, a probabilistic model will be derived, describing the likelihood of some whereabouts or movements, from every sensor's point of view.

Not only relevant for visualization purposes, but also for limiting impossible movements or for providing routing information to derive the best path towards some destination, the building's floorplan represents the second major research objective. Conforming with sensors and aforementioned aspects, probabilistic movement models will be derived, where the floorplan is used to describe potential and unlikely pedestrian movements.

The information from individual smartphone sensors is combined by *sensor fusion*, based on *recursive density estimation* [MU49; Mar51; Sär13]. This is used to determine the globally most likely whereabouts, based on *all* sensors observations since starting the estimation process.
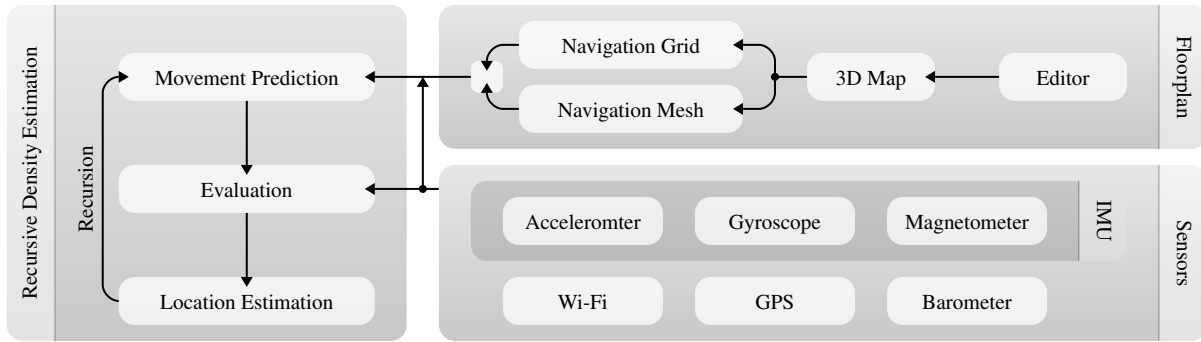
Figure 1.3: Brief overview of the overall system. Floorplan and Sensors represent the main source of information, combined via recursive density estimation, determining the most likely whereabouts.

By considering the *history* of all sensor observations, *relative* location information, like aforementioned dead reckoning, are supported as well, and results are refined over time. Throughout this process, the floorplan will be included, used to e.g. filter impossible movements that would cross a wall or other obstacles. To include individual errors, chances and similar, all required calculations are given on a probabilistic basis.

Figure 1.3 provides an overview of the overall system, its individual components, and the way they interact with each other. This figure is intended to provide a brief impression on the global research objective, without going into details of each and every component. As can be seen, the sensors and the building's floorplan represent the two main sources of information, combined via recursive density estimation. Both, sensors and floorplan, are intended to be interchangeable, with the ability to include new sensors and spatial models, scaling with new future components. To get an impression on the impact of choosing some specific data structure, two different spatial floorplan models, as well as their advantages and disadvantages, will be discussed. This also addresses the topic of how to include *semantic* information, e.g. to label a room, or to include additional information, useful for routing or people with special needs.

To summarize, the focus of this research is on deriving a smartphone-based pedestrian indoor localization and navigation system, enabling to localize *oneself* within a building, e.g. for navigating to a desired destination. This is achieved by adapting existing techniques to this use case, combining the information form several smartphone sensors with movement prediction based on the building's floorplan, by using probabilistic sensor fusion. Other use cases, such as localizing all pedestrians currently residing within a building [Xu+13], are not covered by this work. Also excluded are topics that are related to indoor localization, but not to pedestrians, like determining the current location of some equipment within a large industrial compound [Nuc+04; Kar+17]. Furthermore, the focus is solely on ubiquitous components. Special hardware for accurate localization indoors, such as ultra-wideband [FG02], is thus not considered.

1.3   State of the Art

This section provides a brief overview on the current *state of the art*, concerning the main topics identified during previous remarks and the research objective. More detailed overviews, and *related work* from other researchers, are given within each of the chapters, and individually for every topic.

While indoor localization and navigation became of increasing interest to researchers during the last decade, there is no standardized solution yet. Even when referring solely to smartphone-based systems, the sensors used, the way they are integrated and combined, the required infrastructure, and the underlying spatial models for the floorplan, if used, are completely varying. Most systems refer to some sort of probabilistic setup, combining individual components, based on likelihoods. However, the scale of integration, that is, the number of sensors that are combined, and the degree of additional information added, like the floorplan, is significantly varying. Often, limited fusion techniques are applied, being computationally efficient, but unable to fully include all available information, such as obstacles, or the pedestrian's desired destination [Tia+15; Hel+13; Ndz+17; NRP16; EBS16; Zha+18b].

Probabilistic Sensor Models   As mentioned, core components of the system are sensors, providing information on whereabouts or movements. While the latter can be performed using solely dead reckoning, that is, starting from a known location with incremental updates based on detected movements, this also leads to incremental errors [Ser28]. These errors eventually were considered, estimating the likelihood for certain whereabouts, and their changes over time [Goy+11; Li+12]. Yet, the degree of considered information varies significantly. While some works consider only two sensors and their respective uncertainties, others include additional observations from other components, and further assumptions, affecting the way the probabilistic models are defined and handled [Hel+13; KGD14; Tia+15]. As shown by others, and discussed in a later chapter, probabilistic sensor models that consider prior information, such as the floorplan, can mitigate growing uncertainties, and increase the quality [NRP16; Kna17].

Probabilistic Wi-Fi Localization   With Wi-Fi representing an infrastructure already available within most public buildings, it is also part of many indoor localization and navigation systems. Yet, implementations often rely on a complex and time-consuming setup procedure, conducting fine-grained measurements throughout the whole building, to estimate the behavior of radio signal propagation, required for inferring potential whereabouts [Men+11; YWL12; Zha+18b]. These initial measurements can later be compared against readings from the pedestrian's smartphones, to determine the best matching one, representing the current whereabouts. This variant of localization is rather discrete, and based on the density of these initial measurements. While

interpolation techniques exist, they suffer from various drawbacks, and come with a computational overhead, often exceeding the capabilities of embedded devices [Par62]. Furthermore, resulting accuracy comes at the cost of setup and maintenance times, whenever the architecture or Wi-Fi infrastructure is modified. When on a tight budget, different approaches are required.

These are e.g. given by describing radio signal behavior, using some sort of model [SR92; PC94; JLH11]. Similarly to the initial measurements approach described above, the model's predictions can then be compared against current readings from the smartphone. However, as the model is typically able to perform this prediction for any location within the building, it is continuous, and does not require for additional interpolation. Yet, for every prediction model several parameters are required to describe the behavior of radio signals. The prediction quality thus not only depends on the accuracy of the model itself, but also on the chosen parameters [Sey05; Hee+11]. For use cases where a reduced accuracy is sufficient, empiric values can be chosen, allowing for a fast deployment and adaption to infrastructural changes.

However, for most setups, a compromise between both techniques represents a viable trade-off, with sufficient accuracy and fast setup times, thus being the focus within this work.

Building Floorplans and Probabilistic Movement Prediction   With the floorplan representing an important component of every localization and navigation system, not only for visualization but also for limiting impossible movements and routing, it is part of many state of the art systems. Yet, as there is no standardized format for indoor floorplans, and many spatial representations are suitable [Led06; Yan06; Wu10; ARC12], different approaches have established over time, most of which limited to a specific use case.

Simple 2D setups e.g. describe each floor with lines that can be used for intersection tests, to determine impossible walks [EBS16]. This, however, is not suitable for most buildings, as they consist of multiple stories. Therefore, 2.5D setups were derived, created by stacking multiple 2D floors, with a discrete connection in between [GF06]. Yet, these setups suffer from various drawbacks. On the one hand, intersection tests are costly, thus requiring some sort of pre-calculated approximation for use on embedded devices [Köp+12; NRP16]. On the other hand, due to the *discrete* interconnection, changing floors requires some sort of heuristic or additional sensor information. Besides, this also yields a reduced user experience in visualization.

For both, visualization and prediction, actual 3D representations thus are preferred. To be suited for use on smartphones, the spatial model should be conservative in use of memory. Viable is e.g. a polygonal representation of the walkable surface [WH08], or some other type of primitive [BJK05]. Referring to the aforementioned problem of costly intersection tests, the 3D spatial model should also be able to quickly determine whether two whereabouts are connected or separated by an obstacle, and, if navigation is desired, the shortest path in between.

Independent of the chosen inclusion and spatial representation, the floorplan must be defined in some way or another. Besides manual creation, crowd-based approaches can be suitable, e.g. determining the walkable area by recordings from hundreds of pedestrians, refined over time [AY12]. Yet, this only allows for a coarse representation, not ideal for visualization purposes.

Alternatives are e.g. given by robots equipped with a laser-scanner, recording the building's interior to derive a 3D representation [SCI13; Hes+16], using several panoramic images to estimate depth [CF14], or scanning the blueprint and using algorithms to derive walls, doors, stairs and similar [Liu+17]. However, dependent on the chosen strategy, expensive hardware might be required, stairs are not supported, or semantic information, like room numbers, still has to be added manually.

The quality of the resulting floorplan strongly depends on the chosen technique and the building's architecture. The same holds true for the time needed to acquire all required information. A manual setup, using some sort of editor, thus also is a viable choice.

Sensor and Information Fusion As identified earlier, individual sensors and information should be fused together, including the history of all observations, to derive the globally best solution, based on all previous inputs. Ideally, individual uncertainties are included as well, to decide how trustworthy each information is. The domain of sensor/information fusion, also referred to as *recursive density estimation*, is well-researched, both, analytically and experimentally. Initial analytical approaches were limited to linear and Gaussian problems only [Kal60]. While this is sufficient for some setups, such as basic inertial predictions [Meh70], or general tracking approaches [CHP79], for more complex problems, such as indoor localization and navigation, including the building's floorplan, it is not.

When relaxing some requirements, and slightly modifying the analytical process, nonlinear problems are supported as well [SSM62]. Concerning indoor localization, these changes add support for basic parts of the overall system, like step-detection and tracking [Goy+11; Jim+12; Gar+16]. Yet, more complex information, such as a building's floorplan, can still not be included, as it is impossible to describe the impact of walls, stairs, and similar, on a purely analytical basis.

For this, non-analytical variants were developed, *approximating* the recursive density estimation problem via *simulations* [Del96; LC98; Del98; IB98]. In doing so, they also support discrete and discontinuous problems, like a wall abruptly blocking all movements. However, they either come at the cost of reduced accuracy, or require significantly more computations, as the approximation's quality depends on the number of simulations [CGM07]. Nevertheless, with the steady increase in computational power, they became viable even for use on embedded devices, such as smartphones.

1.4 Scientific Contribution

Throughout the course of this work, a smartphone-based indoor localization and navigation system is derived. While many of the required topics, like pedestrian dead reckoning and probabilistic sensor fusion, are already well researched, some transfer is required to make them suitable for smartphone use, not requiring any additional sensors attached to the pedestrian's body. Similarly, the building's floorplan is to be considered as well, not only for visual representations, but also for determining valid movements, and for navigation indoors. Besides discussing all required theoretical mathematical backgrounds to determine implications and potential limitations of each individual component, the following scientific contributions will be provided throughout the course of this work:

Probabilistic Sensor Models   While dead reckoning [Ser28; ND97], pedestrian dead reckoning [Li+12; Cas+14], step-detection [Goy+11; TS12; SD16; PHP17; Kir+18], and activity-detection [Elh+14; Zho+15; Zha+18a] all are well-established fields of research, only few works focus on predictions that rely solely on a smartphone. Holding the device upfront, e.g. required for navigating while looking at the device's screen, represents a special case, as information on leg movement or similar is unavailable, and the pedestrian's step size can hardly be determined. Furthermore, when using probabilistic relative movements, the building's floorplan imposes constraints that are to be considered.

Therefore, besides discussing required theory, all sensors installed within commodity smartphones are examined concerning their contribution towards smartphone-based indoor localization and navigation, with *holding the device upfront* in mind. For each of the sensors, a probabilistic model is derived, denoting the likelihood of potential pedestrian movements, with respect to recently received sensor readings, and the building's floorplan.

Probabilistic Wi-Fi Localization   While some works focused on a probabilistic point of view, they often imply either a tremendous setup time for conducting measurements throughout the building, and/or are based on very simple signal strength prediction models, coarsely approximating real-world behavior. For most setups, a tradeoff between both is required, delivering an accuracy sufficient for the intended use case, while minimizing setup and maintenance times.

Therefore, a fast setup strategy is presented, using a few reference measurements and numerical optimization to train advanced signal strength prediction models. Each of which is examined, regarding quality and suitability for probabilistic evaluations. Additionally, strategies for enhancing the quality of predictions, suitable for most public buildings, are introduced. Finally, probabilistic evaluations are presented, denoting the likelihood of certain whereabouts, based on some arbitrary signal strength prediction model.

Building Floorplans and Probabilistic Movement Prediction   For every localization and navigation system, a corresponding map is required to perform calculations and for a visual representation to the user. Recently, other systems started to integrate the building's floorplan, not only for visualization, but also for limiting impossible walks, similar to car navigation [WH08; AY12; ARC12; Hil+14; NRP16]. However, this is often limited to 2D or 2.5D representations, using discretely connected floors and relying on intersection tests to determine the validity of some potential movement.

Therefore, two novel strategies are introduced, using a spatial representation of the building, to predict potential pedestrian movements. The introduced approaches are computationally efficient, well suited for smartphone use, and allow for true 3D estimations. In contrast to other research, these models are combined with sensor observations and additional knowledge, to estimate pedestrian movement predictions indoors. This also covers the use case of navigation, deriving realistic routes for the pedestrian to reach a desired destination.

Sensor and Information Fusion   All aforementioned aspects are combined using established sensor fusion algorithms [GSS93; IB98]. After discussing the required theoretical background to determine potential limitations, suitable approaches for fusing all components are presented. These will be mainly based on aforementioned simulations, which can be briefly thought of: Instead of describing the result analytically, try several potential movements, that conform with recent sensor observations and uncertainties, removing physically impossible ones by considering the floorplan, with the remaining denoting potential new whereabouts.

As computational power on smartphones is limited, and calculations affect battery life, strategies for an efficient fusion of sensor observations, floorplan-based movement prediction, and additional knowledge are introduced. This also covers the topic of simulations, and how to reduce their number required for a stable and computationally efficient approximation.

1.5   Structure

The structure of this work is divided into four main categories: First, the three initial chapters provide a theoretical overview on *smartphone sensors* suitable for indoor localization, *pedestrian movement prediction* and the *fusion* of both. Second, the overall system is derived, and several real-world aspects are discussed, concerning required indoor *floorplans* and considerations for being used on smartphones. Third, all aforementioned aspects are examined experimentally, followed by a summary and outlook.

Chapter 2 provides an overview on sensors installed within commodity smartphones, and their contribution towards indoor localization and navigation. They can be divided into two

major groups: sensors providing absolute information, that is, hints on potential whereabouts, and sensors providing relative information, or hints on potential movements. To also consider sensor noise and errors, the focus is on a probabilistic analysis, determining the likelihood of current sensor observations matching with certain pedestrian movements or whereabouts.

Chapter 3 introduces the topic of pedestrian movement prediction, and resulting differences when additionally including the building's floorplan. Again, the focus is on a probabilistic interpretation, determining the likelihood for certain movements, restricted by the floorplan, and, if available, additional knowledge. This e.g. includes the aspect of *navigation*, how to determine realistic walking paths within buildings, and how to include them when predicting potential pedestrian movements. Intentions are comparable to a navigation system for cars, where potential movements are limited by the car's velocity, roads, and the requested destination.

Having introduced two viewpoints of indoor localization and navigation, the perspective of sensors observations, and restrictions imposed by pedestrian walking behavior and a floorplan, chapter 4 discusses the theoretical background required for fusing all available information probabilistically. This topic is examined from both, an analytical viewpoint, limited to certain types of problems, and a simulation-based implementation, required for the overall system.

Hereafter, several peculiarities are discussed briefly in chapter 5. This covers implications for the intended use on smartphones with limited memory and computational power, optimizations for real-world scenarios, as well as generating required building floorplans.

All aforementioned aspects are examined experimentally in chapter 6. To ensure the general suitability of each sensor and component, several synthetic tests are performed beforehand. Hereafter, actual pedestrian walks, conducted within several buildings, are used to examine the contribution of the individual components, presented in chapter 2 and chapter 3. The experiments conclude with localization results, determined from the combined, final system.

Finally, all discussed aspects are summarized in chapter 7, accompanied by an outlook on topics to address and improve in the future, given in chapter 8.

Chapter 2

Probabilistic Sensor Models

As shown in figure 1.3, core component of localization and navigation systems are sensors, returning a plethora of data, used to infer the current location and guide the navigation process. Depending on the sensor's type, the source for its data, the way provided readings are handled, and the contribution to the overall system, can be completely different. Concerning the localization problem, two major groups can be identified. Sensors providing *absolute* information on the location or orientation of the pedestrian, and *relative* ones, describing location or orientation changes. Within car navigation systems, the GPS returns approximate whereabouts of the car, and thus an absolute information. The speedometer allows for inferring the distance taken within some time period, that is, details on relative changes. At a first glance, relative sensors might appear unnecessary, and absolute sensor components seem able to solve the problem of localization on their own. However, they are similarly valuable to the overall system. On the one hand, to compensate for sensor faults, e.g. when a car drives through a tunnel and the GPS is lost, on the other hand, to stabilize the overall system performance. While the speedometer does not provide any absolute location, the returned data is more stable, compared to the GPS. Yet, neither of both sensors return exact readings, and every indication contains some degree of uncertainty. That must be known and addressed, when working with the provided data. The same facts hold true for smartphone-based indoor localization and navigation, relying on a variety of different sensors, installed in today's smartphones. Within this section, available sensors, their potential contribution towards indoor localization and navigation, as well as how to handle their readings on a probabilistic basis, including expected errors, will be examined. While most sensor outputs belong to exactly one of the two groups, absolute or relative, some can be applied to both, resulting in different advantages and disadvantages. The following discussions will later be revisited and combined with other prior information to derive the overall system.
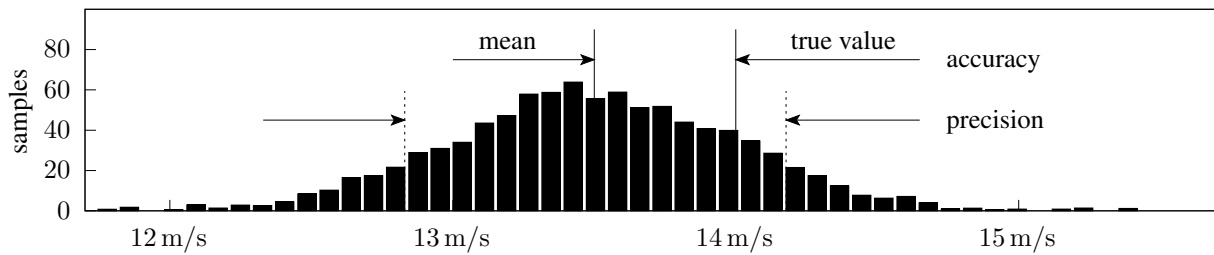
Figure 2.1: Several synthetic readings from a velocity sensor, observing a constant velocity of $14\,\mathrm{m/s}$. The histogram of all observed samples indicates that the sensor's *accuracy* is off by $0.5\,\mathrm{m/s}$. The width of the histogram, that is, the amount of deviation around the mean value, denotes the sensor's *precision*, where smaller is better. Adapted from [Smi99, p. 33].

2.1   Sensor Errors

If every sensor always provided exact readings, one absolute sensor on its own would suffice to solve the problem of (indoor) localization, and many other problem formulations as well. For real-world conditions, every sensor faces some sort of error present within its readings. Depending on the requirements, this error is either acceptable or needs to be addressed in some way. Early car navigation systems used the GPS as single data source, and even though provided readings were off by several meters, this was sufficient for large scale outdoor navigation. Major drawbacks only occurred when the car slowed down and had to take an intersection in locations with several possible options. For addressing such situations, the error of the sensor must be known. Yet, the term *error* is ambiguous, as there are two main types, each sensor is influenced by, and thus must be distinguished. Both are shown in figure 2.1.

On the one hand, the sensor might not provide the *true value*. In such cases, there is an *offset* between every indicated value and the corresponding truth. This is referred to as the *accuracy* of the sensor's measurements. The second type of error addresses the sensor's noise. When measuring the same constant measurand several times, provided readings will not be constant but varying. The amount of variation denotes the sensor's *precision*.

From a statistic point of view, the difference between a constant measurand and the *mean* of several measurements, represents the accuracy, and the variance among all measurements equals the sensor's precision. Ideally, the sensor provides both, a high accuracy and a high precision. Either, or both, requirements will often not hold true for real-world scenarios. While accuracy issues can be addressed via calibration, precision is a given factor that can not be altered directly. At least, it can not when referring to only a single measurement [Smi99].

To improve a sensor's accuracy by calibration, its offset from the true value must be determined. Depicted in figure 2.2, several offset types must be distinguished. Additive offsets can be addressed by subtracting a calibrated constant. The same holds true for multiplicative offsets,
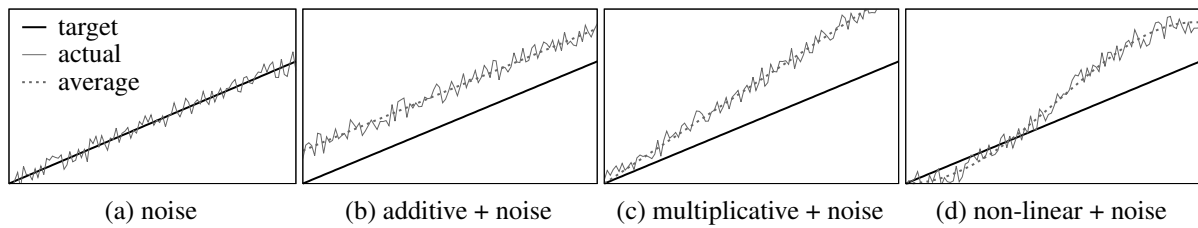
Figure 2.2: Types of errors observable between a target value (x) and its measurement (y). The most desirable is just noise around the target value (a), and should be provided by calibrated sensors. Without calibration, the sensor might provide readings that are: shifted by a constant offset (b), scaled by a constant offset (c), a combination of both, or, at worst, a non-linear modification of the target value (d).

using a division by a calibrated factor. Non-linear modifications of the underlying target-value, however, require a linearization of the actual readings. When used within digital components, this can e.g. be achieved by using a calibrated lookup table (LUT), containing several pairs of actual sensor readings and corresponding true values, provided by a calibrator. Entries must be provided for the whole measuring range, and spaced as closely as possible. To reduce the number of samples needed, readings are often assumed to behave linearly between adjacent entries, which allows for linear interpolation [LC13]. The three error types, additive, multiplicative and non-linear are also referred to as *offset*, *gain* and *linearization* errors [PPG05].

What kind of calibration procedure is the best, depends not only on the use-case and the type of error, but also on the kind of data provided by the sensor. For single valued sensors, like speed or temperature, a simple n-point calibration is often sufficient. Here, n pairs of sensor reading and corresponding true value are used to estimate the sensor's behavior between two adjacent pairs, similar to the aforementioned LUT approach. While a one-point calibration can only mitigate offset errors, a two-point calibration is able to address both, offset and gain. For non-linear sensors, like many temperature sensors that are based on electric resistance (NTC thermistor), more than two reference measurements are required [JP04; SB13; LC13]. In case of multi-valued sensors, the correct calibration strategy depends on whether the individual values are independent or connected in some way. For a 3-axis accelerometer, it is apparent, that all three axes are dependent on each other, and should be calibrated together, e.g. by rotating it around all three axes, hereafter ensuring that the observed measurements denote a sphere [Ols+16]. However, if the three axes are misaligned, no sphere can be constructed and additional compensations are required. That is, several levels of calibration complexity can be identified.

Besides this obvious case of dependency, others are less apparent, and dependencies can also exist between physically unconnected sensors. Temperature and Hall effect sensors (magnetometer) seem independent at a first glance, but the latter is dependent on the ambient temperature, and its readings will vary with changing ambient conditions [Cho+12]. To receive

correct readings for various working conditions, independent of the current temperature, the magnetometer must be calibrated for several temperature ranges beforehand.

Due to the behavior of many electronic components, most sensors can only be calibrated for specific environmental conditions, and calibration procedures are required to follow strict ambient conditions. The German Accreditation Office (DAkkS), for example, requires official calibration documents to contain the values of all influencing ambient conditions, prevailing during the calibration [Deu10]. For electronic devices, the *ambient temperature* is the most critical value. In rare cases, *relative humidity* and *atmospheric pressure* are also required [VDIb]. Detailed calibration requirements are mentioned within the series [VDIa], and depend on the type of the unit under test. Many vendors thus explicitly specify allowed conditions for using their equipment. A temperature around $(23 \pm 5)\,°\text{C}$ and relative humidity $< 90\,\%$ are common requirements for using calibrated electronic devices [Flu99].

Besides aforementioned simple n-point calibrations there are many other variants, differing in required calibration time, computational complexity, necessary amounts of memory during runtime, resulting accuracy, and whether they need to be supervised. Bouhedda [Bou13] suggests using neural networks for calibrating non-linear sensors, and compares a network-based calibration of a temperature sensor against using a LUT and a known polynomial describing the sensor's nonlinearity. He concludes that this approach is very accurate and requires only a few basic mathematical operations, making it suitable for implementation within FPGAs.

If a sensor is used outside of its calibrated range for temperature and humidity, the indicated values might not match the calibrated ones, due to new errors in both accuracy and precision. Even if a sensor is calibrated, changing ambient conditions can affect provided readings. This has to be kept in mind, to avoid unexpected drifting and other issues, e.g. by adjusting the expected precision accordingly. Depending on sensor and measurand, self-calibration might be supported and automatically triggered, whenever the values returned by the sensor seem questionable. Such recalibrations and other strategies are presented and compared in [PPG05].

Besides ambient conditions, the sensor's error can be affected by multiple other factors. Clausen et al. [Cla+17] describe various sources, influencing accuracy and precision of a gyroscope and how to address them via calibration in a stochastic manner. Furthermore, they envision to apply their approach over time, to compensate for changes of stochastic environmental influences on the sensor. Additional calibration schemes, as well as their advantages and disadvantages can be found in [LC13; JP04; Ols+16; SB13].

The sensor's expected precision, or uncertainty, within each measurement is usually determined during calibration, but may also be estimated on its own, when the calibration results are unknown or undisclosed. Latter is important, as many (smartphone) sensors are factory calibrated, but their precision is unknown to the user. It can e.g. be estimated by taking several

samples of a constant measurand and creating a histogram, as depicted in figure 2.1. According to the central limit theorem, the histogram usually follows a normal distribution, with σ denoting the precision, and, for calibrated sensors, its mean equal to the measurand's true value [Smi99]. While σ often is a constant value throughout the whole range of the sensor – same uncertainty when measuring e.g. $0\,°\text{C}$ or $100\,°\text{C}$ – the applied calibration can modify the constant throughout the measuring range. Quantization noise, for example, induced by the analog-to-digital converter (ADC) present within most sensors, is an additive zero mean uniform noise, independent of the magnitude of the to-be-digitized value x, with the simplified representation

$$f_{\text{quant}}(x) = \lfloor x + 0.5 \rfloor\,, \quad \varepsilon_{\text{quant}} = (x - f_{\text{quant}}(x)) \sim \overbrace{\mathcal{U}(-0.5, +0.5)}^{\text{uniform distribution}}. \tag{2.1}$$

When a calibration function $f_{\text{calib}}(f_{\text{quant}}(x))$ performs (non-linear) scaling on the quantized result, $\varepsilon_{\text{quant}}$ is also scaled, yielding a change in precision throughout the measuring range.

Often, the precision can be increased, e.g. by averaging several measurements. Success, however, is strongly related to the initial cause of the precision error. Depending on the used measurement hardware and the measurand, various sources for stochastic and non-stochastic errors exist. While typical sampling errors, like quantization noise, can be reduced by averaging, (temporal) environmental influences like temperature, humidity or ambient surroundings, can not [Cla+17]. Furthermore, as a moving average filter is the same as a convolution with several constants, that is, a rectangle, it introduces a delay to the filtered output [Smi99]. Especially for sensors with low sample rates, this might introduce new use case dependent issues. Success of averaging, and other filtering approaches thus strongly depends on the sample rate of the sensor, the amount of noise present, and the delay introduced by the filter.

2.2 Probabilistic Problem Formulation

Instead of reducing the error via filtering, potentially introducing new issues, inaccuracies can be included within calculations, using probabilities for all indicated values, based on the sensor's known precision. Assuming the GPS to indicate a current location ϱ using Cartesian coordinates $(x, y, z)^T$ with an estimated error of $3\,\text{m}$. When observations are provided only once per second [TY09], averaging increases delays beyond limits acceptable for car navigation. Instead, the $3\,\text{m}$ uncertainty can be addressed probabilistically, assigning a certain likelihood to the indicated location and its vicinity. While the observation from the GPS might describe the most likely whereabouts of the receiver, depending on the error, surroundings are likely as well.

From the user's point of view, this might read as: *"assuming I am currently here, how likely is it to receive the values that are currently indicated by the GPS?"*, that is

$$p(\boldsymbol{\varrho} \mid \tilde{\boldsymbol{\rho}}), \quad \boldsymbol{\varrho}, \tilde{\boldsymbol{\rho}} = (x, y, z)^T, \tag{2.2}$$

the probability of the GPS indicating $\boldsymbol{\varrho}$ as current location while actually residing at $\tilde{\boldsymbol{\rho}}$. Yet, this formulation is not limited to 3D positions and GPS sensors. In general, an *observation* $\boldsymbol{o}$ provided by a sensor, yields a hint on some current *state* $\boldsymbol{q}$, which e.g. refers to a 3D location, a current speed, heading, or other metric relevant to a given problem. Matching with the recursive nature briefly mentioned in the introduction (cf. figure 1.3), both, observation and state, are time-dependent, and there is not a single instance, but many, belonging to different points in time. They are therefore referred to as $\boldsymbol{o}_t$ and $\boldsymbol{q}_t$, and are part of a time series

$$\begin{aligned}
\langle \boldsymbol{o} \rangle_t &= \boldsymbol{o}_{1:t} = \boldsymbol{o}_1, \ldots, \boldsymbol{o}_{t-1}, \boldsymbol{o}_t \quad \text{with} \quad \langle \boldsymbol{o} \rangle_t = \langle (\ldots) \rangle_t \\
\langle \boldsymbol{q} \rangle_t &= \boldsymbol{q}_{0:t} = \boldsymbol{q}_0, \ldots, \boldsymbol{q}_{t-1}, \boldsymbol{q}_t \quad \text{with} \quad \langle \boldsymbol{q} \rangle_t = \langle (\ldots) \rangle_t,
\end{aligned} \tag{2.3}$$

where $\boldsymbol{o}_1$ is the first observation at time $t = 1$, $\boldsymbol{o}_t$ the current observation at time t, and $\boldsymbol{o}_{t-1}$ the previous one. The general version of (2.2) is thus given by the probability (2.4), of receiving sensor observations $\boldsymbol{o}_t$ at a point t in time, given some state $\boldsymbol{q}_t$

$$p(\boldsymbol{o}_t \mid \boldsymbol{q}_t). \tag{2.4}$$

As (2.4) uses a direct comparison between a state and various sensor observations, it is limited to sensors providing absolute values concerning the problem, like the GPS when questioning the current location. Referring to the introduction, velocity readings, from e.g. a speedometer, denote a relative indication. The currently indicated speed can not be directly compared against a potential location, but only against a change in location. Relative sensor observations can be included by considering both, the current state $\boldsymbol{q}_t$ and the previous one $\boldsymbol{q}_{t-1}$, yielding

$$p(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}). \tag{2.5}$$

(2.5) allows for both, absolute and relative comparisons, hereafter referred to as *evaluation*, as it evaluates the probability for observations $\boldsymbol{o}_t$, given a state $\boldsymbol{q}_t$, or change in state $\boldsymbol{q}_{t-1} \rightarrow \boldsymbol{q}_t$.

The point of view can be inverted to *"assuming the previous state was $\boldsymbol{q}_{t-1}$, and the previous sensor observations were $\boldsymbol{o}_{t-1}$, what could the next state $\boldsymbol{q}_t$ look like?"*, written as

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}), \tag{2.6}$$

and hereafter referred to as *transition*, as it describes potential transitions from a previous state $\boldsymbol{q}_{t-1}$ into a new state $\boldsymbol{q}_t$, given some sensor observations $\boldsymbol{o}_{t-1}$. The minor difference of (2.5)

using $\boldsymbol{o}_t$ and (2.6) using $\boldsymbol{o}_{t-1}$ as observation, is due to mathematical definition, and can be considered to be identical [TBF05]. While both viewpoints hereafter appear to be the same, there is an important difference between both, that will become relevant within chapter 4.

Due to essential differences in requirements for absolute and relative sensors, the contents of $\boldsymbol{o}_t$ and $\boldsymbol{q}_t$ are dependent on available sensors, and the actual problem formulation. Throughout this work, $\boldsymbol{o}_t$ contains values provided by various sensors installed within commodity smartphones, discussed within this chapter. The state $\boldsymbol{q}_t$ contains attributes needed to locate, track or navigate a pedestrian within a building. For single floors, this covers at least the current 2D location $(x, y)^T$. For multistory buildings, $(x, y, z)^T$ is required, to include the current floor. Depending on the sensors available within a phone, additional attributes, such as the current walking direction Θ, can also be part of the state. When individual values from the state or observation are used within equations, this is indicated by e.g. $q_t^{(x)}$ from $\boldsymbol{q}_t$, or $o_{t-1}^{(\Theta)}$ from $\boldsymbol{o}_{t-1}$.

The following sections focus on several sensors available within modern smartphones, providing valuable information towards localization and navigation within a building. For every sensor, its group (absolute/relative), potential calibration approaches, to be expected errors, and contributions towards the overall system are examined. Depending on the sensor's group and use case, probabilistic models for (2.4), (2.5) and (2.6) are derived and discussed in detail.

2.3 Global Positioning System

As of today, the global positioning system (GPS) is a well-known method for outdoor localization almost anywhere on earth. Being expensive at first, receivers are now available as cheap modules, installed within tracking devices, smartphones, and portable navigation systems, offering outdoor localization on land, sea and in the air. While GPS does not work indoors [Och+14; GGB12], the sensor is still valuable for indoor localization and navigation, e.g. directly before entering, or when walking between adjacent buildings [Hut+16; CPP10; Tor+17]. Furthermore, many aspects discussed for the GPS apply to other localization components as well.

To provide location information, a receiver listens for data frames from moving satellites, equipped with high-precision atomic clocks. Each frame contains the satellite's current position and the timestamp it was sent at. Combining the frames from several satellites allows the receiver to use the time difference of arrival (TDOA) method, to infer the time needed for the signals to travel from each satellite, hereafter converted into a distance, based on the speed of light. Each estimated distance denotes the radius of a sphere around the known position of its corresponding satellite. The receiver's location can be inferred using multiple measurements, and resides where all their spheres intersect. For a 3D estimation (including altitude) at least four measurements are required. This process is known as lateration or multilateration [DH10].

Multilateration   Due to measurement noise, however, the distance information is inaccurate, and there is no exact point of intersection between all surfaces. The receiver's whereabouts are thus approximated by determining the location that fits all measurements best. Without loss of generality, using Cartesian 3D coordinates $(x, y, z)^T$ instead of latitude, longitude and altitude: The relationship between the receiver's real location $\tilde{\varrho}$, its estimated position $\hat{\varrho}$, the position ϱ_{sat} of some satellite and the measured distance d_{sat} towards the satellite is constrained by

$$\|\overbrace{\varrho_{\text{sat}}}^{\text{known}} - \overbrace{\tilde{\varrho}}^{\text{unknown}}\| \approx \overbrace{d_{\text{sat}}}^{\text{measured}} \overset{!}{=} \overbrace{\|\varrho_{\text{sat}} - \hat{\varrho}\|}^{\text{calculated}}, \quad \|\rho\| = \sqrt{\left(\rho^{(x)}\right)^2 + \left(\rho^{(y)}\right)^2 + \left(\rho^{(z)}\right)^2}. \qquad (2.7)$$

The location ϱ_{sat} of every satellite is known from transmitted data, and d_{sat} represents the measured distance towards it, given by the TDOA. Using several of above constraints, the receiver's estimated location ϱ^* is assumed to be the location that minimizes the (quadratic) error between the measured and the calculated distance when assuming the receiver to reside at $\hat{\varrho}$

$$\varrho^* = \arg\min_{\hat{\varrho}} \sum_{\text{sat}} \left(\|\varrho_{\text{sat}} - \hat{\varrho}\| - d_{\text{sat}}\right)^2. \qquad (2.8)$$

Assuming (2.8) to be a continuous and convex function, ϱ^* can be determined using numerical optimization strategies such as gradient descent [HS06; RHG13] or the Nelder-Mead method, also called downhill simplex method [NM65; Pow62]. These try to estimate a function's global minimum (or maximum) using multiple iterations, like known from Newton's method [New67]. If the given function is not convex or non-monotonic, they face the risk of getting stuck within a local minimum instead, thus being only suited for certain types of functions.

Alternatively, (2.8) can be solved using an analytical approximation. Applying a linearization to the quadratic equation (2.7), allows for rewriting the problem as a linear system $Ax = b$ [GV13]. For better readability, the linearization is presented in general, relating distances d_i measured towards known locations $(x_i, y_i, z_i)^T$ to the unknown location $(\hat{x}, \hat{y}, \hat{z})^T$

$$d_i \overset{!}{=} \sqrt{(x_i - \hat{x})^2 + (y_i - \hat{y})^2 + (z_i - \hat{z})^2}, \quad i \in \{1, \ldots, N\}. \qquad (2.9)$$

Squaring (2.9) to remove the root, and expanding the parentheses yields

$$d_i^2 = (x_i^2 - 2x_i\hat{x} + \hat{x}^2) + (y_i^2 - 2y_i\hat{y} + \hat{y}^2) + (z_i^2 - 2z_i\hat{z} + \hat{z}^2). \qquad (2.10)$$

For a system of linear equations, the three unknowns with exponent, $\hat{x}^2$, $\hat{y}^2$ and $\hat{z}^2$, must be removed from (2.10). One common solution is to subtract the first equation, the one that belongs

to d_1, from all others $\{d_2, \ldots, d_N\}$, resulting in a system of $N-1$ equations [Li+05]:

$$
\begin{aligned}
d_i^2 - d_1^2 &= \left((x_i^2 - 2x_i\hat{x} + \hat{x}^2) + (y_i^2 - 2y_i\hat{y} + \hat{y}^2) + (z_i^2 - 2z_i\hat{z} + \hat{z}^2)\right) - \\
&\quad \left((x_1^2 - 2x_1\hat{x} + \hat{x}^2) + (y_1^2 - 2y_1\hat{y} + \hat{y}^2) + (z_1^2 - 2z_1\hat{z} + \hat{z}^2)\right) \\
&= x_i^2 - x_1^2 + 2\hat{x}(x_1 - x_i) + y_i^2 - y_1^2 + 2\hat{y}(y_1 - y_i) + z_i^2 - z_1^2 + 2\hat{z}(z_1 - z_i),
\end{aligned}
\tag{2.11}
$$

which can hereafter be rewritten as a system of linear equations, $\boldsymbol{Ax} = \boldsymbol{b}$

$$
\begin{pmatrix}
x_1-x_2 & y_1-y_2 & z_1-z_2 \\
x_1-x_3 & y_1-y_3 & z_1-z_3 \\
& \vdots & \\
x_1-x_i & y_1-y_i & z_1-z_i
\end{pmatrix}
\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix}
= \frac{1}{2}
\begin{pmatrix}
(d_2^2 - d_1^2) + (x_1^2 - x_2^2) + (y_1^2 - y_2^2) + (z_1^2 - z_2^2) \\
(d_3^2 - d_1^2) + (x_1^2 - x_3^2) + (y_1^2 - y_3^2) + (z_1^2 - z_3^2) \\
\vdots \\
(d_i^2 - d_1^2) + (x_1^2 - x_i^2) + (y_1^2 - y_i^2) + (z_1^2 - z_i^2)
\end{pmatrix}.
\tag{2.12}
$$

When more than four equations are given, the system is overdetermined and (2.12) can be solved using $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b}$, where $\boldsymbol{A}^+$ is the Moore–Penrose inverse $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ [Pen55]. This yields an approximate solution $\boldsymbol{\varrho}^* = (\hat{x}, \hat{y}, \hat{z})$ that, similar to (2.8), minimizes the quadratic error between measured and calculated distances. However, due to the linearization via subtraction, the result represents an approximation, slightly differing from the one provided by (2.8) [Li+05].

For simplicity, above calculations were based on 3D Cartesian coordinates. The values $\boldsymbol{g}$ provided from actual GPS sensors, however, are polar coordinates with *latitude* $g^{(\text{lat})}$, *longitude* $g^{(\text{lon})}$, an *altitude* $g^{(\text{alt})}$ and an error estimation $g^{(\text{err})}$, e.g. returned by the *NMEA* protocol [AAO11] of portable receivers, or smartphone operating system APIs. The longitude is given between $-180°$ and $180°$ or from west to east. When walking along the equator, only the longitude is changed. It will hereafter be referred to as the x-axis within the building's coordinate system. The latitude runs from $-90°$ to $90°$ or from south to north, and is referred to as y-axis.

Floorplan Mapping For being used by indoor localization, coordinates supplied by the GPS must be converted to the coordinate system of the floorplan. This requires a calibrated reference $(\text{ref}_{\text{lon}}, \text{ref}_{\text{lat}}, \text{ref}_{\text{alt}}) \rightarrow (\text{ref}_{\text{x}}, \text{ref}_{\text{y}}, \text{ref}_{\text{z}})$, e.g. for the building's center, and a rotation angle ref_α, aligning the x and y axes with longitude and latitude, depending on the building's orientation. As the GPS is only intended for outdoor use, altitude information can usually be omitted. Hereafter, changes in longitude and latitude around their reference correspond with changes in x and y around the building's center. While the transformation between both is nonlinear due to the earth's curvature, it can be assumed as linear throughout the relatively small size of the building. Using the earth's perimeter around the equator and converting this value to $\text{m}/°$

$$
\frac{2\pi r_{\text{earth}}}{360°} = \frac{400\,007\,862\,\text{m}}{360°} = 111\,132.95\,\text{m}/°\,,
\tag{2.13}
$$

yields a constant, relating changes of ° in latitude to m, valid at $\mathrm{alt} = 0\,\mathrm{m}$. As the earth's latitudinal-perimeter decreases when moving towards the poles, the reference for changes in longitude along this perimeter is varying, dependent on the current latitude:

$$f_{\mathrm{lat}}\left(\Delta\mathrm{lat}\right) = \Delta\mathrm{lat}\,111\,132.95\,, \quad f_{\mathrm{lon}}\left(\Delta\mathrm{lon}\right) = \Delta\mathrm{lon}\big(\cos(\mathrm{ref}_{\mathrm{lat}})\,111\,132.95\big)\,. \tag{2.14}$$

As the earth is not perfectly round, but an ellipsoid, (2.14) represents a simplification. Usually several compensation terms are added to the equation to improve correctness [MS10]. Alternatively, other approaches, such as the one presented by Vincenty [Vin75], can be used. However, using this linear approach with a constant value, yields a computationally efficient, yet sufficiently accurate solution for the intended use case. The final conversion from polar coordinates within $\boldsymbol{g}$ to Cartesian coordinates, including the calibrated reference, is hereafter given by

$$\mathbf{pos}_{\mathrm{gps}}\left(\boldsymbol{g}\right) = \begin{pmatrix} \mathrm{ref}_{\mathrm{x}} \\ \mathrm{ref}_{\mathrm{y}} \\ \mathrm{ref}_{\mathrm{z}} \end{pmatrix} + \begin{pmatrix} \cos\left(\mathrm{ref}_{\alpha}\right) & -\sin\left(\mathrm{ref}_{\alpha}\right) & 0 \\ \sin\left(\mathrm{ref}_{\alpha}\right) & \cos\left(\mathrm{ref}_{\alpha}\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_{\mathrm{lon}}\left(g^{(\mathrm{lon})} - \mathrm{ref}_{\mathrm{lon}}\right) \\ f_{\mathrm{lat}}\left(g^{(\mathrm{lat})} - \mathrm{ref}_{\mathrm{lat}}\right) \\ \left(g^{(\mathrm{alt})} - \mathrm{ref}_{\mathrm{alt}}\right) \end{pmatrix}\,. \tag{2.15}$$

Independent of the strategy used to estimate the location $\boldsymbol{\varrho}$ of the GPS receiver, it will deviate from the device's actual position. Depending on satellite visibility, quality of the time measurements and used hardware components, the sensor's precision and accuracy will vary. Most GPS receivers provide an error estimation $g^{(\mathrm{err})}$ (in m), that yields an indication on the quality of the current location estimation. This value is determined by the sensor itself, in a similar way to examining the variance among all individual distance errors from (2.8):

$$\sigma_{\mathrm{gps}}^2 = \mathbb{E}(\mathcal{X}^2) - (\mathbb{E}(\mathcal{X}))^2\,, \quad \mathcal{X} = \left\{\left|\,\|\boldsymbol{\varrho}_{\mathrm{sat}} - \boldsymbol{\varrho}^*\| - d_{\mathrm{sat}}\right| \mid \forall\mathrm{sat}\right\}\,. \tag{2.16}$$

The resulting error indication can be used to describe the likelihood of the receiver residing directly at, or near the estimated position, also addressing bad reception conditions when buildings or trees occlude the satellite signals [AAO11]. Based on the notation introduced in section 2.2

$$\begin{array}{c} \overbrace{p_{\mathrm{gps}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right)}^{\text{GPS evaluation}} = \overbrace{\mathcal{N}\left(d \mid 0, \sigma_{\mathrm{gps}}^2\right)}^{\text{normal distribution}}\,, \quad \overbrace{d = \|\mathbf{pos}_{\mathrm{gps}}\left(o_t^{(\boldsymbol{g})}\right) - \mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right)\|}^{\text{difference between GPS indication and unknown state}} \\[2ex] \underbrace{\langle\boldsymbol{o}\rangle_t = \langle(\boldsymbol{g}, \ldots)\rangle_t}_{\text{observations (2.3)}}\,, \quad \underbrace{\langle\boldsymbol{q}\rangle_t = \langle(x, y, z, \ldots)\rangle_t}_{\text{unknown state (2.3)}}\,, \quad \underbrace{\boldsymbol{g} = (\mathrm{lon}, \mathrm{lat}, \mathrm{alt}, \mathrm{err})}_{\text{GPS receiver provides}}\,, \quad \underbrace{\sigma_{\mathrm{gps}} = g^{(\mathrm{err})}}_{\text{uncertainty}}\,, \end{array} \tag{2.17}$$

(2.17) denotes the corresponding evaluation (2.4). It describes a probabilistic relation between potential pedestrian whereabouts $(x, y, z)^T$, part of the unknown state $\boldsymbol{q}_t$, and current GPS readings $\boldsymbol{g}$, part of the observation $\boldsymbol{o}_t$. This probability depends on the distance between both, es-
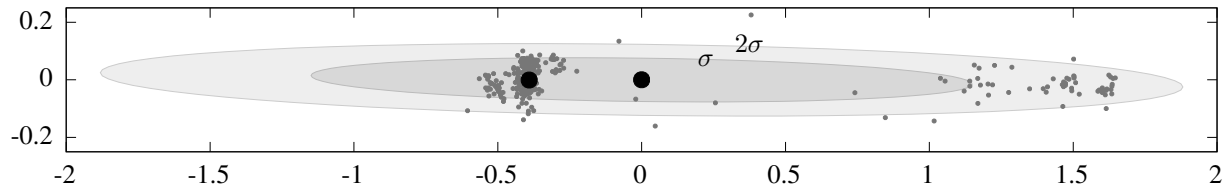
Figure 2.3: GPS localization deviation (in m) for a static receiver under good reception conditions. Assuming normally distributed measurements, the two ellipses depict the corresponding confidence intervals for $\sigma(\approx 68\,\%)$ and $2\sigma(\approx 95\,\%)$. Shape and rotation depend on the actual location on earth. Axis labels are therefore omitted. The median (left dot) does not resemble the average (centered dot).

timated after converting the GPS readings to the building's Cartesian coordinate system using (2.15), and extracting $(x, y, z)^T$ from the unknown state

$$\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right) = \left(q_t^{(x)}, q_t^{(y)}, q_t^{(z)}\right)^T .\tag{2.18}$$

That is, (2.17) relates unknown states and the observation from the GPS by using a normal distribution based on their distance, and the uncertainty indication $g^{(\mathrm{err})}$, given by the receiver.

Sensor Uncertainty   For many sensors and use-cases, their error/noise is assumed to follow a zero mean Gaussian distribution, like in (2.17) [Smi99]. This implies, that the values returned by some sensor are symmetrically distributed around the real value, and the average of an infinite number of measurements, produces the real value. While being a correct assumption for most calibrated sensors, aforementioned environmental conditions can affect the measurement noise. Especially a bad line of sight towards the sky can greatly affect the accuracy of the location estimations returned by a GPS receiver [Och+14; DH10]. Such conditions can yield a non zero mean measurement noise, where all estimations drift from the actual location. Figure 2.3 depicts several measurements from a static receiver with a clear line of sight towards the sky. Even under ideal conditions, they can be influenced and might be shifted from the actual location. Comparing their median (left dot) and average (centered dot) indicates that this is the case for the depicted observations. Still assuming a zero mean Gaussian noise in such cases, yields an error, as the density is shifted from the real value.

With the direction of the shift unknown, and the sensor only providing a hint on its amount, a uniform distribution comes to mind as potential alternative. However, while this ensures that every location within the confidence-radius around the sensor's estimation has the same probability, all values outside of this region receive a likelihood of zero, which is usually incorrect. This can be addressed by combining both aforementioned distributions: A uniform distribution for all values within the range of the indicated error $g^{(\mathrm{err})}$ and a normal distribution for all values outside of this region. The transition between both distributions should be continuous and the
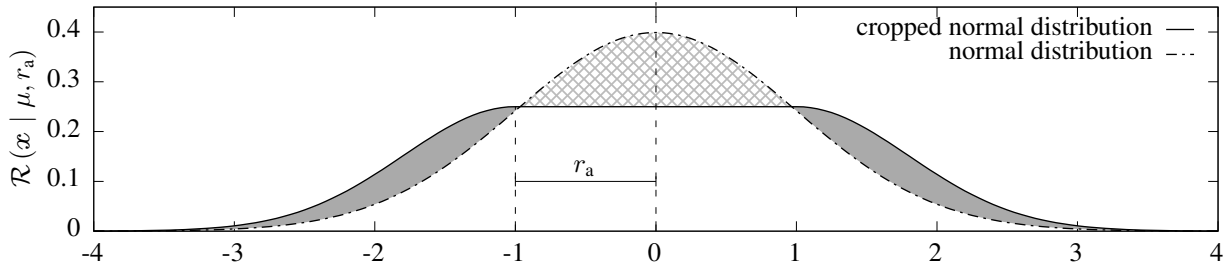
Figure 2.4: Modification of the normal distribution, created by inserting a uniform distribution in the center. The filled and crossed parts denote the redistribution, ensuring the result sums to 1.0.

area of the resulting probability density function (PDF) must satisfy

$$\int_{-\infty}^{\infty} p(x) \ \mathrm{d}x = 1.0 \,. \tag{2.19}$$

To meet those requirements, the overall density can e.g. be conceived as an equal split between the uniform and the normal distribution, shown in figure 2.4. The true location often resides somewhere within the region of $\pm g^{(\mathrm{err})}$ around the indicated value, or $\mu \pm r_{\mathrm{a}}$ in general. This is covered by a uniform distribution $\mathcal{U}(\mu - r_{\mathrm{a}}, \mu + r_{\mathrm{a}})$ around the center μ. The width $2r_{\mathrm{a}}$ of the distribution is variable and depends on the error currently indicated by the receiver. Considering just the uniform distribution, the height r_{h} must be chosen to ensure the area satisfies (2.19), that is: $2\,r_{\mathrm{a}}\,r_{\mathrm{h}} = 1.0$. To seamlessly connect both distributions, the normal distribution's highest point has to be on par with the height of the uniform distribution. Its σ is thus constrained by

$$r_{\mathrm{h}} \stackrel{!}{=} \max\left(\mathcal{N}(\ldots, \sigma^2)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \quad \Rightarrow \quad \sigma = \frac{1}{\sqrt{2\pi}r_{\mathrm{h}}} \,. \tag{2.20}$$

The combined distribution is calculated by making each input variable x zero mean, conditioning the next step on whether this intermediate value resides within the uniform or within the normal distribution, and normalizing the result by 0.5, to satisfy (2.19), summing up to 1.0

$$\mathcal{R}\left(x \mid \mu, r_{\mathrm{a}}\right) = \frac{1}{2} \begin{cases} \mathcal{U}(x' \mid -r_{\mathrm{a}}, +r_{\mathrm{a}}) & \text{for } x' < r_{\mathrm{a}} \\ \mathcal{N}\left(x' - r_{\mathrm{a}} \mid 0, \sigma^2\right) & \text{else} \end{cases} \tag{2.21}$$

$$x' = |x - \mu| \,, \quad r_{\mathrm{h}} = \frac{r_{\mathrm{a}}}{2} \,, \quad \sigma = \frac{1}{\sqrt{2\pi}r_{\mathrm{h}}} \,.$$

When using (2.21), the evaluation (2.17) of the pedestrian's whereabouts is rewritten as

$$p_{\mathrm{gps}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \mathcal{R}\left(d \mid 0, g^{(\mathrm{err})}\right) \,. \tag{2.22}$$

One major issue presented by the GPS is the time needed until the first fix, or location information, is provided. Especially when leaving a building to move between adjacent complexes, the

amount of time can be too short for a fix to become available. Furthermore, in case of adjacent buildings, the signal might be shadowed, increasing the time even further, hereafter providing only coarse location estimations due to bad satellite visibility [DH10; Ebn+17]. Another drawback is presented by increased energy consumption when a smartphone's GPS is active. Thus, it should only be enabled when it is expected to provide viable results, that is, outside of a building [Cap+17]. Therefore, Zhou et al. [Zho+12] proposed a system that uses other smartphone components, such as the light sensor, or the celltower signal strength, to distinguish between indoors and outdoors. However, they also denote that the accuracy outdoors will greatly vary depending on surroundings and weather conditions. For large error indications it thus makes sense to ignore readings provided by the GPS, and rely on other sensors instead.

2.4 Inertial Measurement Unit

As mentioned initially, a navigation system for cars benefits from additional information besides the GPS, such as the current driving speed or direction. Both values can be used to stabilize the localization when the GPS is unreliable, or to provide location updates whenever it is unavailable, e.g. when driving through a tunnel. A similar procedure has been used by seafarers for decades, estimating the ship's current location based on its revolution counter and a compass, when astronomical fixes, such as stars, were invisible or ambiguous [Ser28]. Approaches based on those two parameters, speed and heading, are known as inertial navigation systems INS, or dead reckoning. They predict the current position based on the previously known position and heading, using recent information for movement/speed and heading/direction-changes, thus representing a relative location estimation. It also applies to aviation, where the current airspeed, measured by Pitot tubes, is combined with the heading provided by a magnetic compass and gyroscope, to navigate relatively between landmarks with known absolute locations [Pit32]. When the initial location and heading are known, dead reckoning can be applied for pedestrian indoor navigation and localization as well. In many scenarios a pedestrian enters a building at a known location, or this location can be provided by the last GPS readings, just before entering the building [Hut+16; Och+14]. Starting from this known state, dead reckoning can be used to update the pedestrian's location based on relative movements. This itself is a well established field of research, known as pedestrian dead reckoning (PDR).

Almost all modern smartphones contain a so-called inertial measurement unit (IMU), which is a group of several individual sensors, related to inertial measurements. As of today, the IMU within most devices provides an accelerometer for absolute gravity/pose measurements, a gyroscope for relative pose changes, and a magnetometer for absolute magnetic field measurements. Based on those three sensors, it can be inferred whether the pedestrian is currently walking,

taking stairs, or changing the heading. While walking speeds can roughly be approximated [Yu+19], the quality strongly depends on how the pedestrian holds the smartphone while walking. Dedicated foot-mounted accelerometers are more suited for this estimation, as the impact of acceleration is very pronounced, and can be used to estimate the distance taken with every single step [NPM13]. This, however, is impracticable when dealing with smartphone-based indoor localization and navigation, where the pedestrian often holds the phone in front of the body. Here, the required forward acceleration is mainly measurable when starting or stopping to walk, and can easily be confused with shaking the phone during the walk [KWS12].

While aforementioned velocity estimation is rather unstable, single steps made by the pedestrian create a measurable change in acceleration, representing the base for *step-detection*. Combined with average human step lengths, the walking speed is approximated [TS12; Köp+14; SD16]. Likewise, the gyroscope measures relative changes in pitch, roll and yaw, and thus provides a corresponding *turn-detection* for the pedestrian [Ebn+15; Jan+15; Li+12]. Due to only capturing relative changes, the current absolute heading depends on the availability of the initial heading, which is hereafter adjusted by all measured changes. This drawback can be mitigated by readings from a magnetometer, which serves as an eCompass, providing a coarse information on the current *absolute heading*, yet, prone to errors e.g. induced by nearby metal objects [Goy+11; ND97; She+09; Goz+11].

Despite apparent simplicity, this topic is still under extensive research. Even though sensor accuracies are steadily increasing, filtering is still required. Not only to distinguish between actual data and sensor noise, but also to prevent temporal errors, like metallic objects influencing the compass, and to stabilize the resulting estimations over time [Tia+15; ND97]. While most of today's sensors provide very accurate readings, with small amounts of noise [Mou+15], their values do not allow for absolute location estimations, but only for relative changes, representing one of the major issues of dead reckoning. Even if individual readings only face a small error ε, they contribute to every relative adjustment and thus accumulate over time

$$d_{t+1} = d_t + (v + \varepsilon)\Delta t\,, \quad d_{t+2} = d_{t+1} + (v + \varepsilon)\Delta t \quad \rightarrow \quad d_{t+n} = d_t + n(v + \varepsilon)\Delta t\,. \quad (2.23)$$

Within (2.23), the distance d_t from an initially known position is adjusted at fixed time intervals Δt by using the current velocity-indication v, which faces an error ε. As mentioned earlier, this error is often assumed to be normally distributed. The resulting cumulative error E is thus given by a normal distribution, with mean and uncertainty dependent on the number of cumulations

$$\varepsilon_i \sim \mathcal{N}(\mu, \sigma^2)\,, \quad E = \sum_{i=1}^{N} \varepsilon_i \;\Rightarrow\; E \sim \mathcal{N}(N\mu, N\sigma^2)\,. \qquad (2.24)$$
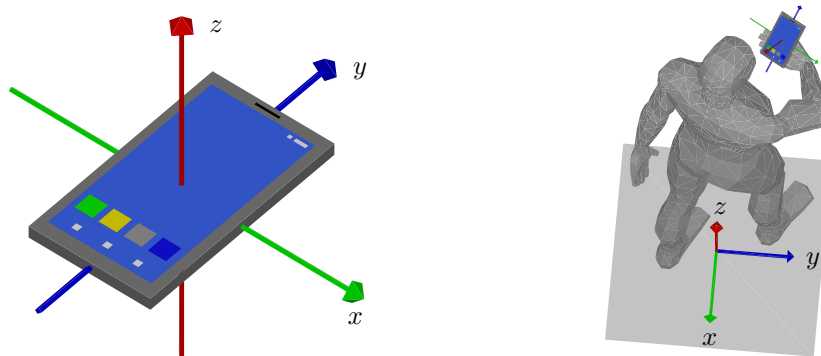
Figure 2.5: Coordinate system of most smartphone's IMU-sensors [Goob; App] (left). When the pedestrian holds the device in front, slightly tilting it for a good view angle, this coordinate system is rotated with respected to the world's coordinate system (right), affecting the readings from contained sensors.

For calibrated sensors, the error should be zero mean, and thus $N\mu = 0$. Cumulation still yields an average of zero and only the variance σ^2 increases linearly with N. That is, the result after cumulation is close to the true value, but its uncertainty has increased, yielding a lower precision. For most real-world scenarios, however, accuracy is not ideal and a constant offset μ cumulates over time, yielding results which are neither accurate, nor precise [Smi99].

Assuming an object to travel with $14\,\mathrm{m/s}$, and the error of its velocity sensor to be zero mean Gaussian with $\sigma = 0.5\,\mathrm{m/s}$. According to (2.24), after $120\,\mathrm{s}$ it traveled $1680\,\mathrm{m}$, with an uncertainty $\sigma = 5.5\,\mathrm{m}$. Based on the integral $\int_{-2\sigma}^{+2\sigma} \mathcal{N}(1680, 5.5^2)$, the real distance is within $(1680 \pm 11)\,\mathrm{m}$, with a confidence of $\approx 95\,\%$. Even though the uncertainty of $0.5\,\mathrm{m/s}$ and the timeframe of $120\,\mathrm{s}$ are rather large, $\pm 11\,\mathrm{m}$ is small compared to $1680\,\mathrm{m}$. However, if the sensor is slightly off, indicating $14.1\,\mathrm{m/s}$ instead, the object is expected to have moved $1692\,\mathrm{m}$ after $120\,\mathrm{s}$. The small offset of $0.1\,\mathrm{m/s}$ yields an average result, outside of the previous confidence interval of $(1680 \pm 11)\,\mathrm{m}$. Especially for problems with cumulative errors, such as dead reckoning, offsets in accuracy must be prevented. To address those offsets, the sensor requires calibration to determine both, the type (cf. figure 2.2) and amount of the induced error [Tia+15].

For constrained use-cases, where a cumulative error is tolerable, and both, starting position and heading, are well known, just using PDR without additional sensors might already be sufficient. State of the art PDR solutions provide viable results with minimal sensor requirements and little or no prior knowledge about the walkable area. For longer walks within complex architecture, however, PDR will suffer from increasing errors and requires stabilization using absolute location information [CHP16; JPP18; Kna17].

To use a smartphone's IMU for PDR, the observations from its sensors must be interpreted correctly. As of today, the contained accelerometers, gyroscopes, and magnetometers all provide readings for three orthogonal axes. Typically, all three sensors are placed in the same way,

sharing the same alignment of these axes. Figure 2.5 depicts the coordinate system used within most smartphones [Goob; App]. When the device is placed display-up on a table, the sensors are aligned parallel to the earth's surface, with the z-axis pointing upwards, and measurements from the accelerometer are approximately $\approx (0, 0, 9.81)\,\mathrm{m/s}^2$. Within the following discussions, the pedestrian is expected to hold the smartphone in one hand in front of the body, slightly tilting it for a good display viewing angle (cf. figure 2.5). This pose is expected to be typical for most pedestrians using indoor navigation, watching routing advice on the device's screen. Due to the tilt, the smartphone's coordinate system is rotated, affecting the readings provided by all IMU sensors. This rotation must be considered to correctly interpret sensor readings, aligning them with the coordinate system assumed for the world, that is, the building's floorplan.

The following sections discuss IMU-based approaches that are viable for a smartphone-based indoor localization and navigation system. This covers the three main topics of step-detection, turn-detection and eCompass. Handling the rotation induced by the pedestrian tilting the smartphone will be covered in detail. Discussions also include expected errors, to derive probabilistic models including each sensor's uncertainty. Where applicable, potential calibrations, to increase accuracy, are examined as well.

2.4.1   Step-Detection

As previously mentioned, determining the pedestrian's current walking speed based on available smartphone sensors is inaccurate, at least when the device is held upfront. This drawback is mitigated by step-detection, serving as an approximate velocity indicator for the dead reckoning process. When a pedestrian walks along a corridor, each step creates a measurable change in acceleration, once when a foot is lifted, and once when touching the ground [TS12]. Those changes yield a unique pattern, detectable for all smartphone poses, within the data provided by the accelerometer, measuring the current gravity.

Due to the IMU coordinate system (cf. figure 2.5), measurements $\boldsymbol{a} = (x, y, z)$ provided by the accelerometer are approximately $\boldsymbol{a} \approx (0, 0, 9.81)\mathrm{m/s}^2$ when the phone is placed display-up on the ground. When holding the device in the same way while walking, that is, parallel to the earth's surface, only the z-axis experiences aforementioned acceleration changes, and fluctuates around $9.81\,\mathrm{m/s}^2$, denoting individual steps. When navigating, the pedestrian usually looks at the smartphone's display by slightly tilting the upper screen edge towards the head, resulting in a rotation around the x-axis (see figure 2.5). After applying this rotation to the readings from the sensor, the x-value still stays near zero and only contains the natural left-right-fluctuation of humans moving their hands while walking. The earth's gravity, however, is redistributed among the y and z-axis, depending on the tilting angle, and the step-pattern can also be observed when
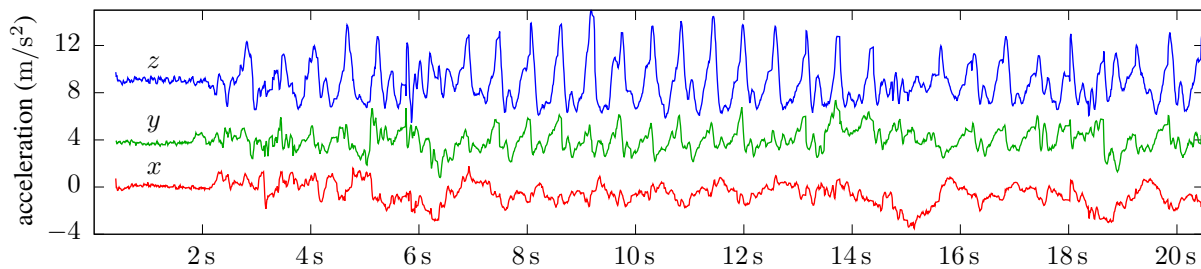
Figure 2.6: Readings from a smartphone's 3-axis accelerometer sensor over time, sampled at 20 Hz for a better visibility. The pedestrian walked along a hallway, holding the smartphone slightly tilted in front of the body (cf. figure 2.5), ≈ 1.3 m above ground. Each major spike on the z or y-axis denotes one single step. The x-axis is based on left-right-movements of the pedestrian's right arm during the walk.

examining the data of the y-axis. A matching example is shown within figure 2.6. While the used smartphone holding pattern is typical for indoor navigation, it can not be assumed to be a given, and the pedestrian might refer to other poses, such as landscape-mode, or carrying the phone within trouser pockets, listening to navigation advice via earphones [Kus+15].

As the direction of change in acceleration is not required for detecting steps, rotation invariancy is achieved by using the *magnitude* of the sensor readings [SD16]. That is, the Euclidean norm $\|(a^{(x)}, a^{(y)}, a^{(z)})^T\|$ of the measurement. This value still yields an average of ≈ 9.81 m/s^2, and preserves the deviations imposed by each step. However, this will usually introduce some side-effects and generate spikes that are not related to actual steps, e.g. when the pedestrian shakes the device, or changes the orientation from landscape to portrait or vice versa [Ebn+15].

Alternatively to using the magnitude, the accelerometer readings can be *transformed*. This is achieved by undoing the rotation introduced by the pedestrian tilting the device. Hereafter, the adjusted sensor readings are similar to the ones from the phone placed parallel to the ground, where $a^{(x)}$ and $a^{(y)}$ are almost 0 and $a^{(z)}$ fluctuates around 9.81 m/s^2, containing the typical step pattern [Tia+15]. Mathematical details on how to undo the rotation induced by the pedestrian holding the smartphone will be discussed in section 2.4.2. When e.g. carrying the phone within trouser pockets, this rotation is not constant, but changes throughout every step. Furthermore, vibrations and the effect of acceleration changes are different when the device is carried in this way. Park et al. [PHP17] suggest additional machine learning via SVM classification to detect potential smartphone poses, improving rotation invariance and the overall detection of steps.

Independent of the approach chosen for rotation invariance, the resulting data features a step-specific pattern, consisting of a steep increase, followed by a steep decrease [TS12; Li+12]. Basic strategies thus e.g. determine peaks that are above an empiric threshold. If a peak is detected, the process is stopped for some time, to prevent multiple small peaks in sequence, usually vibration or noise, from also triggering detected steps. This blocking-period is a heuristic, dependent on the expected step frequency [PHP17]. The requirement for this delay can clearly

be identified when looking at the z-axis signal course in figure 2.6. While the steps between $7\,\mathrm{s}$ and $14\,\mathrm{s}$ are relatively clear, many occurrences of minor local peaks can be observed as well (around $6\,\mathrm{s}$ or $19\,\mathrm{s}$), yielding too many step-detections when not being suppressed.

More robust approaches focus on finding local maxima within a certain timeframe. Using a window of several samples provides the same effect as aforementioned delay times, preventing several nearby peaks from triggering too many detected steps [TS12]. Besides local maxima, zero crossings can also be used to realize step detection, where two zero crossings denote one step. For this to work, readings from the accelerometer must be approximately zero mean. In case of the magnitude-based approach this is e.g. achieved by subtracting the constant gravity of $9.81\,\mathrm{m/s^2}$, or, in general, by subtracting a long-term average of the signal from itself [Kir+18]. Both techniques, detecting zero crossings and making a signal zero mean, are computationally inexpensive, thus ideal for use on embedded devices, such as smartphones [Goy+11].

Efficient Noise Reduction   As indicated in figure 2.6 and figure 2.7, there is a significant amount of noise within the measurements, affecting the step-detection process. Using zero crossings and local maxima detection mitigates the problem only to some degree. For a more robust stabilization, independent of the used detection method, the input signal should be filtered beforehand. With noise typically being a fast-changing component, a low-pass filter can be used to remove high frequencies from the signal [Kir+18]. The most simple low-pass is the moving average filter, using the average of several consecutive samples to suppress fast changes. While being computationally efficient, its configurability is rather limited.

For frequency-based filters to be used on a discretely sampled signal, it must be provided at a constant sample-rate [Smi99]. However, as neither sensor nor API of common smartphones are required to provide samples at equidistant intervals, a constant sample rate can not be assumed in general. This also imposes potential issues when searching for a specifically-shaped pattern within the provided sensor data. To construct a fixed sample rate, a combination of interpolation and resampling can be applied to the incoming measurements. Update rates provided by an accelerometer usually are high, $200\,\mathrm{Hz}$ and above, thus requiring for downsampling to lower rates, sufficient for the step-detection process. This is e.g. achieved by selecting the nearest neighbor, the reading that is temporally next to the required output, or by interpolating between the two nearest ones. If the sample rate of the sensor is much higher than the required output, using the nearest neighbor is sufficient in terms of precision [Smi99]. For all following steps, measurements are assumed to occur at a fixed rate of $100\,\mathrm{Hz}$, which is an empiric choice that is sufficiently fast to detect individual steps, yet conserving computational power [SD16; PHP17].

With the signal now available at a fixed sample rate, a configurable low-pass can be applied, by using either a finite impulse response (FIR) or an infinite impulse response (IIR) filter. Gen-
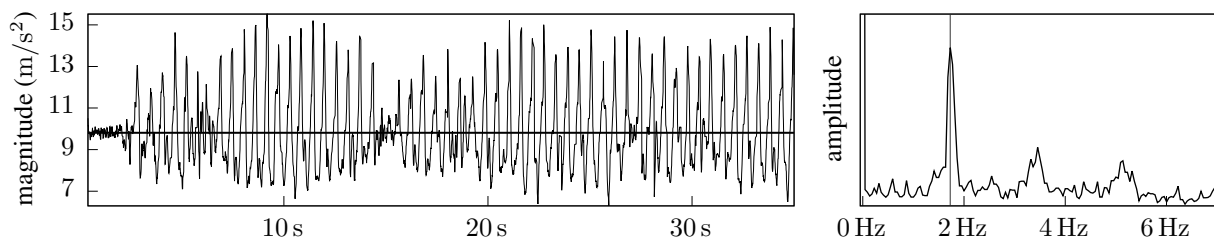
Figure 2.7: Magnitude of accelerometer readings, fluctuating around $9.81\,\mathrm{m/s^2}$, for a pedestrian walking along a hallway while holding the smartphone in front of the body (left). The corresponding frequency spectrum (right) denotes the signal's composition, and indicates the pedestrian's step-rate of $\approx 1.8\,\mathrm{Hz}$.

erally, the latter is computationally less complex, but the former provides better results. In this context, better refers to undesired frequencies being removed more effectively, without affecting the desired ones. That is, the cut-off between required and unwanted parts of the signal is more rapid, and the attenuation of the unwanted *stopband* is higher. However, while the IIR involves just a few multiplications and additions with small delays, the improved attenuation of FIR filters comes at the cost of required computational power, and larger delays [Smi99].

To be consistent with the notation typically used in digital signal processing, the following filter equations will re-purpose previously introduced variables.

A well known variant of IIR is the *second-order* or *biquad* filter, using three coefficients b_0, b_1, b_2 on the discrete input signal $x_{[]}$, and three a_0, a_1, a_2 on previous discrete outputs $y_{[]}$, thus creating an *infinite* recursion. An implementation in so-called *direct form 1* is given by

$$y_{[n]} = \frac{1}{a_0}\left(b_0 x_{[n]} + b_1 x_{[n-1]} + b_2 x_{[n-2]} \quad - a_1 y_{[n-1]} - a_2 y_{[n-2]}\right). \tag{2.25}$$

It only requires a few multiplications, additions and a history of two previous inputs and outputs, well suited for realtime use. The coefficients a_0, a_1, a_2 and b_0, b_1, b_2 are determined based on the desired effect. Noise can e.g. be removed from the input signal by configuring a low-pass filter, removing all frequencies above a certain cut-off frequency. The required values for the six components can directly be calculated based on the desired cut-off frequency, and sensor sample rate. Equations and further details are found in [Ror93; WT06; Smi99].

The low-pass' cut-off frequency depends on the pedestrian's walking speed, shown on the right side of figure 2.7. Its result is sufficient for most step-detection strategies, such as aforementioned peak detection. Others, like searching for zero-crossings, require the signal to be zero mean, with the constant gravity offset removed. To include this additional requirement, a band-pass filter can be used, keeping only the desired frequencies around $\approx 1.8\,\mathrm{Hz}$ (cf. figure 2.7). While all six coefficients are again directly calculable, there is less control on the actual size of the passband, which, however, is an important requirement to correctly adapt to significantly
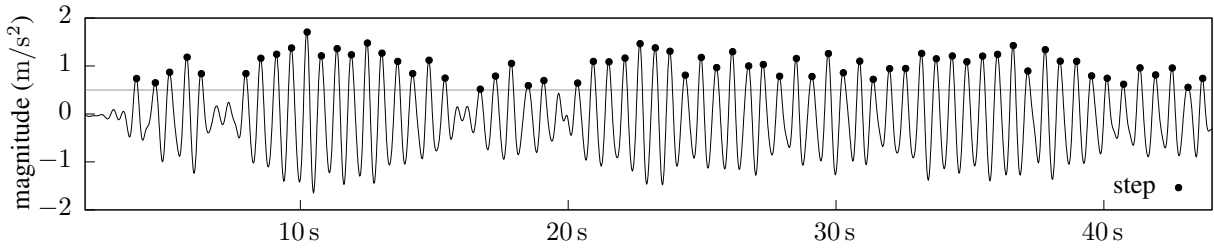
Figure 2.8: Band-pass filtered zero mean accelerometer magnitude, using the input from figure 2.7. The applied step-detection searches for local maxima that are above a certain threshold (grey line).

varying individual step frequencies [Sau+11]. More control is given by cascading multiple IIR filters. For example, one high-pass to remove the constant gravity and low frequency noise, like the pedestrian shaking the phone while walking, and one low-pass to suppress noise.

More versatile, in terms of configurability, are FIR filters, based on a discrete convolution of the input signal with some kernel. As a convolution in the time domain equals a multiplication in the frequency domain, the kernel can easily be created by choosing the frequencies to keep, and applying an inverse Fourier transform [Smi99]. The drawback of this approach is increased computational overhead. The filter's quality directly depends on the size kernel, which is thus required to have a decent size. Furthermore, the convolution introduces delays based on this size, delaying the output of the accelerometer, and thus the step-detection process. For example, a sample rate of $100\,\text{Hz}$ required $N \geq 101$ coefficients for the band-pass' kernel, yielding a $\frac{N-1}{2} \geq 500\,\text{ms}$ delay. Further details and considerations can be found in [Ror93; Smi99].

Besides using an IIR or FIR band-pass, low frequency noise and the constant gravity offset can also be removed by subtracting the input's moving average of size N' from it

$$y_{[n]} = x_{[n]} - \frac{1}{N'} \sum_{n'=0}^{N'} x_{[n-n']} \,, \tag{2.26}$$

hereafter sending the result through a low-pass filter to remove high frequency noise. While the moving average also introduces a delay based on N', it can be neglected, as the to-be-removed components (e.g. gravity) are approximately constant, not causing any delays for the actual step-pattern. Which of the discussed filtering variants serves best for an actual system, strongly depends on required quality, acceptable delays, and available computational performance.

Probabilistic Assembly   After filtering, one of the described strategies can be used to detect if the n-th sample denotes a step. A local maximum detection on the discrete, filtered magnitude

$$A_{[]} = \text{filter}\left(\|\boldsymbol{a}_{[]}\|\right) \tag{2.27}$$

requires values before and after the maximum for a correct identification, that is, future values, thus introducing slight delays to drop this requirement when evaluating live readings

$$f_{\text{step}}\left(A_{[\,]}, n\right) = \left(A_{[n-2\Delta]} < A_{[n-\Delta]}\right) \wedge \left(A_{[n-\Delta]} > A_{[n]}\right), \quad \Delta \in \mathbb{N}_{>0}. \tag{2.28}$$

Whether a detected maximum denotes a step also depends on its magnitude [Köp+14; PHP17]. This is depicted in figure 2.8, where only peaks above a certain threshold heuristic are classified as steps. Instead of a constant, heuristic threshold, the signal's moving variance could be used, detecting all peaks that are beyond the signal's average fluctuation. Yet, this imposes issues for situations where the pedestrian isn't walking. Standing still yields a low overall variance, and peaks induced by noise can cause false-positives, thus requiring additional handling [TS12].

As mentioned earlier, when computational power is limited, step-detection can be performed without prior filtering, at the expense of accuracy. The maximum detection (2.28) is still viable, but requires a dead-time after each detected step to prevent noisy spikes within the magnitude from causing false positives. While more sophisticated algorithms can cope with multiple smaller peaks and are also able to search for periodic occurrences within the signal, well-suited for step-detection [SBW12], they come at the expense of required computational power.

All of the mentioned detection approaches performed a binary yes/no classification for steps, based on some metric or heuristic. For values that are far beyond the chosen threshold, the classification is clear. If the value is only slightly above or below the threshold, the correctness of the algorithms classification is uncertain. This can be addressed by using a probabilistic classification instead, where each potentially detected step is assigned a probability for it to represent a real step. The probability e.g. depends on how pronounced the step appears within the accelerometer readings, where the magnitude can be used to infer a probability $p_{\text{step}}\left(A_{[n]}\right)$ for $A_{[n]}$ to denote a step, by applying a probability distribution. As more pronounced peaks usually are more likely to represent a step, the corresponding probability can e.g. be inferred by sending the reciprocal of $A_{[n]}$ into an exponential distribution with some value for λ

$$p_{\text{step}}\left(A_{[n]}\right) = \lambda \exp\left(-\lambda \left(A_{[n]}\right)^{-1}\right), \quad \lambda > 0, \quad A_{[n]} \geq 0. \tag{2.29}$$

When results are preferred within the range of $[0, 1]$, (2.29) can be unnormalized

$$p_{\text{step'}}\left(A_{[n]}\right) = \exp\left(-\lambda \left(A_{[n]}\right)^{-1}\right). \tag{2.30}$$

The result of applying this unnormalized version to the data from figure 2.8, is shown in figure 2.9. The probability of all obvious steps is approximately the same. Peaks that were below the threshold within figure 2.8 are now also detected, but rated with a much lower probability.
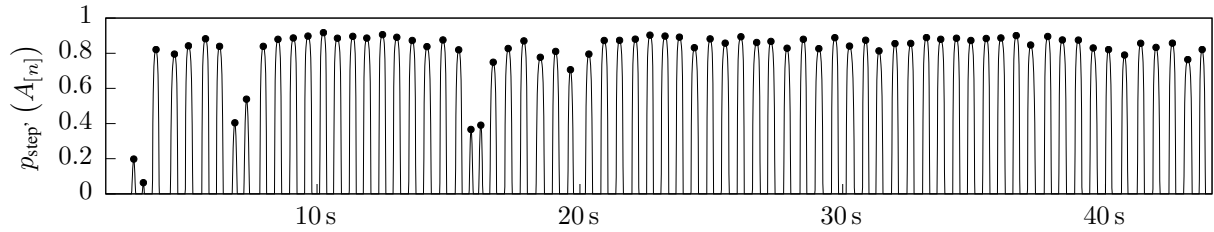
Figure 2.9: Result after applying (2.30) with $\lambda = 0.15$ to the data from figure 2.8, omitting all values ≤ 0. While all obvious steps share approximately the same probability, uncertain steps are still detected, but rated with a much lower probability.

Köping et al. mention that, while more pronounced peaks usually denote an increased probability, peaks above a certain threshold might not be related to steps, but to other events, like the smartphone being shaken by the pedestrian. Therefore they propose using a Gamma distribution instead, to model an area of interest up to a certain threshold [KGD14].

Independent of whether a binary or probabilistic classification is used, step-detection only estimates the number of steps taken. Unlike a car's speedometer, there is no hint on the current walking speed, which also depends on the pedestrian's step size. This size not only varies from person to person, but also over time, and is dependent on current ambient conditions. While climbing stairs, the pedestrian usually takes one tread at a time, and the step size matches the width of each tread. For stairs to be comfortable, the tread size should follow some rules, based on average step sizes along ground and the stair's inclination, first mentioned by Blondel in the 17th century [Blo83, pp. 672]. As of today, most stairs use a tread size somewhere around $29\,\mathrm{cm}$ [RRF02; SJP13]. The step size on ground, however, is more variable, and depends on age and sex of the pedestrian. Saunier et al. mention that reliable sources for pedestrian step sizes are hard to find, and thus conducted own experiments with subjects from several countries. Depending on the dataset, they conclude that the average step size is around $0.68\,\mathrm{m}$ with a high deviation of ± 0.22 [Sau+11]. Other approaches therefore try to dynamically estimate the step length, which is only practical when the smartphone is held in a specific pose [Yu+19].

Combining the mentioned aspects, an evaluation (2.5) working for both, discrete yes/no $o_t^{(\text{step})} \in \{0, 1\}$, and continuous probabilistic $o_t^{(\text{step})} \in [0, 1]$ step observations, is given by

$$p_{\text{step}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) = \overbrace{o_t^{(\text{step})}\mathcal{N}\left(d \mid \mu_{\text{step}}, \sigma_{\text{step}}^2\right)}^{\text{step made}} + \overbrace{\left(1 - o_t^{(\text{step})}\right)\mathcal{N}\left(d \mid 0, \sigma_{\text{stand}}^2\right)}^{\text{no step made}} \qquad (2.31)$$

$$d = \text{dist}_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right), \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, z, \dots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\text{step}, \dots)\rangle_t,$$

using the two dimensional Euclidean distance

$$\text{dist}_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right) = \|\mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_{t-1}\right) - \mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_t\right)\| \qquad (2.32)$$

between two states, based on the two dimensional location contained within each one

$$\mathbf{pos}_{xy}\left(\boldsymbol{q}_t\right) = \left(q_t^{(x)}, q_t^{(y)}\right)^T .$$ (2.33)

(2.31) converts every potential state change from $\boldsymbol{q}_{t-1}$ to $\boldsymbol{q}_t$ into a distance, which is compared using two normal distributions, depending on whether a step has been detected, $o_t^{(\text{step})} = 1$, or not $o_t^{(\text{step})} = 0$. When detected, this distance should be near the pedestrian's step size μ_{step}, with some uncertainty σ_{step}. Otherwise, there should be no movement, thus comparing against a zero mean normal distribution. The same holds true if $o_t^{(\text{step})}$ is not a discrete yes/no decision, but a continuous probability for the current detection to denote a step. Here, the two distributions are combined based on the value of $o_t^{(\text{step})} \in [0, 1]$, resulting in a mixture distribution.

Depending on the pedestrian's step frequency [Sau+11] and the timeframe between $\boldsymbol{q}_t$ and $\boldsymbol{q}_{t-1}$, more than a single step might have occurred. For handling such cases, the discrete $o_t^{(\text{step})}$ could e.g. be replaced by a number of detected steps, resulting in

$$p_{\text{steps}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) = \begin{cases} \mathcal{N}\left(d \mid n\mu_{\text{step}}, n\sigma_{\text{step}}^2\right) & n > 0 \\ \mathcal{N}\left(d \mid 0, \sigma_{\text{stand}}^2\right) & n = 0 \end{cases}$$ (2.34)

$$n = o_t^{(\text{steps})}, \quad o_t^{(\text{steps})} \in \mathbb{N}_0, \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, z, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\text{steps}, \ldots)\rangle_t .$$

All presented evaluations used the readings from a phone's accelerometer, to infer the likelihood of potential movements based on detected steps. Combined with either an estimated or empiric choice for the step size, it constrains the distance, walkable within a certain timeframe.

2.4.2 Turn-Detection

Besides the distance moved within a certain timeframe, inferred by e.g. a car's speedometer, or step detection for pedestrians, this movement's direction is also required, to perform relative location updates. In case of outdoor navigation within cars, this heading is only of minor importance, as potential movements are spatially limited by the road network. Assuming the driver to stay on the last known road, the direction of the movement is defined by this road's course. In case of U-turns and intersections, however, heading changes are mandatory. A sensor directly installed within the car's steering wheel is able to provide very high resolutions at almost no error [XYH09], and yields the current steering angle, chosen by the driver. Combined with the car's velocity, a turn-rate in $°/\text{s}$ can be determined. The sensor thus provides *relative* heading changes, requiring the previous heading to be known. For car navigation, this value can often be estimated, e.g. by using the course of the currently expected road as absolute heading.

Within open environments, as in seafaring, aviation or large buildings, estimating the heading based on the current location often is unfeasible, as potential directions are rarely constrained by the environment. Furthermore, in contrast to the sensor of the car's streering wheel, sensors within a ship's or airplane's rudder are less reliable, as the actual direction is also affected by ocean current, crosswinds or similar. That is, even though the sensor's indication itself is precise and accurate, the actual direction can deviate due to environmental influences. Concerning pedestrians, the problem is even more pronounced, requiring for different approaches.

While a compass is one of the first sensors that comes to mind when thinking of heading, there are several drawbacks. The earth's magnetic field is relatively weak, compared to the influence of metal objects or magnets near the sensor. Provided readings are thus easily distorted, and often unreliable, especially indoors [Goz+11]. Furthermore, the compass relies on a *declination* angle, to relate *magnetic* and *geographic* north. As the earth's magnetic field is subject to changes, this value is not constant, and also depends on the current location on earth, requiring the absolute location on earth to be approximately known [MMM96]. Aforementioned issues lead to the compass not being the first sensor-choice for heading estimation indoors.

Typically, cars, airplanes, ships and pedestrians move within a plane that is parallel to the ground, which, regarding the coordinate system from figure 2.5, is the (x, y)-plane. Changes in heading are thus given by a rotation around the perpendicular z-axis. As building floors, and thus the (x, y)-plane, are parallel to the earth's surface, gravity manifests along the z-axis of this coordinate system. Rotating around this axis will therefore not affect the measurable gravity, and accelerometers are unable to infer the current heading when moving along that plane.

To address the problem, ships use a so-called gyrocompass. This instrument contains a fast-spinning flywheel, attached to a gimbal ring, aligning itself parallel to the earth's rotation axis, and thus pointing towards geographic north/south, independent of a magnetic field. The flywheel keeps this alignment, similar to the wheels of a fast moving bicycle. When turning the gyrocompass, the gimbal ring allows the wheel to stay as-is. The device thus provides the ship's current absolute heading with respect to the earth's geographic north/south [Lu+94; Wes50].

Airplanes often rely on a more general version of the gyrocompass, called gyroscope, allowing for more degrees of freedom, needed for the plane's attitude, heading and turn. The gyroscope also features a fast spinning flywheel, but is allowed to rotate freely around all three axes. Due to spinning, the flywheel keeps its orientation, even when its exterior (the plane itself) is moving. Compared to the gyrocompass, there is no absolute alignment for the flywheel. It just keeps the pose it had during the time of spin-up. Especially within smaller aircrafts, the pilot thus has to manually adjust the instrument to display the correct value for the current point in time, e.g. by using an additional compass. Hereafter, the instrument shows all movements

relative to this initial position [ND97]. Due to the relative aspect and mechanically moving parts, the instrument is expected to suffer from drifts, increasing over time [FM63].

This type of sensor, yet much smaller and less mechanical, is installed within most smartphones' IMU. Opposed to instruments found in ships and airplanes, this sensor does not provide an absolute angle $\boldsymbol{\theta}'(t)$ with respect to the earth's north/south (ship's gyrocompass), or the initial spin-up (airplane's gyroscope), but denotes how fast it is *currently* being turned by influences from the outside. This *angular velocity* $\boldsymbol{\omega}'(t)$ is given in $\mathrm{rad/s}$ (or $°/\mathrm{s}$) for any given instant in time t. An absolute value is thus given by cumulating all changes since $t = 0$, that is, the integral over all consecutive readings

$$\boldsymbol{\omega}'(t) = \frac{\mathrm{d}\boldsymbol{\theta}'}{\mathrm{d}t}, \quad \boldsymbol{\theta}'(t) = \int_0^t \boldsymbol{\omega}'(t)\,\mathrm{d}t, \quad \boldsymbol{\omega}' = (x, y, z)^T. \tag{2.35}$$

When dealing with discrete sensor readings $\boldsymbol{\omega}'_{[]}$, provided by a smartphone's operating system, an approximation of (2.35) must be used instead. As mentioned earlier, readings will not necessarily be provided at equidistant points in time. Therefore, the (varying) time Δt since the last reading has to be included as well, to correctly convert $\mathrm{rad/s}$ to rad

$$\boldsymbol{\theta}'(t) \approx \sum_{n=1}^N \Delta t \boldsymbol{\omega}'_{[n-1]}, \quad \Delta t = f_{\text{time}}(n) - f_{\text{time}}(n-1), \quad f_{\text{time}}(N) \leq t. \tag{2.36}$$

(2.36) approximates the continuous integral, with the error becoming infinitesimally small for $\Delta t \to 0$. However, this value depends on the sensor's sample rate and e.g. callbacks provided by a smartphone's operating system. Besides other workarounds, the trapezoid rule (2.37) is an often used improvement when dealing with discrete integrals [KU94]:

$$\boldsymbol{\theta}'(t) \approx \sum_{n=1}^N \Delta t \boldsymbol{\omega}'_{[n-1]} + \Delta t \frac{\boldsymbol{\omega}'_{[n]} - \boldsymbol{\omega}'_{[n-1]}}{2} = \sum_{n=1}^N \Delta t \frac{\boldsymbol{\omega}'_{[n-1]} + \boldsymbol{\omega}'_{[n]}}{2}. \tag{2.37}$$

(2.36) and (2.37) provide the three angles $\theta'^{(x)}$, $\theta'^{(y)}$ and $\theta'^{(z)}$ in rad, the gyroscope has been rotated by since $t = 0$. When adding the initial, absolute alignment, this derives the smartphone's current pose. To prevent errors, Δt should not only be as small as possible, according to above equations, it should be as exact as possible. Every error in measuring this time difference will directly affect the result of (2.36) and (2.37). If it is off by $1\,\%$, so is the cumulated output.

As long as the IMU's z-axis is aligned parallel to the world's z-axis, that is, the smartphone being placed parallel to the ground, $\omega'^{(z)}$ read from the gyroscope directly denotes the pedestrian's change in heading per second. If the phone is not parallel to the ground, this turn-rate is split among all three axes, $\omega'^{(x)}$, $\omega'^{(y)}$ and $\omega'^{(z)}$, depicted in figure 2.10. This is similar to the
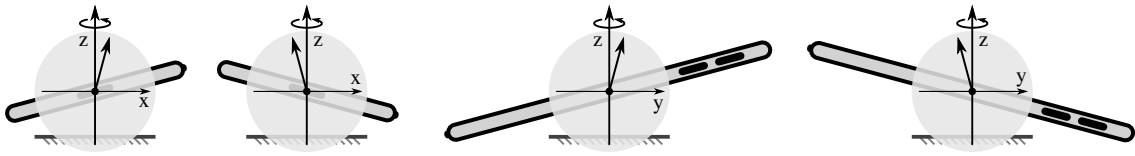
Figure 2.10: Impact of the smartphone's pose with respect to the ground on its gyroscope's readings when being rotated counter clockwise around z. The length of the vector denotes the rotation speed.

issues previously described in section 2.4.1, examining rotation invariance of the step-detection, where the issue was addressed by using the accelerometer's magnitude to overcome the impact of rotation. In theory, the same approach is feasible for turn-detection, but only if several constraints are met. As the magnitude removes the sign, and thus the direction of the turn, it needs to be reconstructed. When the smartphone is held by the pedestrian in front of the body, most rotation changes will be similar to the ones shown in figure 2.10. For typical holding patterns with the phone slightly tilted (see figure 2.5), the z-axis will often contain a significant amount of the pedestrian's turn-rate ω, and can thus be used as basis for the sign reconstruction

$$\omega_{[n]} = \mathrm{sgn}\left(\omega'^{(z)}_{[n]}\right) \left\|\boldsymbol{\omega}'_{[n]}\right\| . \tag{2.38}$$

If the pedestrian continuously holds the smartphone in the same pose, (2.38) will provide viable results. Shaking the phone left/right or forward/backward, however, also contributes to the vector's magnitude. Those movements can not be detected and removed from ω, artificially increasing this value, even if the pedestrian keeps the current heading.

Tilt Compensation   For a general solution to the problem, the gyroscope's readings $\boldsymbol{\omega}'$ must be projected onto the ground-plane (parallel to the earth's surface) prior to integration (2.37). That is, converting them into what they would look like if the phone was placed parallel to the ground. To perform this alignment, its current pose with respect to this plane must be determined. As mentioned earlier, the accelerometer is able to provide this information. Examining the constant gravity present within $\boldsymbol{a}$, the phone's rotation around the x and y-axis can be identified. Figure 2.11 describes expected accelerometer readings, depending on the phone's pose. Comparing those values against previous findings within figure 2.10, both sensor values – accelerometer and gyroscope – are influenced in the same way. Therefore, any function that reverses the effect of the pedestrian tilting the phone, converting the accelerometer's readings back to $\boldsymbol{a} \approx (0, 0, 9.81)^T$, will also convert the gyroscope's readings $\boldsymbol{\omega}'$ to a form where solely $\omega'^{(z)}$ denotes rotations around the global z-axis, and thus the pedestrian's turn-rate ω.

One solution is given by estimating a rotation matrix, that reverses the rotation imposed by the pedestrian holding the phone, hereafter multiplying all sensor readings $\boldsymbol{\omega}'$ with this matrix,
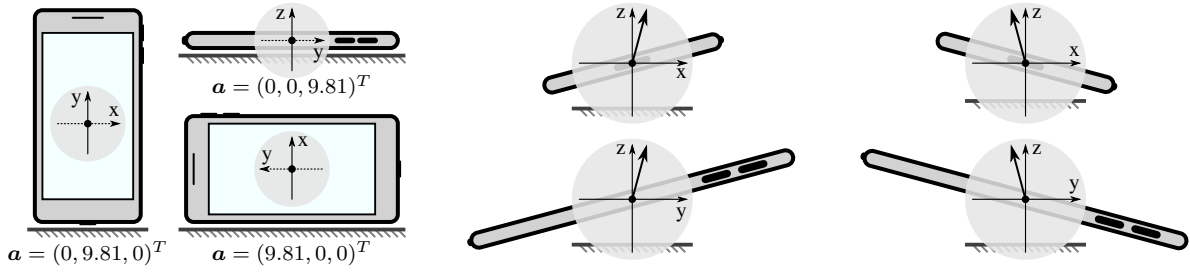
Figure 2.11: Impact of the smartphone's pose with respect to the ground on its accelerometer's readings.

prior to the integration step (2.37). As shown within figure 2.11, the accelerometer's readings $\boldsymbol{a}$ denote the z-axis of the phone's coordinate system with respect to the world's coordinate system. The two remaining axes are created using the cross product and an auxiliary vector, which must not be parallel to z. As rotations around the z-axis do not need to be considered to project the gyroscope's readings, $(1, 0, 0)^T$ or $(0, 1, 0)^T$ are viable choices. Depending on the chosen temporal vector, its cross product with $\boldsymbol{a}$ either denotes the y or x-axis. The order within the cross product is important, to ensure that a *right-handed coordinate system* is created. The third axis is then given by the cross product of z and the one just created. After normalizing each to a length of 1, they denote an *orthonormal* coordinate system and a rotation matrix $\boldsymbol{R}$, that converts $(0, 0, 9.81)^T$ into $\boldsymbol{a}$. The inverse $\boldsymbol{R}^{-1}$ of $\boldsymbol{R}$, given by $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$ due to orthogonality, then represents the required rotation matrix that reverses the effect of tilting the phone

$$
\begin{aligned}
\boldsymbol{u}_z &= \boldsymbol{a} \\
\boldsymbol{u}_x &= (0, 1, 0)^T \times \boldsymbol{u}_z \\
\boldsymbol{u}_y &= \boldsymbol{u}_z \times \boldsymbol{u}_x
\end{aligned}
\qquad
\begin{aligned}
\boldsymbol{R} &= \left(\frac{\boldsymbol{u}_x}{\|\boldsymbol{u}_x\|} \quad \frac{\boldsymbol{u}_y}{\|\boldsymbol{u}_y\|} \quad \frac{\boldsymbol{u}_z}{\|\boldsymbol{u}_z\|} \right) \\
\boldsymbol{R}^{-1} &= \boldsymbol{R}^T .
\end{aligned}
\tag{2.39}
$$

The projected turn-rate ω is then given by

$$
\omega = \left(\boldsymbol{R}^{-1} \boldsymbol{\omega}' \right)^{(z)} .
\tag{2.40}
$$

However, when projecting the gyroscope's readings to undo the rotation, changing the phone's orientation, e.g. from portrait to landscape, manifests as a large heading change. This problem can be addressed by examining the behavior of the accelerometer's readings $\boldsymbol{a}$. The vector's direction changes only slightly while walking and turning, but rapidly, when the phone's orientation is changed. Whenever detected, e.g. by using a *Principal Component Analysis* (PCA), readings from the gyroscope should be ignored or assumed to be uncertain [Ebn+15].

Similarly, previous discussions of the accelerometer's readings indicated the presence of noise (cf. figure 2.6), demanding for low-pass filtering to derive a stable smartphone pose estimation (2.39). Overmuch filtering, however, will suppress minor pose changes, yielding poten-

tially incorrect projections, and thus divergent turn-rates (2.40). A similar effect occurs when the pedestrian turns rapidly, creating a measurable change in acceleration, and thus invalid pose estimations, affecting the turn-rate.

Besides turns, the three axes of the gyroscope also provide information on pose changes, and, due to the integration step, suffer from a reduced amount of high frequency zero mean noise, at the drawback of increasing drifts [Smi99]. Examining the advantages and disadvantages of both sensors, it becomes clear that the accelerometer, due to its noise level, is well suited for slow changes and absolute indications. In contrast, the gyroscope very well captures rapid changes, but suffers from cumulating drifts. By combining both sensors, the best of each can be used. For the described use case, this *sensor fusion* is e.g. provided by a *complementary filter* [GY15], depicted in figure 2.12, meant for combining the data $x'_{[]}$ and $x''_{[]}$ from two sensors using a low-pass $f_{\text{Lo}}\,()$, and complementary high-pass filter $f_{\text{Hi}}\,()$

$$y_{[]} = f_{\text{Lo}}\left(x'_{[]}\right) + f_{\text{Hi}}\left(x''_{[]}\right) , \tag{2.41}$$

with both filters together satisfying

$$f_{\text{Lo}}\left(x_{[]}\right) + f_{\text{Hi}}\left(x_{[]}\right) \overset{!}{=} x_{[]} . \tag{2.42}$$

Designing a complementary low-pass and high-pass is simple for FIR based filters, described in section 2.4.1, as a direct complement can be calculated for every filter kernel [Smi99]. For second- and higher order IIR filters, designing complements is possible, but results will not always satisfy (2.42). This requires special complementary versions, like *Linkwitz-Riley* filters [Har+13]. For both setups, the two sensors are required to provide readings at the same instant in time, sharing a well known sample rate. As discussed, this can not be guaranteed when referring to smartphone sensors. However, as exact cut-off frequencies are not necessarily required for gyroscope/accelerometer fusion, real-world setups often refer to a simpler form of (2.41), inspired by IIR filters, satisfying (2.42), and mitigating aforementioned requirements [Isl+16]

$$y_{[n]} = \underbrace{\kappa\Big(y_{[n-1]} + \overbrace{\left(x'_{[n]} - x'_{[n-1]}\right)}^{\Delta x'_{[n]}}\Big)}_{\text{high-pass}} + \underbrace{(1-\kappa)x''_{[n]}}_{\text{low-pass}}, \quad 0.0 < \kappa < 1.0 . \tag{2.43}$$

κ within (2.43) adjusts the strength of the high-pass, and $(1-\kappa)$ the complementary low-pass. As $x'_{[]}$ is high-pass filtered, it can be provided as relative input $\Delta x'_{[]}$, just like turn-rates from the gyroscope. The low-pass on $x''_{[]}$ filters accelerometer readings. By combining both, the accelerometer provides a stable pose, delayed by the low-pass, which is compensated by
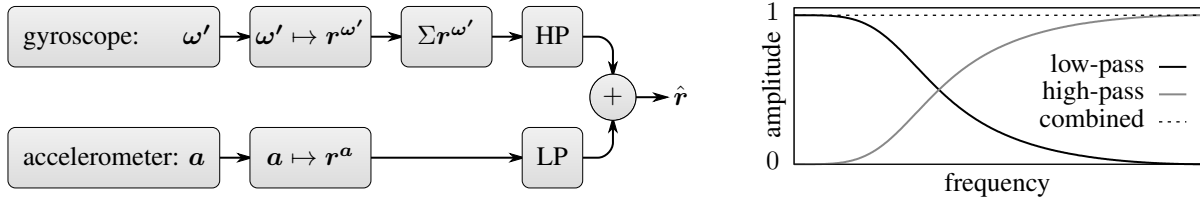
Figure 2.12: Layout and behavior of a complementary filter which combines the values provided by an accelerometer and a gyroscope. After converting their readings to absolute rotations, the converted results are applied to a low-pass and a high-pass filter, which are complementary to each other.

the relative adjustments from the gyroscope, passed through the high-pass. Using this setup provides a low-noise, yet low-delay, pose estimation for the phone without drift [YN01].

However, as both sensors supply different kinds of data, angular velocity from the gyroscope, and absolute acceleration from the accelerometer, readings must be converted into a shared unit, before (2.43) can be used.

One option is to convert both into Euler angles, describing a rigid body's orientation by three *consecutive* rotations around orthogonal axes [Die06]. According to figure 2.11 and [Jan+15], these three Euler angles α, β, γ, consecutively rotating around x, y, and z, can be derived from the direction of the constant gravity, present within the accelerometer's readings $\boldsymbol{a}$

$$\alpha = \operatorname{atan2}\left(a^{(y)},\, a^{(z)}\right)\,, \quad \beta = \operatorname{atan2}\left(-a^{(x)},\, \sqrt{\left(a^{(y)}\right)^2 + \left(a^{(z)}\right)^2}\right)\,, \quad \gamma = 0\,, \quad (2.44)$$

where $\operatorname{atan2}(y,\, x)$ is a modified version of $\tan^{-1}(y/x)$, respecting the four possible quadrants. As mentioned earlier, the accelerometer does not provide any valuable information on the rotation around the z-axis with respect to the world coordinate system, and thus $\gamma = 0$ in (2.44).

At first glance, the integration (2.37) seems sufficient for deriving a similar representation from the gyroscope readings $\boldsymbol{\omega}'$. However, they describe changes based on the sensor's point of view, often referred to as *body frame* [NPM13]. With respect to the outside view, or *inertial frame*, the orientation changes after every new sensor reading. This would require a cumulative adjustment prior to the integration, converting the angular velocity $\boldsymbol{\omega}'$ as seen from the sensor, to a change in Euler angles, or *Euler angle rates*, as seen from the outside, described in [Jan+15]. However, due to required trigonometric operations, this approach is limited to $[-\pi, +\pi]$, causing discontinuous behavior near $\pm 90°$. While not being a problem in general, this imposes issues when applying digital filters, e.g. fading from $-\pi$ to $+\pi$ or vice versa. Furthermore, required calculations suffer from constraints, known as Gimbal lock problem [Jan+15].

All issues can be addressed by using *quaternions Q* instead of Euler angles [Vin17]. In place of three consecutive rotations, orientation is described using a complex representation of

a *rotation axis* $\boldsymbol{u}$ and a corresponding *angle* α to rotate around it

$$Q = (\alpha, \boldsymbol{u}) = \left(\cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right) \left(u^{(x)}\mathbf{i} + u^{(y)}\mathbf{j} + u^{(z)}\mathbf{k} \right) \right), \quad Q \in \mathbb{H}$$
$$\text{with } \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \quad |\boldsymbol{u}| \overset{!}{=} 1. \tag{2.45}$$

Similar to (2.39), in order to convert the orientation measured by the accelerometer into a quaternion, the rotation axis must be chosen so that rotating around it converts $\boldsymbol{a}$ to $\approx (0, 0, 9.81)^T$, or $(0, 0, 1)^T$ when normalized. It is thus given by the normalized cross product between those two vectors. The amount of rotation is denoted by the angle between both, that is, their dot product

$$Q_{[n]}^{\boldsymbol{a}} = \left(\alpha, \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \right), \quad \alpha = \cos^{-1}\left(\frac{\boldsymbol{a}_{[n]}}{\|\boldsymbol{a}_{[n]}\|} \bullet (0, 0, 1)^T \right),$$
$$\boldsymbol{u} = \boldsymbol{a}_{[n]} \times (0, 0, 1)^T = \left(a_{[n]}^{(y)}, -a_{[n]}^{(x)}, 0 \right)^T. \tag{2.46}$$

The angular velocity $\boldsymbol{\omega}'$, provided by the gyroscope, can directly be converted into a quaternion $Q^{\boldsymbol{\omega}'}$, where the normalized vector describes the rotation axis, and the rotation angle is defined by the vector's magnitude, multiplied with the elapsed time

$$Q_{[n]}^{\boldsymbol{\omega}'} = \left(\Delta t \left\| \boldsymbol{\omega}'_{[n]} \right\|, \frac{\boldsymbol{\omega}'_{[n]}}{\left\| \boldsymbol{\omega}'_{[n]} \right\|} \right). \tag{2.47}$$

Subsequent $Q_{[n]}^{\boldsymbol{\omega}'}$ can then be cumulated by quaternion multiplication. In contrast to using Euler angles, a prior adjustment of the current readings is not required, as the quaternion multiplication directly adjusts the rotation axis

$$Q_{[N]}^{\boldsymbol{\theta}'} = \prod_{n=1}^{N} Q_{[n]}^{\boldsymbol{\omega}'} \quad \Rightarrow \quad Q_{[n]}^{\boldsymbol{\theta}'} = Q_{[n]}^{\boldsymbol{\omega}'} Q_{[n-1]}^{\boldsymbol{\theta}'}. \tag{2.48}$$

This aspect simplifies using gyroscope readings within the complementary filter

$$\hat{Q}_{[n]} = \kappa \overbrace{\left(Q_{[n]}^{\boldsymbol{\omega}'} \hat{Q}_{[n-1]} \right)}^{\text{integral like (2.48)}} + (1 - \kappa) Q_{[n]}^{\boldsymbol{a}}, \quad 0 < \kappa < 1. \tag{2.49}$$

(2.49) multiplies the current turn-rates $Q_{[n]}^{\boldsymbol{\omega}'}$ from the gyroscope with the previous filter output $\hat{Q}_{[n-1]}$. This represents an integration similar to (2.48), and denotes the high-pass part. Its result is interpolated with the quaternion $Q_{[n]}^{\boldsymbol{a}}$ from the accelerometer, which contributes slowly by $(1 - \kappa)$, representing the low-pass. For increased accuracy, a spherical interpolation [Vin17] should be used. The required rotation matrix $\boldsymbol{R}^{-1}$ can hereafter be derived by converting $\hat{Q}$ into a 3×3 rotation matrix, as shown in [Die06]. An example is provided in appendix A.1.

(2.49) provides a stable pose estimation, suppressing drift and noise. The resulting rotation matrix $\boldsymbol{R}^{-1}$ is used within (2.40), undoing the effect of the pedestrian tilting the phone, deriving the projected turn-rate ω. Hereafter, mentioned discrete integration techniques are applied to cumulate the turn-rate, yielding the change in heading within a certain timeframe

$$\theta_t = \sum_n \Delta t \omega_{[n]}, \quad (t-1) \leq f_{\text{time}}(n) < t, \tag{2.50}$$

or the absolute heading with respect to the pedestrian's initial orientation Θ_0

$$\Theta_t = \Theta_0 + \sum_n \Delta t \omega_{[n]}, \quad 0 \leq f_{\text{time}}(n) < t. \tag{2.51}$$

Probabilistic Assembly Both values can be used for a probabilistic evaluation of potential pedestrian movements, constrained on the absolute heading Θ, or the change in heading θ within a certain timeframe. While the latter can be expected to be stable for shorter timeframes, the absolute Θ will suffer from cumulating errors, independent of the chosen pose estimation and gyroscope projection, as it can not be compensated by the sensor fusion with the accelerometer. Depending on the building's architecture, one way of dealing with drifts and cumulating uncertainties is to discretize turns, e.g. to multiples of $90°$. Using discrete options, no-turn, $90°$ left and $90°$ right, mitigates the impact of cumulating sensor drift and spreading uncertainties, yet, at the cost of accuracy, especially when walking within large, open areas [Köp+14; Ebn+14].

Drifts and other errors are therefore addressed by probabilistic models, including uncertainties when comparing the observed absolute heading against the pedestrian's walking direction in the (x, y) plane, given by the angle between two states, with respect to the x-axis

$$\angle_{\text{xy}}(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t) = \text{atan2}\left(q_t^{(y)} - q_{t-1}^{(y)}, \; q_t^{(x)} - q_{t-1}^{(x)}\right). \tag{2.52}$$

The smallest signed difference between two angles in rad is then given by

$$\angle_{\Delta}(\alpha, \beta) = \text{atan2}\left(\sin(\beta - \alpha), \cos(\beta - \alpha)\right). \tag{2.53}$$

This difference, between the angle $\boldsymbol{q}_{t-1} \to \boldsymbol{q}_t$ and the absolute observed heading, should be close to 0, including an uncertainty, e.g. given by a normal distribution

$$p_{\text{absTurn}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) = \mathcal{N}\left(\angle_{\Delta}\left(\alpha, o_t^{(\Theta)}\right) \Big| 0, \sigma_{\text{turn}}^2\right)$$
$$\alpha = \angle_{\text{xy}}(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t), \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\Theta, \ldots)\rangle_t. \tag{2.54}$$

Due to $o_t^{(\Theta)}$ suffering from cumulating errors, the corresponding uncertainty σ_{turn} must increase over time. Furthermore, this approach requires the pedestrian's initial heading offset Θ_0 to be known. Both drawbacks can be mitigated by using relative comparisons instead.

To compare potential pedestrian movements with an observed *change* in heading, three consecutive states $\boldsymbol{q}_{t-2} \to \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t$ must be considered. The heading change is then defined by the difference between the two *absolute* walking angles $\angle_{\text{xy}}\left(\boldsymbol{q}_{t-2}, \boldsymbol{q}_{t-1}\right)$, and $\angle_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)$. This change is then compared against the observed one, also including an uncertainty:

$$
\begin{aligned}
p_{\text{relTurn'}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}, \boldsymbol{q}_{t-2}\right) &= \mathcal{N}\left(\angle_\Delta\left(\alpha, o_t^{(\theta)}\right) \Big| 0, \sigma_{\text{turn}}^2\right) \\
\alpha &= \angle_\Delta\left(\angle_{\text{xy}}\left(\boldsymbol{q}_{t-2}, \boldsymbol{q}_{t-1}\right), \angle_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)\right) \\
\langle\boldsymbol{q}\rangle_t &= \langle(x, y, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\theta, \ldots)\rangle_t.
\end{aligned}
\tag{2.55}
$$

To omit the need for including $\boldsymbol{q}_{t-2}$ within the equation, the previous heading $\angle_{\text{xy}}\left(\boldsymbol{q}_{t-2}, \boldsymbol{q}_{t-1}\right)$ can be remembered by adding it as a variable to the state, slightly altering above equation

$$
\begin{aligned}
p_{\text{relTurn}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) &= \mathcal{N}\left(\angle_\Delta\left(\alpha, o_t^{(\theta)}\right) \Big| 0, \sigma_{\text{turn}}^2\right) \\
\alpha = \angle_\Delta\left(q_{t-1}^{(\Theta)}, \angle_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)\right)&, \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, \Theta, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\theta, \ldots)\rangle_t.
\end{aligned}
\tag{2.56}
$$

While the normal distributions used in (2.54), (2.55) and (2.56) are a common choice for modeling uncertainties, they are not ideal when referring to angular units, as $\int_{-\pi}^{+\pi} \mathcal{N}\left(x \mid 0, \sigma^2\right) \, \mathrm{d}x$ will only sum up to ≈ 1.0, and only for smaller σ, yielding potential normalization issues. This can be addressed by using distributions intended for angular use cases, with the range of $\pm\pi$ in mind, such as the von Mises distribution [Mis18; For+10]. By being both, normalized for the interval $[-\pi, +\pi]$, and periodic, angular uncertainties are handled more suitably.

Independent of the chosen distribution, an uncertainty value must be determined. Keller et al. [KWS12] provide an overview on expected uncertainties, by examining sensors within both, laboratory and real-world conditions. They estimated noise levels, scale offsets and long-term drifts of gyroscope sensors, by using accurate references. Hereafter, they compared the results of an $80\,\mathrm{m}$ long walk using step and turn detection, with and without prior sensor calibration, clearly indicating calibration improvements. While hardware based drifts or scale offsets can be estimated during a calibration process, the behavior of the pedestrian can not [ST14]. As mentioned earlier, some movements, like fast turns, affect the measured acceleration, and thus the pose estimation and gyroscope projection, yielding incorrect turn-rates. It therefore makes sense to not assume a constant uncertainty in $\mathrm{rad/s}$ or $°/\mathrm{s}$, but also a dynamic amount that is increased whenever major changes within the gyroscope or accelerometer readings are detected.

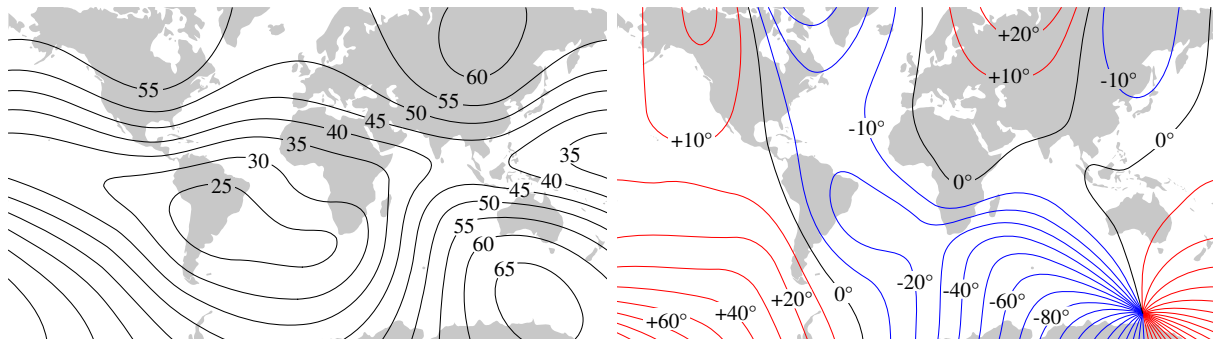Figure 2.13: Local intensity of the magnetic field (magnitude of $\boldsymbol{B}'$) in µT, and declination correction Θ_{dec}. Adapted from the World Magnetic Model 2019, developed by NOAA/NCEI/CIRES [NOA].

A potential heuristic for smaller timeframes could thus read as follows:

$$
\sigma_{\mathrm{turn}} = \Delta t \overbrace{\left(1\,°/\mathrm{s}\frac{\pi}{180°}\right)}^{\text{constant}} + \overbrace{\left(\kappa\,|o_t^{(\theta)}|\right)}^{\text{variable}}, \quad \kappa \geq 0\,. \tag{2.57}
$$

Within (2.57), a constant turn error in $°/\mathrm{s}$ is combined with a variable component, depending on the amount of change, with its impact adjusted by κ. Both values are an empiric choice and depend on the quality of the sensors installed within a phone. For a truly dynamic uncertainty estimation, based on observations and other prior knowledge, more sophisticated approaches are required [Bra+05; CYJ15]. Especially when working with absolute headings (2.54), the uncertainty depends on the time elapsed since start, and eventually reaches a level where the sensor stops to provide usable information. Thus, the gyroscope is mainly suited for relative predictions, such as (2.56). For absolute indications, other sensors are better suited.

2.4.3 eCompass

As shown, relative turn information suffers from cumulating errors, eventually becoming unstable. This is mitigated by sensors providing absolute heading indications. Mentioned earlier, ships use a gyrocompass pointing towards the earth's geographic north/south by aligning itself onto the rotation axis. A *compass* provides similar information, except that the alignment is based on magnetism, and that it points towards the earth's *geomagnetic north*. The geomagnetic north differs from the geographic north, and is subject to changes over time, currently moving by over $55\,\mathrm{km}$ per year from Canada towards Siberia [Cla16; Wit19].

The indication from an analog compass is provided by a magnetized needle that is allowed to spin freely around one axis. If this axis is aligned parallel to the earth's z-axis, pointing upwards, the needle aligns itself towards north/south of the earth's magnetic field. Within smartphones, similar readings are provided by the *magnetometer*, present within most IMUs. Instead of a
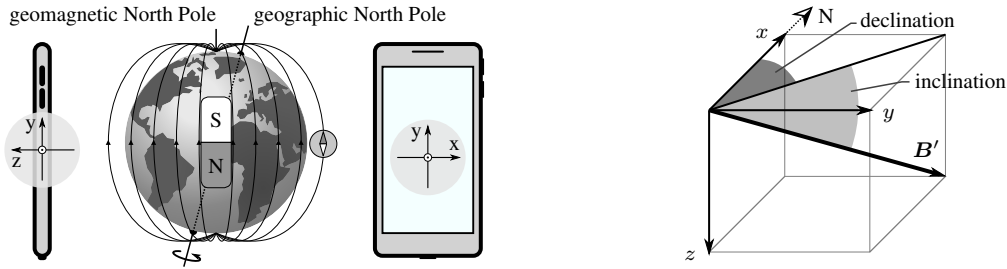
Figure 2.14: Behavior of the earth's magnetic field and its relation to the geographic poles. If a smartphone is placed parallel to the surface, just like an analog compass, the direction towards magnetic north is given by the magnetometer's readings $B'^{(x)}$ and $B'^{(y)}$. Right half adapted from [Cla16, p. 198].

magnetized needle, the sensor's internals leverage the principle of the Hall effect. When a constant current is passed through a conductive material, every magnetic force perpendicular to the movement direction of the electrons passing the conductor, deflects them in a direction that is perpendicular to both, the moving direction and the magnetic flux, forcing the electrons towards one side of the conductor (Lorentz force). Between this side and its opposite, a voltage proportional to the applied magnetic force can be measured [Pop+07]. In contrast to other probes, such as inductors, Hall sensors are also capable of measuring the influence of *static* magnetism, like non-moving magnetic objects [Cul56]. Most smartphones are equipped with a 3-axis magnetometer where three Hall sensors are combined, measuring the current *magnetic flux density* $\boldsymbol{B'} = (x, y, z)$ in µT, along three orthogonal directions. Actual intensities are location dependent, with typical values shown in the left half of figure 2.13.
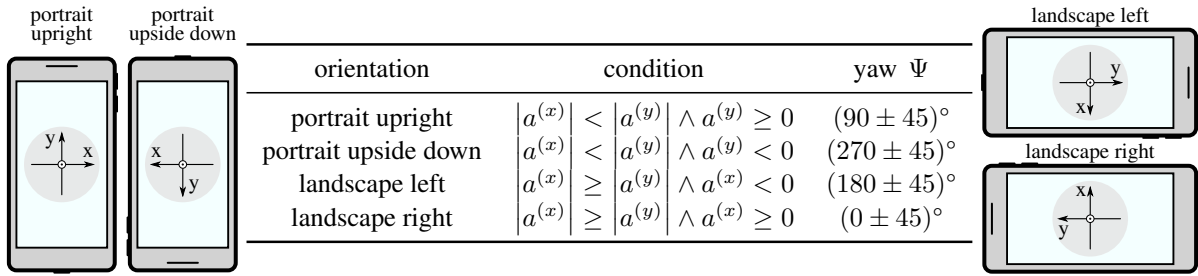
Figure 2.14 depicts the geographic and magnetic relations needed to derive a heading from the readings $\boldsymbol{B'}$ of a magnetometer. While the earth's geographic north and south poles (true north/south) are defined by its rotation axis, the geomagnetic north and south poles are defined by its internal magnetic field, which is slightly different. This difference is called declination, depends on the actual location on earth and, as mentioned, is subject to changes over time. Current values are shown in the right half of figure 2.13.

The geomagnetic north pole can be thought of the magnetic south pole of a dipole bar magnet, the compass points to. If a smartphone is placed parallel to the ground, the direction towards geomagnetic north $\Theta_{N_{mag}}$ is given by the magnetometer's readings in x and y

$$\Theta_{N_{mag}} = \text{atan2}\left(B'^{(y)}, B'^{(x)}\right) . \tag{2.58}$$

To derive the angle $\Theta_{N_{geo}}$ towards the geographic north, the current location's declination correction Θ_{dec} from figure 2.13 must be added

$$\Theta_{N_{geo}} = \Theta_{N_{mag}} + \Theta_{dec} . \tag{2.59}$$

orientation	condition		yaw Ψ
portrait upright	$\left|a^{(x)}\right| < \left|a^{(y)}\right|$	$\wedge\, a^{(y)} \geq 0$	$(90 \pm 45)°$
portrait upside down	$\left|a^{(x)}\right| < \left|a^{(y)}\right|$	$\wedge\, a^{(y)} < 0$	$(270 \pm 45)°$
landscape left	$\left|a^{(x)}\right| \geq \left|a^{(y)}\right|$	$\wedge\, a^{(x)} < 0$	$(180 \pm 45)°$
landscape right	$\left|a^{(x)}\right| \geq \left|a^{(y)}\right|$	$\wedge\, a^{(x)} \geq 0$	$(0 \pm 45)°$

Table 2.1: Smartphone orientation dependent on accelerometer readings. Adapted from [Bos14].

While Θ_{dec} is location dependent, it changes hardly within the range of a few hundred kilometers, visualized in figure 2.13. For indoor localization and navigation scenarios, it can thus e.g. be stored within a floorplan, where the building is aligned towards the geographic north.

Shown in figure 2.14, there is another angle besides $\Theta_{\mathrm{N_{geo}}}$. The inclination, or angle of *dip*, describes the upwards/downwards direction of the magnetic field, mainly dependent on the current latitude. While the exact direction of the magnetic field is defined by those two angles combined, inclination is usually omitted for north/south estimations [Cla16].

To determine $\Theta_{\mathrm{N_{mag}}}$ for arbitrary smartphone poses, a combination of magnetometer and accelerometer, also referred to as eCompass [KC15], is required. This need can be confirmed when referring to the previously discussed turn-detection, but also when observing an analog compass. To prevent its needle from touching the casing, the latter must be aligned parallel to the angle of dip, which is approximately parallel to the ground, when not residing near the poles. Just like earlier for the turn-rate, a heading estimation requires a projection $\boldsymbol{R}^{-1}$ onto the world's (x, y)-plane, using the techniques discussed in section 2.4.2, referred to as *tilt compensation*

$$\Theta_{\mathrm{N_{mag}}} = \mathrm{atan2}\left(B^{(y)},\, B^{(x)}\right), \quad \boldsymbol{B} = \boldsymbol{R}^{-1}\boldsymbol{B}'. \tag{2.60}$$

Due to sensor noise, low-pass filtering is advisable, if not already conducted by the magnetometer itself [Pop+07; BS12; KS18]. To prevent jumps near $0° \leftrightarrow 360°$ or $-180° \leftrightarrow +180°$, the input vectors $\boldsymbol{B}'$ or $\boldsymbol{B}$, should be filtered instead of $\Theta_{\mathrm{N_{mag}}}$. In (2.60), all candidates for estimating $\boldsymbol{R}^{-1}$, previously described in section 2.4.2, are applicable. However, in contrast to the relative turn-detection based on the gyroscope, the absolute eCompass requires more caution as the result from (2.60) rotates together with the phone. This refers to the four different smartphone orientations: landscape and portrait, each up and down. Changing the orientation directly affects the eCompass' heading indication. To compensate this, the pedestrian is either required to hold the phone using an exactly defined pose, as it is the case for most analog compasses, or the system must be able to distinguish between several pre-defined poses, such as portrait and landscape mode [Ngu+16]. Integrated sensor components therefore contain some heuristic to

distinguish between various orientations, based on the current accelerometer readings a and the resulting yaw angle $\Psi = \mathrm{atan2}\left(a^{(y)},\, a^{(x)}\right)$ shown in table 2.1 [Bos14]. A general solution for the pedestrian e.g. carrying the phone within trouser pockets, requires additional sensors besides the smartphone. To relate the tilt-compensated compass' north with the pedestrian's current axis of forward motion. When estimating this axis using just the smartphone's sensors, results can be unstable, and are subject to delays [Kus+15]. The heading provided by the eCompass is usually off by several degrees [Cas+14]. Using the phone's estimated orientation for projection thus is a viable choice for smartphone-only indoor localization and especially navigation, where the pedestrian faces the phone's display, similar to holding an analog compass.

Besides tilting issues, everything that affects the magnetic field will also cause a measurable effect on the sensor's readings and thus the heading estimation. While this is often not a problem within large, open-space outdoor environments, it is a problem indoors, where many metal objects, such as steel-reinforced concrete, handrails, door handles, elevators and inductive electricity, are present [Goz+11; Par+06].

However, despite negative influences on heading estimation, this allows for completely different inputs towards indoor localization. Architectural effects on the magnetic field can be very unique for different regions within a building, allowing for *magnetic matching* [Goz+11]. Here, the readings of the magnetic field are recorded once, for many locations within the building. Every recording describes the magnetic behavior at a location, serving as its fingerprint. During the localization process, the current readings from the phone's magnetometer are compared against all known fingerprints. The location of the best matching one is likely to denote the pedestrian's current whereabouts [Shu+15]. However, it requires significant effort to record the fingerprint database, which is also subject to changes over time. Setup and update issues can be mitigated, e.g. using a *self location and mapping* (SLAM) approach, where localization is initially provided by other sensors and the, yet unknown, magnetic field is recorded during the localization process. The system then stabilizes itself over time, refining both, the localization and the magnetic information, with every additional walk through the building [KS18].

Alternatively to complex fingerprinting, Ehrlich et al. [EBS16] suggest installing a few special coil-transmitters within the building, each creating a unique magnetic signature within a confined area. Those signatures can be measured by the magnetometer, and compared against a database of known transmitter signatures. This approach allows for additional location hints by using a controlled setup, as the strong coil-signature is barely affected by ambient conditions. If three or more signatures can be detected by the magnetometer, even multilateration is possible (cf. section 2.3) by inferring the distance from the strength of the magnetic field.

However, within this work, the focus is on absolute heading estimation, not requiring additional hardware installations or time-demanding setups.

Sensor Calibration Independent of the chosen approach, the magnetometer must be calibrated to compensate for *soft iron* and *hard iron* effects. Magnetic or metallic objects mounted together with the magnetometer, like the phone's speaker, introduce hard iron effects. They offset all provided readings, independent of the phone's current orientation. On the other hand, soft iron effects are induced by the outside world, through nearby objects, thus yielding influences on sensor readings, dependent on its location and orientation. Therefore, location independent hard iron effects are easier to calibrate [Kon09]. While alignment differences between the accelerometer, providing the pose projection, and the magnetometer should be considered as well, this is not an issue for today's smartphones, as the alignment is usually accurate [ZY15].

When placing the smartphone parallel to the ground, and rotating it around the z-axis (cf. figure 2.14), the measurements $(B'^{(x)}, B'^{(y)})^T$ provided by a calibrated magnetometer denote a circle, centered at $(0,0)^T$. As the actual magnitude of the sensor readings is not required to derive the heading (2.60), the calibrated circle's radius can be of any size. When performing a manual calibration, normalizing the measurements to denote a unit-circle is thus fine for the intended use case. The same holds true for the three dimensional case. Rotating the phone around all three axes, the resulting sensor readings should denote a sphere centered at $(0,0,0)^T$. A corresponding mean squared error metric for both cases is thus given by

$$\varepsilon = \frac{1}{N} \sum_{n=1}^{N} \left(1 - \left\| \boldsymbol{B'}_{[n]} \right\| \right)^2, \quad \boldsymbol{B'} = \begin{cases} (x,y)^T & \text{for 2D} \\ (x,y,z)^T & \text{for 3D} . \end{cases} \tag{2.61}$$

During the calibration process, all $\boldsymbol{B'}_{[]}$ are adjusted using a set of parameters, depending on the calibrations to address, to reduce the error (2.61). Deviations from the center $(0,0,0)^T$ can be addressed by one offset parameter per axis. Similarly, the scale for each axis can be adjusted using one scale parameter for each. Calibration hereafter represents an optimization problem

$$\arg\min_{\boldsymbol{u},\boldsymbol{w}} \sum_{n=1}^{N} \left(1 - \left\| \left(\boldsymbol{B'}_{[n]} - \boldsymbol{u} \right) \odot \boldsymbol{w} \right\| \right)^2, \tag{2.62}$$

where the offset is adjusted by subtracting $\boldsymbol{u}$, and the scale is corrected by a componentwise multiplication $\odot$ with $\boldsymbol{w}$. By calculating the 2nd partial derivative for all to-be-optimized parameters, and examining the requirements for each derivative to be > 0, it can be shown that the function is convex for typical sensor inputs, offsets starting at $(0,0,0)^T$, and scales > 0. Thus, optimization can be performed using common approaches such as gradient descent [HS06] or downhill simplex method [Pow62; NM65]. (2.62) covers most constellations, but omits rare cases where readings provided by the sensor denote a rotated ellipse/ellipsoid, requiring addi-

tional rotation parameters for correction [Fan+11]. In case of many outliers, (2.62) will not provide ideal results. A solution can be the *random sample consensus* (RANSAC) [FB81], where not all, but a (random) fraction of all inputs is used for the optimization. This step is repeated several times, depending on the amount of outliers, eventually deriving a superior result.

As shown later, for many smartphone models aforementioned manual calibration schemes are not required, as they are integrated directly into the operating system or sensor. The procedure is triggered whenever the sensor is active, and the user rotates the phone around all three axes, hereafter removing the influences of constant offsets, such as hard iron effects.

Probabilistic Assembly   While, in theory, the eCompass can hereafter be used as single (absolute) heading source, its practical use is limited by aforementioned environmental effects and the pedestrian. Together with the turn detection from section 2.4.2, however, it can compensate the cumulating heading drifts on the z-axis, the accelerometer isn't able to address [Har+03]. Besides directly fusing both readings, e.g. via a complementary filter, the sensor can be evaluated on its own, using a probabilistic approach to model expected uncertainties, allowing for later combination. Similarly to the heading evaluation for the gyroscope, the current eCompass heading $\Theta_{N_{mag}}$ is compared against the pedestrian's potential heading, either given by $\boldsymbol{q}_{t-1} \to \boldsymbol{q}_t$ or $q_t^{(\Theta)}$. The difference between geomagnetic north from the sensor, and the orientation of the building is addressed by Θ_{bldg}, describing the angle between the floorplan's x-axis and geomagnetic north, estimated once per building. Potential walks are hereafter evaluated by

$$p_{comp}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) = \mathcal{N}\left(\angle_\Delta\left(\alpha, o_t^{(\Theta_{N_{mag}})}\right) \,\Big|\, 0, \sigma_{comp}^2\right)$$

$$\alpha = \angle_{xy}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right) + \Theta_{bldg}, \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\Theta_{N_{mag}}, \ldots)\rangle_t. \tag{2.63}$$

If the heading is stored as part of the unknown state, (2.63) simplifies to

$$p_{comp}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \mathcal{N}\left(\angle_\Delta\left(\alpha, o_t^{(\Theta_{N_{mag}})}\right) \,\Big|\, 0, \sigma_{comp}^2\right)$$

$$\alpha = q_t^{(\Theta)} + \Theta_{bldg}, \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, \Theta, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\Theta_{N_{mag}}, \ldots)\rangle_t. \tag{2.64}$$

The value for σ_{comp} depends not only on the quality of accelerometer and magnetometer, but also on the pedestrian's behavior when holding the phone (steady/shaking), and nearby metallic or magnetized objects. The resulting heading precision can thus vary between errors as low as $1°$ [Kon09] and up to tens of degrees depending on location and tilt [Li+12; Hil+14; WGN15]. Changes in tilt can be detected when analyzing the accelerometer's readings within the pose estimation. The same holds true for changes of the magnetic field. Without architectural influences, the magnitude $\|\boldsymbol{B}'\|$ varies only slightly throughout the building. Whenever large changes are detected, σ_{comp} can be increased accordingly. Similarly, the uncertainty can be in-

creased whenever the eCompass reports major heading changes, but the gyroscope does not. Depending on hardware and surrounding architecture, it thus can be advisable to use a discrete comparison instead, limiting potential directions based on heuristics κ_{comp} and τ_{comp} [Ebn+17]

$$p_{\text{comp}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \eta \begin{cases} \kappa_{\text{comp}} & \angle_\Delta\left(\alpha, o_t^{(\Theta_{\text{Nmag}})}\right) < \tau_{\text{comp}} \\ (1 - \kappa_{\text{comp}}) & \text{else} \end{cases}, \quad \alpha = q_t^{(\Theta)} + \Theta_{\text{bldg}}, \quad (2.65)$$

or a compromise between (2.64) and (2.65), using the distribution from (2.21)

$$p_{\text{comp}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \mathcal{R}\left(\angle_\Delta\left(\alpha, o_t^{(\Theta_{\text{Nmag}})}\right) \,\Big|\, 0, \sigma_{\text{comp}}^2\right). \quad (2.66)$$

Yet, in case of e.g. turn rates beyond the pedestrian's normal behavior, the eCompass can be considered unstable, not providing a viable evaluation and thus $p_{\text{comp}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \text{const.}$

2.5   Barometer

The first sensor providing absolute location hints is the barometer. It measures the current atmospheric pressure in hPa, which is influenced by the amount of air that is "*piled up*" above the sensor, allowing to infer the current altitude, usually measured above *mean sea level*. If this altitude increases, the amount of air above the sensor decreases, indicating a drop in pressure. This change of atmospheric pressure can be measured e.g. by analyzing the change in size of an air-filled, flexible object, like a balloon. An increasing height above mean sea level will yield a growing balloon, as the outside pressure decreases, while the pressure inside remains constant. Most common analog barometers used another approach, based on a vertical column filled with mercury, placed within a reservoir. Depending on the surrounding pressure, mercury is pressed from the reservoir into the column, or moves from the column into the reservoir. This yields a vacuum of varying size within the column, which adjusts until the forces inside and outside are equal. The height of mercury remaining within the column is proportional to the current atmospheric pressure. Therefore, besides hPa, *inch of mercury* (inHg), *millimeter of mercury* (mmHg) or *Torr* also are common measuring units for atmospheric pressure [Tor44; Tim82].

As of today, some smartphone models are equipped with a barometer sensor, based on piezo-resistive pressure sensing [Bos15; Bos18]. Initially, it was intended to speed-up the GPS' initialization times by providing a coarse altitude information. However, experiments indicated that its success was questionable, as GPS lock times did not improve with a barometer present [KWS12]. As the sensor remained present within many newer models, its contribution towards indoor localization and navigation was examined [Mur+14], e.g. by relating atmospheric pres-

T_0	288.15	°K	temperature at sea level	ϱ_0	1013.25	hPa	pressure at sea level
R	8.314 46	Nm/molK	(ideal) gas constant	L	0.0065	°K/m	temperature lapse rate
g	9.806 65	m/s²	gravity of earth	M	0.028 96	kg/mol	molar mass of *dry* air

Table 2.2: Constants required to relate atmospheric pressure and altitude [TT08; Stu15].

sure and altitude above the *mean sea level*

$$f_\varrho\left(h\right) = \varrho_0 \left(\frac{T_0}{T_0 + Lh}\right)^{\frac{gM}{RL}} \qquad f_h\left(\varrho\right) = \frac{T_0}{L}\left(\left(\frac{\varrho}{\varrho_0}\right)^{\frac{-RL}{gM}} - 1\right). \qquad (2.67,\ 2.68)$$

(2.67) and (2.68) require several physical and chemical constants listed in table 2.2 [TT08; Stu15]. To be consistent with their typical notation, some previously introduced variables are re-purposed for the barometer. For both relations, the pressure at the mean sea level is assumed to remain constant at $\varrho_0 = 1013.25\,\text{hPa}$, at a temperature of $T_0 = 15\,°\text{C}$, decreasing constantly with $L = 6.5\,°\text{C/km}$, referred to as *standard atmosphere*. The *ideal gas law* derives the air's density, expecting it to be absolutely dry (0 % rel. humidity). It thus relates atmospheric pressure to a column of height h, filled with a gas of some density. Corresponding results are valid up to $\approx 11\,\text{km}$ in altitude, before the temperature lapse rate L becomes zero [OAS76; Stu15]

For pedestrian navigation indoors and outdoors, the first few hundred meters above sea level are the most important. The $6.5\,°\text{C/km}$ change in temperature can thus be omitted. Hereafter, the relation of altitude and pressure is approximately exponential, and (2.67)/(2.68) simplify to

$$\begin{aligned} f_\varrho\left(h\right) &\approx \varrho_0\, e^{\left(\frac{-gM}{RT_0}h\right)} \\ &\approx \varrho_0\, e^{\left(\frac{-0.0342°\,\text{K/m}}{T_0}h\right)} \end{aligned} \qquad f_h\left(\varrho\right) \approx \frac{\ln\left(\frac{\varrho}{\varrho_0}\right)T_0}{-0.0342\,°\,\text{K/m}}, \qquad (2.69,\ 2.70)$$

where all constants are combined as a single value [Stu15]. Figure 2.15 shows the relation between altitude and pressure. While the approximations are similar in shape during the first few hundred meters, a closer look reveals a noticeable offset, potentially causing issues when working with absolute pressure readings and altitudes, e.g. yielding an invalid floor number estimation for the pedestrian. However, due to aforementioned assumptions on dry air, fixed temperature and pressure at mean sea level, the more correct equations also represent an approximation. Every change in climatic conditions will void these assumptions, producing incorrect results, decreasing or increasing local pressure indications, as visualized in figure 2.16.

For compensation, pilots e.g. adjust their plane's altimeter to a reference pressure during takeoff and landing, currently measured at the corresponding airport, known as *QNH* or *QFE*. In doing so, the instrument denotes the height above the airport with respect to the current weather conditions [Eur04]. During flight, the exact altitude is less important, as long as obstacles like
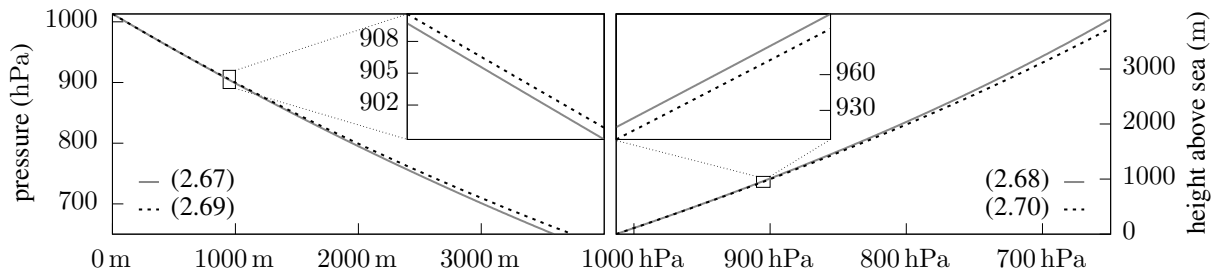
Figure 2.15: Relation between atmospheric pressure and height above sea level according to (2.67), (2.69) and (2.68), (2.70). As can be seen within the zoomed regions, the approximation is only suited for relative comparisons due to absolute errors of several meters.

mountains are safely overflown. The distance towards other airplanes nearby is more important, and pilots therefore adjust their altimeter to the same reference of $1013.25\,\mathrm{hPa}$. While the indicated height above ground will change with weather conditions, nearby aircrafts are influenced by the same effect, ensuring that their distance in altitude remains measurable [Fed16].

Probabilistic Assembly   Azevedo and Crisóstomo propose a similar approach for smartphone based localization. The current relative pressure at several known points is recorded and shared with the phone, to provide a relative alignment. This data is e.g. provided by nearby weather stations and hereafter interpolated to estimate the pressure at the building's altitude [AC16]. Or the weather station resides within the building itself [EBS16]. Instead of using the pressure readings from nearby transmitters, readings from the phone itself can also be used. Here, an initial measurement ϱ_{ref} from the barometer at a known altitude h_{ref} serves as reference. This also prevents potential calibration offsets, mitigates the impact of current weather conditions, and does not require additional infrastructure or a data connection to a server. The altitude h_{ref} of this reference measurement can e.g. be inferred by using the last GPS fix before entering the building [AY12], performing an optical localization using the phone's camera [HB08], the pedestrian scanning nearby QR-Codes or RFID/NFC tags [Lee+14; Jim+12].

In order to include an obtained reference, (2.67) and (2.68) must be adjusted, to operate relative to an arbitrary altitude instead of the mean sea level [OAS76]. The current altitude can hereafter be estimated directly based on incoming sensor readings, as long as local climatic conditions are not changing, and the reference remains valid [KWS12]

$$f_{\varrho}\left(h\right) = \varrho_{\mathrm{ref}}\left(\frac{T_0}{T_0 + L(h - h_{\mathrm{ref}})}\right)^{\frac{g\,M}{R\,L}} \qquad f_h\left(\varrho\right) = h_{\mathrm{ref}} + \frac{T_0}{L}\left(\left(\frac{\varrho}{\varrho_{\mathrm{ref}}}\right)^{\frac{-R\,L}{g\,M}} - 1\right). \quad (2.71, 2.72)$$

The readings from the barometer can hereafter be converted to an altitude, and compared against potential whereabouts. However, all barometer altitudes are with respect to mean sea level, and the pedestrian's unknown z-location with respect to e.g. the first floor of the building. Thus, an
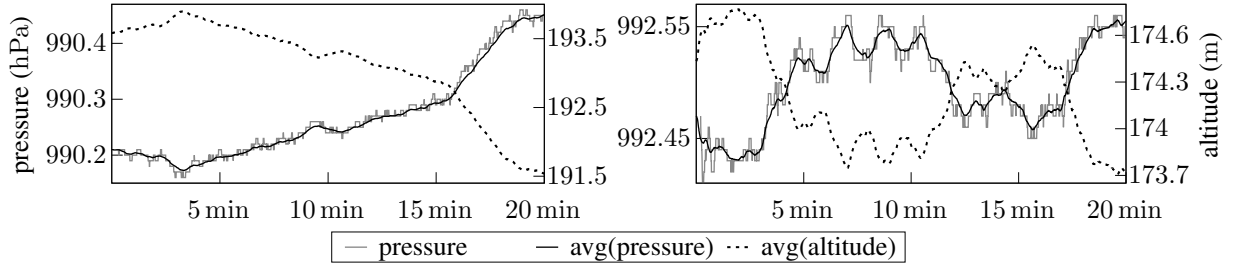
Figure 2.16: Readings from a barometer at a fixed location on two different days (left and right). While the left shows a significant variation of $> 2.5\,\text{m}$ in altitude, the right one denotes only $\approx 1\,\text{m}$ of variation. However, due to different climatic conditions there is an absolute difference of $\approx 20\,\text{m}$ between both.

offset constant h_{bldg} is required, to relate both. Independent of whether (2.68), (2.70) or (2.72) is used, the pedestrian's potential whereabouts in z can then be compared against the barometer's readings, e.g. by using a normal distribution to model sensor uncertainties

$$p_{\text{baroAbs}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \mathcal{N}\left(q_t^{(z)} + h_{\text{bldg}} \;\middle|\; f_h\left(o_t^{(\varrho)}\right), \sigma_{\text{alt}}^2\right)$$

$$\langle \boldsymbol{q}\rangle_t = \langle(z, \ldots)\rangle_t, \quad \langle \boldsymbol{o}\rangle_t = \langle(\varrho, \ldots)\rangle_t.$$

(2.73)

As discussed, a pressure reference requires either additional hardware or infrastructure within the building, or the correct altitude of a reference measurement observed by the phone itself. To drop those requirements, but still benefiting from the advantages of relative pressure readings, a workaround is required. By changing (2.73) from an absolute to a relative comparison, with respect to the first smartphone pressure reading, the reference becomes obsolete. In other words, an initial atmospheric pressure is known from the barometer, but the corresponding altitude is not. That is, the pedestrian is assumed to have started at an unknown height. While walking, the sensor's pressure readings, and the pedestrian's altitude, both change with respect to their initial values. Even though the starting altitude is unknown, the evaluation can examine if the pedestrian's *change* in altitude since the start at $t = 0$ resembles the barometer's *change* in pressure since $t = 0$ [Ebn+15; Tor+17].

To compare pressure changes with altitude changes, the *relative* relation between both is required. That is, the change in pressure per meter of altitude (hPa/m) at a certain altitude h above sea-level, given by the derivative of (2.67)

$$\dot{f}_\varrho\left(h\right) = -\frac{gM\varrho_0 T_0 \left(\frac{T_0}{T_0 + Lh}\right)^{\frac{gM}{LR} - 1}}{R\left(T_0 + Lh\right)^2}.$$

(2.74)

For buildings with rather few floors, only a small range of the function is required. Within this range, (2.67) can be assumed to behave linearly, and its derivative (2.74) denotes a constant,

Altitude h	$f_\varrho(h)_{(2.67)}$	$\dot{f}_\varrho(h)_{(2.74)}$	$1/\dot{f}_\varrho(h)_{(2.74)}$
0 m	1013.25 hPa	0.120 hPa/m	8.32 m/hPa
200 m	989.56 hPa	0.117 hPa/m	8.56 m/hPa
400 m	966.53 hPa	0.114 hPa/m	8.81 m/hPa
600 m	944.13 hPa	0.110 hPa/m	9.05 m/hPa

Table 2.3: Expected atmospheric pressure and corresponding change rate, dependent on the altitude.

solely dependent on the building's height above mean sea level h_{bldg}. Resulting change rates for typical altitudes are shown in table 2.3. The adjusted evaluation

$$p_{\text{baroRel}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \mathcal{N}\left(\overbrace{\frac{\Delta\varrho}{\dot{f}_\varrho\left(h_{\text{bldg}}\right)}}^{\Delta\text{hPa}\rightarrow\Delta\text{m}} \,\middle|\, \Delta z, \sigma_{\text{alt}}^2\right) \quad (2.75)$$

$$\langle\boldsymbol{q}\rangle_t = \langle(z,\dots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\varrho,\dots)\rangle_t, \quad \Delta z = q_t^{(z)} - q_0^{(z)}, \quad \Delta\varrho = o_t^{(\varrho)} - o_0^{(\varrho)},$$

matches the pedestrian's potential whereabouts against current sensor readings, by comparing the change in altitude Δz since start $t = 0$ with the change in atmospheric pressure $\Delta\varrho$ since start. For comparing, the latter is converted to meters based on the expected pressure change per meter at the building's altitude h_{bldg}, given by the reciprocal of (2.74). To remove the required $\boldsymbol{o}_0$ from (2.75), a virtual barometer sensor can be created, directly providing the required $\Delta\varrho$ as its observation. Similarly, Δz can be added directly to the unknown state, and is hereafter updated for every movement $\boldsymbol{q}_{t-1} \rightarrow \boldsymbol{q}_t$, dropping the requirement for $\boldsymbol{q}_0$. However, even with pressure readings relative to the start of the walk, changing weather conditions, opening/closing doors or windows, can still cause several meters of fluctuation. Changing weather conditions usually take at least several minutes to manifest. Those long-term changes become relevant for longer localization and navigation scenarios, where the initial reference ϱ_{ref} or $o_0^{(\varrho)}$ gets invalid. This can e.g. be addressed by not using $t = 0$ as reference within (2.75), but a short timeframe of several seconds, sufficient for determining floor changes [Mur+14].

Ye et al. [Ye+14] compensate ambient conditions by *crowdsourcing*, using measurements from other smartphones within the same building. Local temporal effects can be compensated, as they manifest similarly for all users. This allows to distinguish them from pressure changes due to stairs or elevators. Their approach could also be used to estimate σ_{alt}, which is barely possible without additional information. As depicted in figure 2.16, temporal effects can cause pressure changes similar to changes in altitude > 1 m, and a floor-correct estimation is not always possible [Mur+14]. When considering data from nearby pedestrian's, σ_{alt} can be adjusted accordingly, based on ambient conditions. Yet, due to the requirement of a server for exchang-

ing the data, this imposes potential data privacy issues, and requires other pedestrians using the same software within the same building at the same instant in time.

Another option for uncertainty estimation and/or suppression of temporal influences is similar to approaches described in section 2.4.3. By including additional sensors, it becomes more clear whether pressure is changing due to taking stairs/elevators, or temporal effects, as e.g. the pedestrian's step interval and pattern will change when stairs are involved (cf. section 2.4.1).

However, with ambient conditions changing rather slowly, it can also make sense to use the barometer only for short-term indications, to e.g. determine whether the pedestrian is currently taking stairs or not.

2.6   Activity-Detection

Previous components provided a continuous estimation of absolute and relative changes in position or heading, but suffered from various uncertainties, such as drifts, which can be hard to address. Besides a direct indication of location or heading, a discrete classification can be used, e.g. to determine the pedestrian's current *activity*. As mentioned for the barometer, due to environmental effects, it can be more robust to determine whether the pedestrian is currently taking a stair, than estimating a continuous change in altitude on a per meter basis [Fet+16; Ebn+17]. The same holds true for other sensors besides the barometer, allowing to distinguish between various types of activities for the pedestrian. Among the most common for indoor localization and navigation are *standing, walking, turning*, and using *stairs*, *elevators* or *escalators* [Fet+16; Elh+14; ABA11]. For an overview, the following activity classes Ω are distinguished

$$\Omega \in \{ \text{standing, walking, stair}\uparrow, \text{stair}\downarrow \} . \tag{2.76}$$

Recognition is based on sensor data, observed during a certain timeframe, estimating the activity that took place within. The actual classification can be as simple as a binary decision tree, with multiple yes/no paths based on some features, extracted from the windowed sensor data [Elh+14; Fet+16]. As described earlier in section 2.4.1, steps cause an observable fluctuation of the accelerometer's magnitude $\|\boldsymbol{a}\|$. The variance $f_\sigma(\boldsymbol{a}_{[]}, \Delta N)$ of all readings within a certain timeframe can thus be used to estimate whether steps were made or not. For better readability, the timeframe is defined by the most recent ΔN samples, received from the sensor

$$f_\sigma(\boldsymbol{a}_{[]}, \Delta N) = \sqrt{ \frac{1}{\Delta N} \sum_n \left(\|\boldsymbol{a}_{[n]}\| \right)^2 - \left(\frac{1}{\Delta N} \sum_n \|\boldsymbol{a}_{[n]}\| \right)^2 }$$

$$\text{with} \quad N - \Delta N < n \leq N , \quad N = |\boldsymbol{a}_{[]}| . \tag{2.77}$$
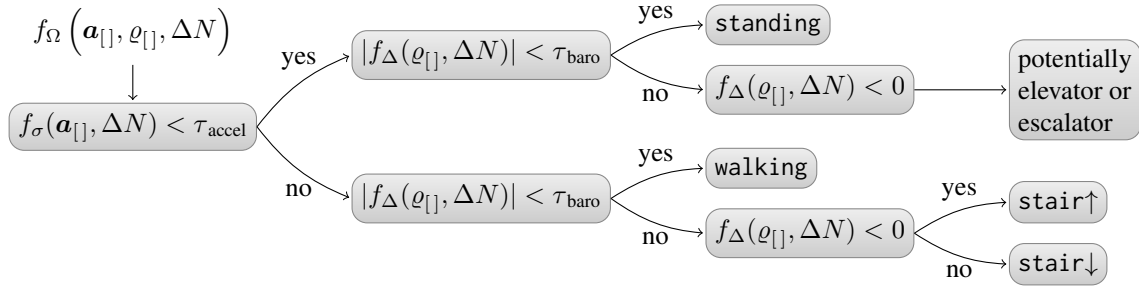
Figure 2.17: Binary decision tree example for activity recognition based on accelerometer and barometer. If there is few variation within the accelerometer's magnitude, the pedestrian is either standing, or e.g. taking an elevator or escalator, distinguishable by the barometer. If there is variation within the magnitude, this is either due to walking normally or taking stairs, distinguishable by the barometer.

If (2.77) is above a certain threshold, the pedestrian is expected to be currently walking along a floor or taking stairs. Accelerating and decelerating elevators also affect the magnitude, however, usually in a much less pronounced way than steps do [Zho+15]. It is important to notice that the two sums from (2.77) can suffer from precision issues, when implemented using floating point numbers. The result will be incorrect for $\sum x_{[]} \gg x_{[n]}$, that is, larger ΔN. This issue can be addressed by data types offering more precision, if available, or via *compensated summation*, e.g. by using the Kahan summation algorithm, tracking individual summation errors and adjusting a compensation value, included for the next addition [Hig02].

While both, step pattern and (2.77), are slightly different when taking stairs compared to walking along a hallway (cf. figure 2.7), information provided by the barometer is more valuable to distinguish between `walking` and `stair↑`/`stair↓`. The change of pressure within a short timeframe indicates changes in altitude, and is invariant to environmental long-term influences

$$f_\Delta(\varrho_{[]}, \Delta N) = \varrho_{[N]} - \varrho_{[N-\Delta N]}, \quad N = |\varrho_{[]}|. \tag{2.78}$$

Similarly, elevators should manifest as anomalies within the magnetometer's readings [She+09]. However, this is rarely described in literature, or the sensor's actual contribution is hidden behind machine learning and neural networks [Zha+18a]. Elevators can also be detected by examining precise gravity changes, yielding a hint on the change in altitude, by integrating the accelerometer's tilt compensated z-axis [Cil+14]. For a simple yes/no decision, elevators could be assumed when a change in altitude is indicated by the barometer, but no steps are detected.

Decision Tree Classification   Figure 2.17 shows a binary decision tree, combining the mentioned aspects to determine the current activity $\Omega = f_\Omega(\boldsymbol{a}_{[]}, \varrho_{[]}, \Delta N)$. The two required thresholds τ_{accel} and τ_{baro} are estimated empirically. To compare the detected activity with potential pedestrian movements, the building's floorplan needs semantic information on the type

of ground for every reachable location $\boldsymbol{\rho}$, indicating whether it belongs to a stair, escalator, elevator or normal ground [Ebn+17]. This information is assumed to be provided by a function $f_{\text{type}}(\boldsymbol{\rho})$. To support all activities from (2.76), two types are distinguished

$$\text{type} \in \{\texttt{floor}, \texttt{stair}\}, \tag{2.79}$$

where $\texttt{stair}$ is only used for the skewed stair parts, and plateaus are handled by the same type as normal ground floor. Potential locations within the building can then be evaluated by comparing the detected activity with a location's type. The uncertainty within the activity recognition is modeled by an empiric mixing value κ_{match}

$$p_{\text{activity}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t, \boldsymbol{q}_{t-1}\right) = \begin{cases} \begin{rcases} \kappa_{\text{match}} & d < \tau_{\text{dist}} \\ (1 - \kappa_{\text{match}}) & \text{else} \end{rcases} & o_t^{(\Omega)} = \texttt{standing} \\[2ex] \begin{rcases} \kappa_{\text{match}} & d > \tau_{\text{dist}} \wedge \text{type} = \texttt{floor} \\ (1 - \kappa_{\text{match}}) & \text{else} \end{rcases} & o_t^{(\Omega)} = \texttt{walking} \\[2ex] \begin{rcases} \kappa_{\text{match}} & d > \tau_{\text{dist}} \wedge \Delta z > 0 \wedge \text{type} = \texttt{stair} \\ (1 - \kappa_{\text{match}}) & \text{else} \end{rcases} & o_t^{(\Omega)} = \texttt{stair}\!\uparrow \\[2ex] \begin{rcases} \kappa_{\text{match}} & d > \tau_{\text{dist}} \wedge \Delta z < 0 \wedge \text{type} = \texttt{stair} \\ (1 - \kappa_{\text{match}}) & \text{else} \end{rcases} & o_t^{(\Omega)} = \texttt{stair}\!\downarrow \end{cases} \tag{2.80}$$

$$d = \text{dist}_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right), \quad \text{type} = f_{\text{type}}\left(\textbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_t\right)\right), \quad \Delta z = q_t^{(z)} - q_{t-1}^{(z)}$$

$$\langle\boldsymbol{q}\rangle_t = \langle(x, y, z, \ldots)\rangle_t, \quad \langle\boldsymbol{o}\rangle_t = \langle(\Omega, \ldots)\rangle_t.$$

To estimate the required thresholds τ_{accel}, τ_{baro}, τ_{dist} and the impact κ_{match}, labeled training data can be used. It is e.g. obtained from recorded pedestrians walks, where each part of the walk is labeled with the current activity. This data is hereafter passed through (2.77) and (2.78), to derive the two features after every sensor reading. All features for one activity can then be used to estimate thresholds, and plotting the entirety yields a hint on separability [Nie83].

Naive Bayes Classification   Instead of a binary decision tree, the training data can be used to derive one multidimensional normal distribution for each activity class Ω, by estimating the mean $\boldsymbol{\mu}_\Omega$ and covariance $\boldsymbol{\Sigma}_\Omega$ of all extracted features $\boldsymbol{c}$ belonging to it

$$p(\boldsymbol{c} \mid \Omega) = \mathcal{N}\left(\boldsymbol{c} \mid \boldsymbol{\mu}_\Omega, \boldsymbol{\Sigma}_\Omega\right), \quad \boldsymbol{c} = \left(f_\sigma(\boldsymbol{a}_{[]}, \Delta N)_{(2.77)}, f_\Delta(\varrho_{[]}, \Delta N)_{(2.78)}\right)^T. \tag{2.81}$$

In the simplest case, omitting any prior knowledge, the most likely activity Ω is the one where the associated normal distribution yields the largest result for a new feature $\boldsymbol{c}$

$$\Omega^* = \arg\max_\Omega \ \mathcal{N}\left(\boldsymbol{c} \mid \boldsymbol{\mu}_\Omega, \boldsymbol{\Sigma}_\Omega\right), \tag{2.82}$$

often referred to as Naive Bayes classifier. Required derivations, corresponding proof and reper-cussions can be found in [Nie83]. Ω^* from (2.82) can be used to replace the result of the decision tree within (2.80). However, doing so discards additionally available information, as only the most likely activity is considered, while others might be likely as well. This is especially important for a transition (2.6) , where potential movements are predicted based on the likelihood of current activities, derived from sensor readings. Depending on the estimated probability of each known activity, some transitions might use the stair, while others stay on the same floor, acknowledging the likelihood for every single activity

$$p_{\text{activity}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}\right) \propto \begin{cases} p(\boldsymbol{c} \mid \texttt{standing}) & \text{when } \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t \text{ is not moving} \\ p(\boldsymbol{c} \mid \texttt{walking}) & \text{when } \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t \text{ moves along floor} \\ p(\boldsymbol{c} \mid \texttt{stair}\!\uparrow) & \text{when } \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t \text{ moves upstairs} \\ \dots \; . \end{cases} \tag{2.83}$$

Besides indoor localization, the field of human activity recognition also attracts public interest, e.g. for supervising elderly people living on their own, where recognized activities are used to detect whether the person might have fallen, and needs assistance [KSG18]. With other types of sensors, such as wristbands, even typical household activities, like vacuuming, can be detected [ABA11]. This information can e.g. be used to detect whether elderly people can cope with their daily chores. Furthermore, detecting changes within the daily routine, gathered over longer periods, allows for early diagnostics of diseases. Using more sophisticated classification approaches such as the support vector machine (SVM), artifical/convolutional neural networks (ANN, CNN) and similar, even more fine grained activities, such as whether a drawer is being opened or closed, can be distinguished [Li+18]. Here, calculations like (2.77) and (2.78) are often not required, as the raw sensor data within some timeframe is used instead.

Li et al. [Li+18] compare a multitude of different approaches with varying number of sensors towards a detailed activity recognition. As expected, by using more sensors, more fine grained distinctions between activities become possible. With rising pervasiveness of smart-watches, and increasing computational power available from CPUs, GPUs and AI accelerators, aforementioned classification and recognition approaches become of interest for smartphone-based indoor localization and navigation as well.

Yet, while recognized activities can already be used to reduce the number of potential where-abouts, e.g. when taking stairs, elevators or escalators is detected, this technique does not provide absolute location hints in general. Thus, other sensors still are required to solve the overall problem of smartphone-based indoor localization and navigation.

2.7   Wi-Fi and Bluetooth Beacons

Aforementioned sensors either provided relative location changes, only limited hints on absolute indications, or were intended for outdoor use. Even if the pedestrian's initial position within the building is known, e.g. from the last GPS fix before entering, most sensors allow for incremental updates only. Any error of the initial whereabouts will thus propagate and, due to cumulating errors, increase over time. Therefore the question arises, whether there is a component that is able to provide absolute estimations for the phone's current whereabouts indoors, at least at a coarse level, but preferably as accurate as the GPS. Matching with section 1.1, this component should be available within every modern smartphone, and the requirements concerning the building itself should be as small as possible. Every architectural style should be supported, and it should not require costly hardware, setup or maintenance.

Back in 2000, Bahl and Padmanabhan conducted an experiment with three Wi-Fi transmitters, statically installed within a floor about $44 \times 23\,\text{m}$ in size. A portable receiver picked up those signals and measured their strength, referred to as *received signal strength indication* (RSSI). Due to physical effects and obstacles within the floor, the three measurable RSSIs vary depending on the receiver's location. The authors used this effect to roughly determine the receivers current location, by comparing the three received measurements to what should be measurable given a certain location. Thereby they created a positioning system, similar to the GPS, but operating indoors, working with almost every Wi-Fi equipped hardware [BP00].

As of today, every smartphone contains a Wi-Fi component, capable of measuring RSSIs by scanning for nearby transmitters. Likewise, the required infrastructure, namely Wi-Fi access points, is present within an increasing number of public buildings, where localization and navigation would provide a benefit, such as airports, rail stations, universities and hospitals. When this infrastructure is not yet available, such as in older museums, a cheap solutions is given by installing small, inexpensive transmitters within the area of interest [Fet+18]. One drawback of this approach is the time, and thus costs, required for an initial setup and maintenance in case of changes afterwards. The behavior of the signal strength has to be well known throughout the whole building, for the described location estimation to work.

To determine how this comparison is conducted, which components and information are required, the following sections will first examine the basic behavior of radio signal propagation. Hereafter, several comparison techniques with their benefits and drawbacks will be discussed, including required prior knowledge, and how to obtain it. Finally, several strategies to significantly reduce setup and maintenance times will be presented and discussed in detail.
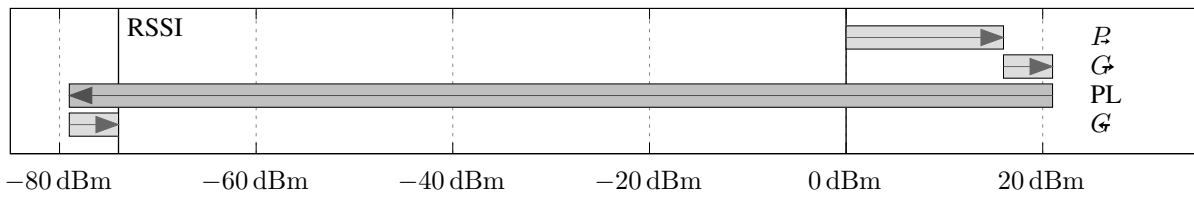
Figure 2.18: Influences on RSSI, adapted from [Rac07, p. 117]. P denotes the power used by the hardware, transmitting signals into the sending antenna. This antenna might bundle the energy, yielding an increase G. While traveling towards the receiver, the signal is attenuated by PL, increasing with distance. Finally, bundling by the receiving antenna yields another increase G, resulting in the final RSSI.

2.7.1 Signal-Strength and Propagation

The main metric in the approach presented by Bahl and Padmanabhan is the RSSI of the three transmitters, measured by the portable receiver. This value provides information on the strength of a signal that remains at the location it was picked up by a receiver. Generally, it decreases with increasing distance, thus providing a rough hint on the distance towards its transmitter. Both, the actual power of a radio signal, and the corresponding RSSI, are given in dBm or dB_{mW}. However, how the RSSI is measured is not standardized, and different devices can provide varying readings for the same absolute power [CB07]. Also, the signal's strength is strongly affected by the architecture between transmitter and receiver, varying with the receiver's whereabouts. Several studies thus concluded that the RSSI is too unreliable to provide a viable localization [PHU09; Lui+11; Jun+12]. While this is true for the analytical relation between RSSI and distance, depending on required accuracy, distance-based localization is still possible [BP00; YA05]. Furthermore, architectural effects often yield unique RSSI behavior, which can be used for matching approaches, to infer a location [JLH11; Sen+12; Zha+18b]. Nevertheless, due to the rising interest in location based services and similar, newer wireless standards aim to include additional metrics besides RSSI. For instance, travel-time measurement, especially targeting localization, addressing aforementioned drawbacks [BSA16; Ibr+18; Dvo+19]. Gradually, they might replace RSSI-based approaches, as soon as hardware and software are commonly available. For now, using the RSSI is still the most versatile approach.

Figure 2.18 gives an overview on all components influencing RSSI readings, for a signal traveling from a transmitter, called *access point* (AP) or *base station*, towards a receiver. The first component is the transmitter itself, where the strength is influenced by the configured transmission power P. While increasing the power yields an increased range for the signal, this factor is limited by local authorities to prevent pollution of radio frequencies and collisions between nearby devices. The generated signal is passed through an antenna, distributing it into the transmitter's surroundings, causing an additional gain G, depending on the antenna. Radio
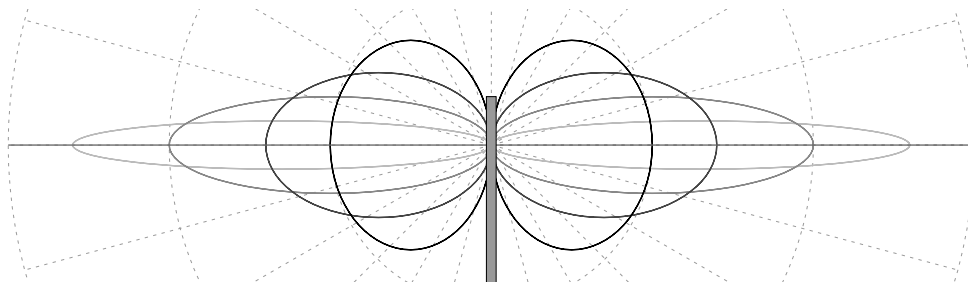
Figure 2.19: Simplified radiation pattern of a dipole antenna, dependent on the opening angle. The distance from the antenna denotes the amount of energy transmitted into a certain direction. Wider ellipses transmit more power to the horizontal plane than to the vertical plane.

waves hereafter propagate through air and architectural obstacles, losing a major part of their initial power, referred to as *path loss* (PL). The receiving antenna slightly increases the power by G_r, as multiple beams are picked up and accumulated. The receiving circuit then measures the signal's power, resulting in the RSSI. Whether this circuit recognizes a signal or not, depends on its remaining power. If it is below a certain threshold, or the amount of ambient noise is too high, there will be no indication at all [Rac07]. Due to environmental, physical and technical influences on radio waves, the resulting reception quality is often distributed unevenly among all places where it is required.

A countermeasure to weak signal qualities often suggested among hobbyists, is using a larger transmitter antenna, to increase G_t. This, however, does rarely work as expected and usually presents other drawbacks. As the transmission power P_t of the base station is constant, the overall output after an antenna also remains the same, independent of the chosen antenna. Different models only affect the way this input energy is distributed into free space. The most common antenna type for Wi-Fi is the dipole. It transmits the signals in a pattern often described as *"donut shaped"*, shown in figure 2.19. Dependent on the antenna's opening angle, the pattern approximates a torus, stretched into a certain direction. Antennae with a smaller opening angle distribute most of the energy to the horizontal plane, and the signal strength, measurable by devices on the same floor, will increase. Energy distribution to the vertical plane, or adjacent floors, however, is reduced. This is similar to the effect of a satellite dish, focusing the signal into a narrow beam, significantly increasing the transmission distance, without changing the overall amount of energy [CB07]. Yet, for a robust two-way communication, the receiver should use the same kind of antenna, to reliably transmit a response back to the transmitter. Antennae influence how available energy is distributed from a transmitter into its surrounding area, and how these radio signals are captured on the receiving side. It thus represents a crucial component when relating to signal strengths and RSSI.

After the antenna, each radio wave follows a straight line, until obstacles are encountered, affecting the signal in both, direction and strength. The latter is also reduced by air molecules, yielding an attenuation, increasing with distance. Combined with the attenuation by obstacles, this is referred to as path loss. Common effects on radio waves along their path are shown in figure 2.20 and described hereafter [Rac07; Sey05]:

Absorption/Shadowing   Architectural materials, but also humans, can *absorb* high frequency radio signals [Gra+11]. Within buildings, this often occurs for steel-reinforced concrete walls and floors. Dependent on the spacing of the contained steel grid, the wall's thickness, and used radio frequency, signals can be completely absorbed. Due to the wavelength's impact, $5\,\mathrm{GHz}$ signals, e.g. used within 802.11ac, behave differently for the same materials.

Reflection   Smooth surfaces, such as metal or glass, tend to reflect the majority of incoming radio waves. For rough surfaces, such as walls, reflection, if any, is usually less pronounced. The amount of reflection also depends on the signal's angle with respect to the surface. Some of the signal's energy is lost within this process, yielding a redirected, weaker signal. This can yield multiple "*copies*" of the original signal, reaching the recipient with differing delays.

Scattering   Similar to reflection is *scattering*. Instead of one reflection with a well-known angle, multiple reflections with varying angles are created. This mainly occurs among rough surfaces, where a smooth reflection is impossible. This effect can also be observed when the surrounding air is filled by many small particles, such as water. Rain, fog and snow will yield large amounts of reflection and scattering, often causing transmissions to fail.

Refraction   The effect of *refraction* is well-known from water surfaces, affecting the direction of light, due to a change in media. For radio signals, this mainly occurs among thick obstacles of a monotonic structure, such as solid concrete objects. When the signal leaves the wall, the direction is changed again. This also causes the signal's overall traveling distance to increase.

Diffraction   Obstacles are unable to occlude a light source in a binary on/off way. As light rays are able to slightly bend around corners, shadows do not exhibit sharp edges, but are faded. The same holds true for radio signals. After doors or corners, a slow fade in signal strength can be recognized between the line of sight towards the transmitter and the edge of the obstacle.

Attenuation/Free Space Loss   All aforementioned effects cause *attenuation*, reducing the signal's strength. However, due to the surrounding air molecules, attenuation also occurs along the line of sight between a transmitter and receiver. The amount of attenuation depends on the used wireless standard, and $2.4\,\mathrm{GHz}$ signals behave differently than $5\,\mathrm{GHz}$ versions [Hee+11].
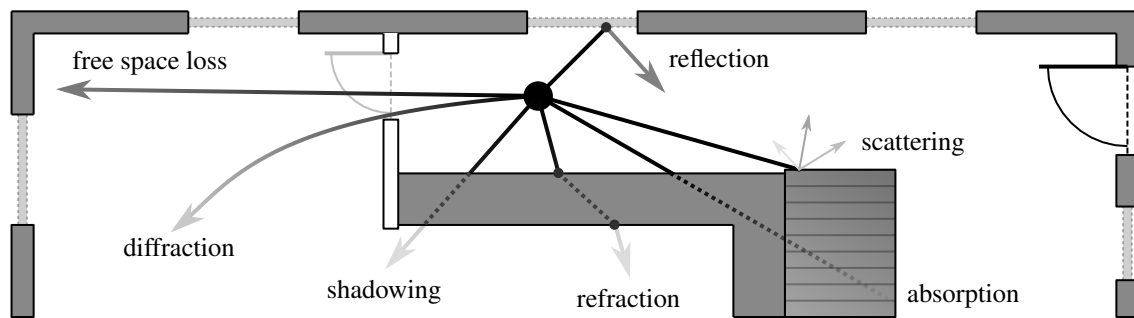
Figure 2.20: Effects on radio signals due to air and architectural components. While some influences just reduce the signal's strength, others also affect the direction, or split it into multiple, weaker signals.

Multipath Propagation   The effect of multiple signal copies reaching the recipient via different paths, is called *multipath propagation*. The delay and attenuation along each path is crucial. If two copies are equal in strength, but delayed by half the wavelength, they cancel each other out. Likewise, it is possible that the first signal reaching the recipient isn't the strongest signal, as there might be longer paths with less attenuation. Therefore, multipath propagation plays a vital role concerning reception quality, and indicated strength of the received signal [Dvo+19].

After propagation, signals are picked up by the receiving antenna and hardware. For the latter, the Wi-Fi standard defines minimum requirements regarding acceptable error rates for certain signal strengths [IEE12; IEE16]. These are a hint towards receiver sensitivity, and the weakest signal that will be receivable by hardware. Yet, actual hardware might be better than the minimum requirements demand. Thus, a weak $-90\,\mathrm{dBm}$ signal might be recognized by one smartphone, but not by another [Liu+07]. This aspect imposes potential issues when comparing RSSI readings to infer the current location.

Figure 2.21 shows real-world RSSI readings and aforementioned influences. Even for direct line of sight conditions between a transmitter and a smartphone, used as receiver, signal strength readings are varying (figure 2.21a). If pedestrians are crossing this line of sight, the average signal strength is slightly reduced and the variance around this average value is increased (figure 2.21b). This clearly depicts the mentioned influence of the pedestrian. Similarly, when holding the smartphone while walking along a hallway, all transmitters behind the pedestrian will be attenuated, yielding smaller RSSIs [Gra+11]. This effect can even be used for passive localization. Youssef et al. [YMA07] presented a system with several transmitters and receivers, all installed at well-known locations. Humans walking between them affect the measurable signal strengths, allowing for wireless human presence detection, and even a coarse location estimation. The impact of antennae is shown in figure 2.21c, where the phone, and thus its antenna, is slowly rotated, at a constant distance of $10\,\mathrm{m}$. While the change of the average value is insignificant, rotating the antenna increases the variance significantly. Most readings are within
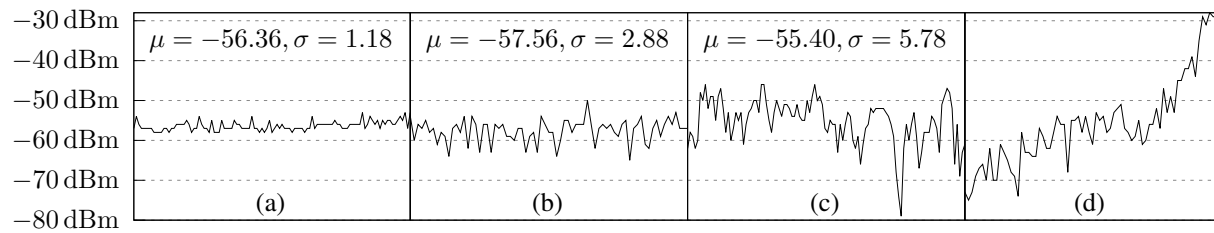
Figure 2.21: Time behavior of measurable signal strength outdoors under line of sight conditions with 10 m distance between phone and transmitter (a), 10 m distance and pedestrians walking between transmitter and phone (b), 10 m distance and changing the phone's orientation (c), a pedestrian holding the phone and walking towards the transmitter (50 m to 1 m) at a constant speed (d).

a $\pm 10\,\mathrm{dB}$ boundary around the mean, with one $-20\,\mathrm{dB}$ outlier. For comparison, figure 2.21d shows the readings for a pedestrian, $50\,\mathrm{m}$ away from the receiver, walking towards it. The signal strength of the outlier from figure 2.21c is below the one at a distance of $50\,\mathrm{m}$.

As shown, the value of RSSI readings depends on surroundings, hardware and antennae. The next section revisits aforementioned aspects from an analytical viewpoint, discussing models that can be used to predict signal strength behavior within buildings.

2.7.2 Signal-Strength Prediction Models

As discussed, free space and obstacles attenuate radio waves during their propagation. These effects can be simulated by signal strength prediction models, providing an estimation of the signal strength, that is expectable at a given distance or location. Such models are often used to predict where, and how many, transmitters should be installed, when buildings are newly equipped with Wi-Fi [Raj+96]. However, as discussions will show, they can also be used for Wi-Fi-based indoor location estimation.

Simple models only focus on aforementioned free space loss, within line of sight conditions, where the signal is solely attenuated by the surrounding air. For walls, floors and other obstacles to be considered as well, more advanced models are required, to include corresponding effects on radio waves. This section provides an overview on models often used in the context of Wi-Fi and indoor localization. While simple free space models offer analytical benefits, they are not well suited for more complex architecture. Therefore, additional parameters and techniques to approximate the buildings architecture will be discussed as well.

Log-Distance Model Most of the simple models are intended for outdoor use, line of sight conditions, and are related to *Frii's transmission equation* [Fri46]:

$$\frac{P_r}{P_t} = \frac{\lambda^2 G_r}{4\pi} \frac{\lambda^2 G_t}{4\pi} \frac{1}{d^2 \lambda^2} = G_t G_r \left(\frac{\lambda}{4\pi d} \right)^2 . \tag{2.84}$$

	Office 1	Office 2	Grocery Store	Retail Store	Textile	Metalworking
MHz	914	914	914	914	4000	1300
γ	3.5	4.3	1.8	2.2	2.1	3.3
σ_{rssi}	12.8 dB	13.3 dB	5.2 dB	8.7 dB	9.7 dB	6.8 dB
	[SR92]		[SR92; ARY95]		[ARY95]	

Table 2.4: Typical values for γ and uncertainty σ_{rssi} within (2.85), both varying significantly based on surroundings and used radio frequency. Details on the actual structure of the listed environments are of less importance here, but can be found within the provided sources.

(2.84) describes the ratio between the power $\mathcal{P}$ used by the transmitter and the power $\mathcal{P}$ remaining at the receiver, dependent on their antennae, used wavelength λ and the distance d between both devices. Antennae are modeled using aforementioned gain factors $\mathcal{G}$ and $\mathcal{G}$, which depend on the chosen type [Fri71]. For isotropic antennae, which distribute energy equally into all directions, $\mathcal{G}, \mathcal{G} = 1$. The wavelength λ depends on the used Wi-Fi standard (2.4 GHz/5 GHz). Both power indications, $\mathcal{P}$ and $\mathcal{P}$, are absolute, and given in dBm. The equation indicates a direct connection between the RSSI, given by $\mathcal{P}$, and the distance d from a transmitter.

For Wi-Fi and indoor use, (2.84) is modified, to approximate the impact of air and obstacles along the line of sight, using an *attenuation factor* γ. Put simply, γ describes the average signal attenuation per meter. This modified version is often referred to as *log-distance model* [Rap02]

$$\mathcal{P}(d) = P_0 - 10\gamma \log_{10} \frac{d}{d_0} \overbrace{+ \, \mathcal{X}}^{\text{optional}}, \quad \mathcal{X} \sim \mathcal{N}(0, \sigma_{\text{rssi}}^2). \tag{2.85}$$

Here, the power $\mathcal{P}$ measurable by a receiver, depends on the distance d from the transmitter, the attenuation factor γ, and a reference P_0. The latter describes the signal strength, measurable at a known distance d_0 from the transmitter's antenna, serving the same purpose as $\mathcal{P}$ within (2.84). In literature, P_0 is often given at a distance $d_0 = 1\,\text{m}$ [SR92; Sey05]. As mentioned earlier, $\mathcal{P}$, and thus P_0, depend on regulations made by local authorities. Typical values for $\mathcal{P}$ of the hardware circuit are around 15 dBm to 23 dBm ($\approx 32\,\text{mW}$ to 200 mW), and -40 dBm to -20 dBm for P_0, after a distance of one meter, when using typical Wi-Fi antennae [Liu+07; Hee+11; Che+12; OCV12]. Uncertainties due to noise and ambient conditions, such as multipath effects, scattering and similar (cf. figure 2.21) are modeled by the zero mean Gaussian random variable $\mathcal{X}$. When this variable is present, the model is also referred to as *log-normal shadowing model* [Rap02]. The degree of attenuation per increase in distance from the transmitter is modeled by the path loss exponent γ, where higher values denote an increased attenuation, similar to more obstacles along the line of sight. Typical values for γ and σ_{rssi} are shown in ta-
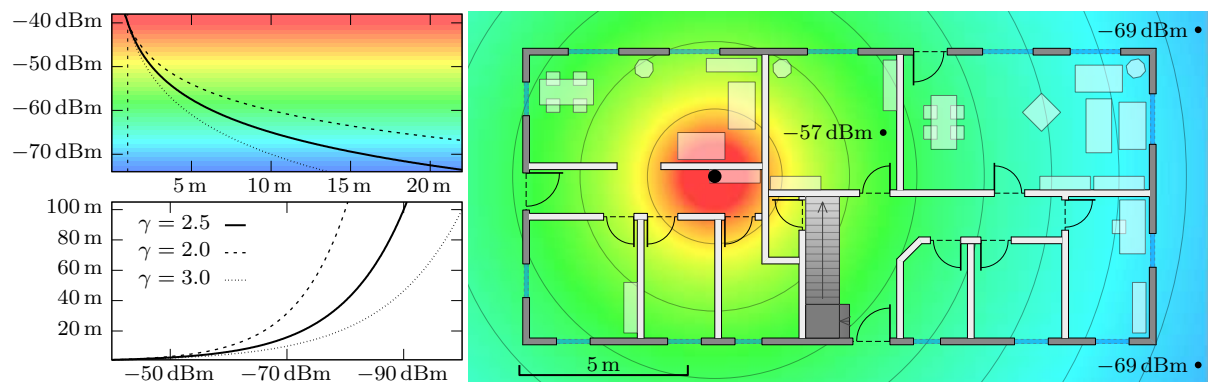
Figure 2.22: Signal strength prediction heat map for a transmitter (black dot), using (2.85) or (2.87) with $\gamma = 2.5$ and $P_0 = -40\,\mathrm{dBm}$ at $d_0 = 1\,\mathrm{m}$. The reference-circles are $2\,\mathrm{m}$ apart. The charts on the left depict the relation between signal strength and distance, when varying the path loss exponent γ,

ble 2.4, whereby the actual structure of the listed surroundings is unimportant. But, as can be seen, due to different architectural materials, like drywall, metal and concrete, the number of obstacles, and their placement, values are varying significantly, including a notable uncertainty.

Similar to (2.84), (2.85) clearly relates the power measurable by a receiver to its distance from the transmitter. This relation can be inverted, to directly convert a signal strength reading into an estimated distance towards its transmitter, based on aforementioned model parameters

$$d_{\mathrm{LD}}\left(P\right) = d_0\, 10^{(P_0 - P + \mathcal{X})/(10\gamma)} . \tag{2.86}$$

As can be seen, (2.86) is a crucial component towards absolute location estimation based on the measurable signal strength of radio waves. Omitting $\mathcal{X}$ for now, a single RSSI reading can directly be converted into a distance. If the location of the signal's transmitter is known, the location of the receiver can be constrained using (2.86), similar to earlier for the GPS (2.7). Potential whereabouts then denote a circle around the transmitter for the 2D case, and a sphere for the 3D variant, with the radius equal to the estimated distance. By using measurements from several transmitters, multilateration (see section 2.3) provides a coarse location estimation for the receiver, e.g. a smartphone, depending on the model's correctness.

For an impression of the model's quality, it is applied to an example floorplan with drywall interior and concrete exterior, using a typical transmission power of $P_0 = -40\,\mathrm{dBm}$ at $d_0 = 1\,\mathrm{m}$, and a path loss exponent $\gamma = 2.5$. The results are depicted in figure 2.22. As can be seen, the attenuation always follows the shown curves. Neither walls nor windows affect the signal strength. γ approximates all of the transmitters surroundings as a single value, without any locality. While this will yield large regional errors, not matching real-world behavior, this location invariance is the basis, required for the existence of the inverse equation (2.86).

Dependent on the use case, more accurate model predictions, considering walls and other obstacles, can be required. To include local elements, the transmitter's and receiver's locations must be part of the equations. Within the following, d is therefore replaced by the Euclidean distance between the known location $\boldsymbol{\rho}$ of the transmitter and the location $\boldsymbol{\rho_\bullet}$ to estimate the signal strength for. To ensure readability, all parameters required for a certain model are grouped within the tuple $\boldsymbol{\psi}$, and $d_0 \overset{!}{=} 1\,\mathrm{m}$, to remove the fraction, changing (2.85) to

$$P_{\mathrm{LD}}\left(\boldsymbol{\rho_\bullet}, \boldsymbol{\psi}\right) = P_0 - 10\gamma \log_{10}\left\|\boldsymbol{\rho} - \boldsymbol{\rho_\bullet}\right\| + \mathcal{X}$$
$$P_0, \gamma, \boldsymbol{\rho}, \mathcal{X} \in \boldsymbol{\psi}, \quad \mathcal{X} \sim \mathcal{N}(0, \sigma_{\mathrm{rssi}}^2). \tag{2.87}$$

Extended Log-Distance Model Especially within larger buildings, using a mixture of massive concrete and decent drywalls, the approximation given by the single path loss exponent γ will yield model predictions that are too strong behind concrete walls, and too weak for the remaining regions of each floor. Therefore, Seidel and Rappaport proposed an extension to the log-distance model, including local effects of floors and walls, named *floor attenuation factor* path loss model (FAF) [SR92]. This extension was picked up by the work of Bahl and Padmanabhan. They slightly adjusted the model and named it *wall attenuation factor model* (WAF) [BP00]. The idea behind both versions is basically the same, and given as

$$P_{\mathrm{eLD}}\left(\boldsymbol{\rho_\bullet}, \boldsymbol{\psi}\right) = P_0 - 10\gamma \log_{10}\left\|\boldsymbol{\rho} - \boldsymbol{\rho_\bullet}\right\| + \Gamma(\boldsymbol{\rho}, \boldsymbol{\rho_\bullet}, \boldsymbol{\phi}) + \mathcal{X}$$
$$P_0, \gamma, \boldsymbol{\rho}, \mathcal{X}, \boldsymbol{\phi} \in \boldsymbol{\psi}. \tag{2.88}$$

The additional $\Gamma(\boldsymbol{\rho}, \boldsymbol{\rho_\bullet}, \boldsymbol{\phi})$ denotes attenuations by floors, walls and other obstacles, blocking the line of sight between the transmitter and the location, the signal strength is estimated for. Based on the type/material of the obstacle, a certain amount is *removed* from the signal strength. This *negative attenuation factor* for each material – like drywall, glass or concrete – must be chosen beforehand, either by measuring, or using values from literature. After determining all obstacles within the line of sight, each single object contributes to the signal strength estimation, by removing a constant factor based on its type, briefly described as

$$\text{isects} = \{\text{all walls intersecting the line from } \boldsymbol{\rho} \text{ to } \boldsymbol{\rho_\bullet}\}$$
$$\Gamma(\boldsymbol{\rho}, \boldsymbol{\rho_\bullet}, \boldsymbol{\phi}) = \sum_{\mathrm{i} \in \text{isects}} f_\phi\left(\mathrm{i}\right), \qquad \boldsymbol{\phi} = \left\{\phi_{\mathrm{ceiling}}, \phi_{\mathrm{concrete}}, \phi_{\mathrm{drywall}}, \dots\right\} \tag{2.89}$$
$$f_\phi : \mathrm{i} \mapsto \phi \in \boldsymbol{\phi}.$$

(2.89) determines all obstacles intersecting the line of sight between $\boldsymbol{\rho}$ and $\boldsymbol{\rho_\bullet}$, and sums their attenuation factors ϕ, describing each material's attenuation as a negative number.

	Window	Drywall	Door	Brickwall	Concrete	Wire Frame
[Wil02]	-0.5	-0.5		-4.4		-21.0
[Ric+00]	-6.4		-2.6	-5.9	-7	
[Rac07, p. 116]	-2 to -8		-2 to -4	-5 to -8	-10 to -15	-5 to -8

Table 2.5: Attenuation factors ϕ for typical building materials, examined using different measuring methods, under varying ambient conditions.

Regarding the influences described earlier, (2.88) additionally models the effects of absorption and shadowing. While reflection, refraction, diffraction and scattering are still omitted, this represents a major improvement compared to (2.87). This, however, comes at the cost of drawbacks. To determine the attenuation by walls and floors (2.89), the building's floorplan must be known, including semantic information. While a scanned image can be sufficient for intersection testing, it does not provide semantic information on used materials, thus enforcing the same attenuation factor for every obstacle, representing an unnecessary approximation [BP00]. Another drawback of (2.88) is its irreversibility. It can not be solved for a distance d or location ϱ, due to the required intersection tests. Furthermore, these tests are costly, especially with increasing complexity of the floorplan, thus potentially unsuited for embedded use [Ebn+15]. Depending on required accuracy, a compromise might be to only include floors and concrete walls, omitting minor obstacles with small attenuation values. These values can either be determined by measurements, or taken from literature. As can be seen in table 2.5, however, there is a large variation among different sources. This is due to variations within the architectural structure, real vs. laboratory conditions, and the used signal frequency [Wil02].

Figure 2.23 depicts a heat map for the example floor when using (2.88). Compared to figure 2.22, the value for γ is reduced to an empirical 2.2, and mainly models free space loss, as wall attenuations are now given separately. White elements depict drywalls, dark gray elements represent concrete walls, including cutouts for windows. Interior obstacles, such as tables and chairs, are not considered by this propagation model. While they do affect the signal strength [Raj+96], including them within the model would require even more computational power. Furthermore, their location is volatile, requiring maintenance, whenever furniture is relocated. Thus, only static furniture, such as cabinets within a museum, should be considered [Fet+18]. Examining the chart in the left half of Figure 2.23, the behavior of the extended model is clearly visible. As soon as a wall is encountered, the estimated signal strength is reduced by the obstacle's constant attenuation factor. For the depicted example, (2.87) and (2.88) provide similar estimations, until the concrete exterior is encountered, $\approx 13\,\mathrm{m}$ along the line of sight.
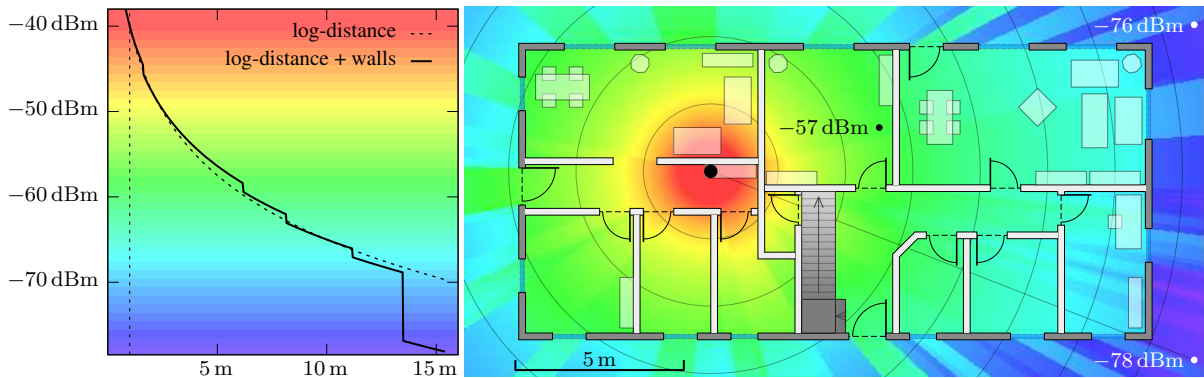
Figure 2.23: Signal strength prediction for a transmitter (black dot), using (2.88) with $\gamma = 2.2$, $P_0 = -40\,\mathrm{dBm}$, $\phi_{\mathrm{drywall}} = -1\,\mathrm{dB}$, $\phi_{\mathrm{door}} = -3\,\mathrm{dB}$, $\phi_{\mathrm{window}} = -4\,\mathrm{dB}$, $\phi_{\mathrm{concrete}} = -8\,\mathrm{dB}$. The left depicts a comparison between (2.87) and (2.88) along the path towards the lower right reference, clearly indicating the concrete wall's attenuation.

Ray Tracing   To further enhance predictions, including reflection and refraction, more complex approaches, not relying on a line of sight assumption, must be used. One potential candidate is ray tracing, known from computer vision, e.g. used for rendering photo-realistic 3D images [Shi03]. Here, one ray per pixel is sent from a virtual camera, eventually colliding with objects, placed within a virtual scene. After hitting an object, its color and lighting information is determined, based on the location and angle it was hit. The same technique can be used to simulate signal strength propagation within a building, by emitting several rays from a transmitter's location. As soon as a ray encounters an obstacle, the collision angle is calculated to determine further actions. Depending on this angle, and the encountered material, the signal is shadowed, absorbed, reflected or refracted, by emitting new rays, starting at the collision, into a direction defined by the physical effect to simulate. This process continues, until the way along consecutive rays reached a certain length, and the signal is too weak to be recognized. Actual signal strengths are then determined similarly to the extended log-distance model, depending on the distance, encountered obstacles, and reflections along a ray. Most locations are crossed by more than one ray. Their signal strength is thus e.g. given by the maximum encountered, or the one from the first ray reaching it, that is, the one with the smallest distance towards the transmitter [ElK+10; Raj+96].

Figure 2.24 depicts a potential prediction result, using the same transmitter parameters and attenuation factors as figure 2.23. Both results are similar, but differ in some regions, especially visible near windows. Signal strengths were estimated by rasterizing all emitted rays. When some location within the building is not traversed by a ray, its signal strength remains unknown. Thus, this approach does not allow for determining the signal strength for arbitrary positions, but relies on a computationally complex simulation for the whole building. To perform actual
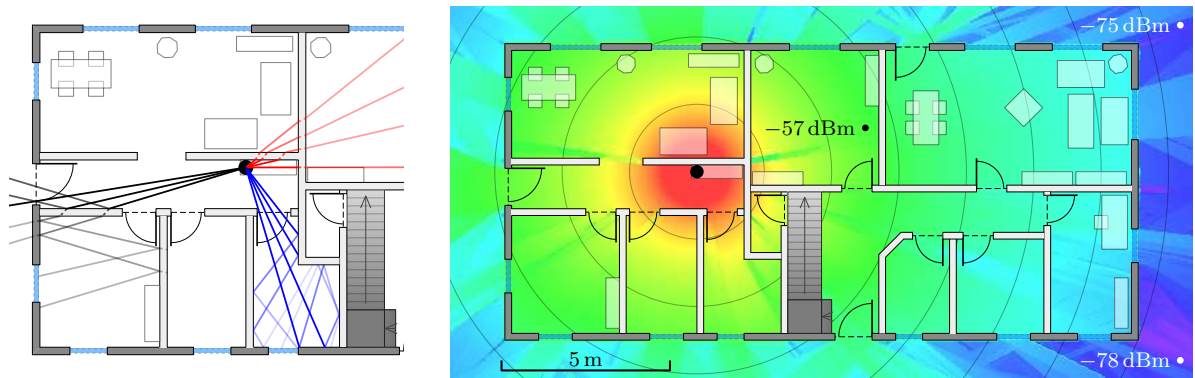
Figure 2.24: General ray tracing procedure (left) with refraction (red), reflection (blue) and both combined (black). The corresponding signal strength prediction (right) for the transmitter located at the black dot uses $\gamma = 2.2$, $P_0 = -40\,\text{dBm}$, $\phi_{\text{drywall}} = -1\,\text{dB}$, $\phi_{\text{door}} = -3\,\text{dB}$, $\phi_{\text{window}} = -4\,\text{dB}$, $\phi_{\text{concrete}} = -8\,\text{dB}$.

lookups, this prediction is calculated once, and persisted hereafter. The amount of required memory depends on the size of the building, and the number of installed transmitters. Furthermore, this technique requires an accurate floorplan representation, including semantic details on obstacle material and thickness. Especially for the 3D case, walls and floors must be modeled in a non-intersecting way, for the ray tracing to correctly determine when the obstacle is entered and left, requiring a complex and semantic 3D model. Hence, ray tracing is mentioned for completeness, but focus remains on aforementioned signal strength prediction models.

2.7.3 Probabilistic Location Estimation

Having discussed signal propagation and prediction by models, localization strategies are now examined. Here, three techniques, varying in complexity and accuracy, will be distinguished.

The simplest approach uses *lateration* via (2.86), to provide a coarse location estimation, only requiring each access point's position within the building, and two parameters P_0 and γ. However, depending on architecture and use case, results can be too vague [Ebn+17].

To overcome these limitations, and include surrounding architecture, a discrete approach can be used, where real-world measurements throughout a building serve as reference. Recording such initial measurements once, is often referred to as *offline* phase [YA05]. Localizing the pedestrian then represents the *online* phase, where current smartphone readings are compared against the offline-database, to determine the most likely whereabouts [BP00]. In literature, this is referred to as *fingerprinting*, as each location within the building presents a unique combination of visible transmitters and signal strengths [CPP10; Lui+11]. While being accurate, due to real-world measurements, initial setups are time-consuming, costly and hard to maintain.

Therefore, a compromise between simple lateration and costly fingerprinting will be discussed, allowing for rapid and inexpensive setups within new environments, still providing viable accuracy for most use cases [Ebn+15; Ebn+17]. Due to aforementioned influences on radio signals, several drawbacks need to be addressed, independent of the chosen localization procedure. Techniques to compensate uncertainty must be examined as well as approaches mitigating the influence of different software, hardware and antennae [YA05; Ebn+17; Fet+17].

To conduct a smartphone-based Wi-Fi signal strength location estimation, the current RSSIs of all nearby transmitters are required, including an *unique* identifier for each of them. In case of Android, this is e.g. achieved by triggering a Wi-Fi scan, searching for all access points nearby, returning their unique MAC address and corresponding RSSI. In general, two different scan-types are available: *active scan*, which is the default, and *passive scan* [Baw+15; YA05]. By briefly discussing the two, some limitations on Wi-Fi location estimation are made clear.

For the active variant, the Wi-Fi hardware selects a channel to scan, broadcasts a request and waits a short period for access points to respond. Transmitters might be overlooked, dependent on the time waiting for this response, the number of nearby devices and package collisions. This procedure must be repeated for every available Wi-Fi channel, which depends on the used Wi-Fi standard, and local regulations [IEE07]. For $2.4\,\text{GHz}$, usually 13 or 14 channels are available. For the $5\,\text{GHz}$ variants, even more channels are usable, but local regulations are strongly varying. European regulations on available channels and allowed transmission powers are given in [ETS12]. Depending on the hardware component, not all of the allowed channels will be available. The number of $5\,\text{GHz}$ channels thus can be up to approximately 30, requiring more time for scanning than $2.4\,\text{GHz}$. While a `Google Nexus 6` takes around $600\,\text{ms}$ to scan the $2.4\,\text{GHz}$ band for nearby transmitters, a `Samsung Galaxy S5`, scanning both $2.4\,\text{GHz}$ and $5\,\text{GHz}$, needs around $3500\,\text{ms}$. Scanning times thus add a noticeable delay to the localization.

The passive variant relies on *beacon frames*, periodically sent by access points, usually every $100\,\text{ms}$, carrying similar information as responses to scan requests. Passive scans take longer, as each channel must be surveyed for at least $100\,\text{ms}$, to reliably detect all transmitters. However, not actively probing keeps the radio channel clean, as no additional frames are transmitted, which can be a benefit, when many smartphones are using the system. As of today, this variant is not available for most smartphone's firmware, and can only be used by dedicated hardware.

Throughout the following, the current signal strength readings for nearby transmitters measured by the pedestrian's smartphone are given by

$$\boldsymbol{s} = \left(s_{\text{ap}_1}, s_{\text{ap}_2}, \ldots\right),\tag{2.90}$$

where the MAC address allows for a direct mapping between any s_{ap} and its transmitter, including all required parameters, such as its position. For fingerprinting, training of signal strength prediction models, and performance evaluation, additional reference measurements throughout a building are required. They are conducted by placing a receiver at various locations within the building, and scanning for nearby transmitters multiple times, to provide a robust estimation for every part of the building. This offline database is given by

$$\varsigma = (\ldots, \varsigma_{\text{fp,ap},n}, \ldots),\tag{2.91}$$

containing n consecutive measurements, at a location denoted by fp, for the access point given by ap. The central question is then given by

$$p(\boldsymbol{\varrho} \mid \boldsymbol{s}),\tag{2.92}$$

the probability for the pedestrian to reside at $\boldsymbol{\varrho}$, given the smartphone currently measures the signal strengths $\boldsymbol{s}$. However, as only the signal strength's behavior given a certain location is known, determined either by fingerprints or a prediction model, the two operands of (2.92) must be swapped. Both points of view are related by Bayes' rule

$$p(\boldsymbol{\varrho} \mid \boldsymbol{s}) = \frac{p(\boldsymbol{s} \mid \boldsymbol{\varrho})\, p(\boldsymbol{\varrho})}{p(\boldsymbol{s})} \propto p(\boldsymbol{s} \mid \boldsymbol{\varrho}) = p_{\text{wifi}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right)$$
$$\text{assuming } p(\boldsymbol{\varrho}), p(\boldsymbol{s}) = \text{const}, \quad \langle \boldsymbol{q} \rangle_t = \langle (x, y, z, \ldots) \rangle_t, \quad \langle \boldsymbol{o} \rangle_t = \langle (\boldsymbol{s}, \ldots) \rangle_t.\tag{2.93}$$

When omitting prior knowledge, assuming $p(\boldsymbol{\varrho})$ and $p(\boldsymbol{s})$ to be constant, both viewpoints are proportional. This notation also matches with the evaluation (2.4), introduced in section 2.2.

2.7.4 Location Estimation Using Lateration

As mentioned, the simplest Wi-Fi localization approach is based on multilateration (cf. GPS in section 2.3), using the signal strengths of nearby Wi-Fi base stations, received by the smartphone. If an access point's transmission power P_0 and path loss exponent γ are known, (2.86) can be used to convert its measured RSSI into an approximate distance d_{ap} towards it. Similar to GPS satellites, for a transmitter to serve as a localization reference, its position $\boldsymbol{\varrho}_{\text{ap}}$ must also be known. The connection between the receiver's real whereabouts $\tilde{\boldsymbol{\varrho}}$, potential whereabouts $\hat{\boldsymbol{\varrho}}$, and the measured signal strength s_{ap}, converted to a distance d_{ap}, is given by

$$\| \overbrace{\boldsymbol{\varrho}_{\text{ap}}}^{\text{known}} - \overbrace{\tilde{\boldsymbol{\varrho}}}^{\text{unknown}} \| \approx \overbrace{d_{\text{ap}}}^{\text{measured}} \overset{!}{=} \overbrace{\|\boldsymbol{\varrho}_{\text{ap}} - \hat{\boldsymbol{\varrho}}\|}^{\text{calculated}}, \quad d_{\text{ap}} = d_{\text{LD}}\left(s_{\text{ap}}\right)_{(2.86)}, \quad s_{\text{ap}} \mathrel{\widehat{=}} P.\tag{2.94}$$

When the phone receives a decent number of transmitters, its estimated location $\boldsymbol{\varrho}^*$ is the one that minimizes the differences between measured and actual distances towards each transmitter

$$\boldsymbol{\varrho}^* = \arg\min_{\hat{\boldsymbol{\varrho}}} \sum_{\text{ap}} \left(\|\boldsymbol{\varrho}_{\text{ap}} - \hat{\boldsymbol{\varrho}}\| - d_{\text{ap}} \right)^2 . \tag{2.95}$$

As (2.12) for the GPS, (2.95) can be estimated using numerical optimization, or by solving a linearized version of the equation [Pow62; Li+05; DH10]. Depending on the chosen approach, at least four measurements are required, to estimate 3D whereabouts. The result is a single location the pedestrian might currently reside at. Similar to (2.17), as nearby locations are likely as well, the evaluation is given by the distance of a potential state $\boldsymbol{q}_t$ from $\boldsymbol{\varrho}^*$

$$p_{\text{wifiLatD}} \left(\boldsymbol{o}_t \mid \boldsymbol{q}_t \right) = \mathcal{N} \left(d \mid 0, \sigma_{\text{ld}}^2 \right) , \quad d = \|\boldsymbol{\varrho}^* - \mathbf{pos}_{\text{xyz}} \left(\boldsymbol{q}_t \right)\| . \tag{2.96}$$

Exemplary measurements and corresponding location results are shown in figure 2.25. As can be seen, there might be real-world constellations, where outliers distort the localization result. While the two leftmost results provide a viable localization for the given reference points and distances, the two rightmost might differ from expectations. In 2.25c, four crossings can be observed, not denoting a clear solution, but (2.96) estimates the result to reside directly within the center. In 2.25d, a single outlier affects the ideal intersection between the three other reference points, shifting the estimation (2.96) away from this intersection.

In contrast to common GPS receivers, where only the result $\boldsymbol{\varrho}^*$ is disclosed by the hardware, for the described Wi-Fi scenario, all individual distances from the transmitters are known. By assigning an uncertainty to every single distance estimation, and assuming statistical independence between the measurements, they can be combined into a *mixture distribution*, which is able to address aforementioned issues, by introducing *multimodalities*

$$p_{\text{wifiLatC}} \left(\boldsymbol{o}_t \mid \boldsymbol{q}_t \right) = \prod_{\text{ap}} \mathcal{N} \left(\left\| \boldsymbol{\varrho}_{\text{ap}} - \mathbf{pos}_{\text{xyz}} \left(\boldsymbol{q}_t \right) \right\| \mid d_{\text{ap}}, \sigma_{\text{lc}}^2 \right) , \quad d_{\text{ap}} = d_{\text{LD}} \left(s_{\text{ap}} \right)_{(2.86)} . \tag{2.97}$$

Instead of estimating a single location, and applying an uncertainty hereafter, (2.97) directly includes every distance within its own uncertainty. The most likely whereabouts are given by the location where all probabilities join together. While (2.97) uses a normal distribution to model these uncertainties, other distributions can be applied as well, such as the one discussed in section 2.3. For visualization reasons, figure 2.26 depicts the evaluation results, when using an exponential distribution. As shown, size and shape of the likely area (dark regions) directly depend on the quality of the calculated distances. In case of accurate observations (figure 2.26a), the probable region is rather small. For poor distance estimations, like within 2.26b and 2.26c,
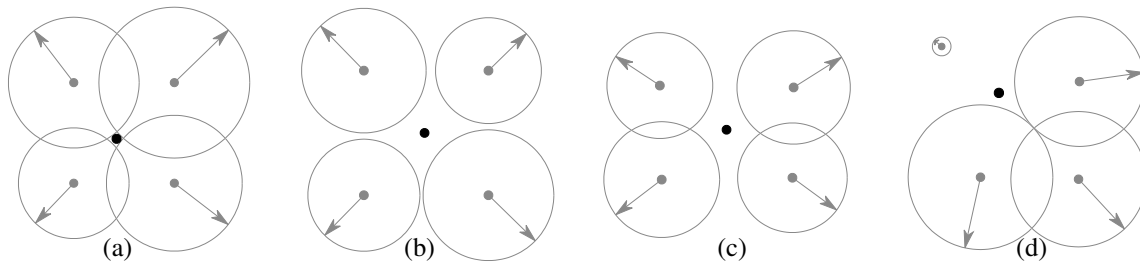
Figure 2.25: Discrete lateration results (black dot) when using (2.95) on four reference points (gray dots) with the measured distances (arrows). While the two leftmost results seem viable, the two rightmost estimations might not be as expected.

the probable area is spreading. In contradictory cases, such as 2.26d, the resulting mixture density adjusts its shape and size to match the distance estimations. While this seems ideal at the first glance, the amount of adjustment in shape and size strongly depends on the chosen distribution and uncertainty σ_{lc}. While larger values allow for joining, when distance estimations are bad, they unnecessarily increase the uncertainty when estimations are good.

One advantage of the discrete approach (2.96) over the continuous (2.97), is quantitative error estimation. While the density (2.97) somewhat adjusts its shape and size, this error can only be visualized, but not directly quantified. (2.95), however, allows for comparing the estimated whereabouts with all measured distances, to infer an error approximation, similar to the one discussed for the GPS. For some use cases, this might be the preferred solution, as the quality of signal strength readings can be quantified, e.g. to suppress potential outliers.

This leads to the question of actual values for σ_{ld} and σ_{lc}. While σ_{ld} can be estimated using aforementioned error quantification, e.g. by using the standard deviation of all differences between measured and actual distance towards (2.95)

$$\sigma_{ld}^2 = \mathbb{E}(\mathcal{X}^2) - (\mathbb{E}(\mathcal{X}))^2 \,, \quad \mathcal{X} = \left\{ \left| \left\| \boldsymbol{\rho}_{ap} - \boldsymbol{\rho}^* \right\| - d_{ap} \right| \mid \forall ap \right\} \,, \tag{2.98}$$

σ_{lc} is a rather empiric choice, based on signal strength variations, discussed in section 2.7.1.

To summarize, lateration enables a simple, computationally efficient localization, able to provide continuous evaluation results. However, localization quality suffers from the simplicity of the underlying signal strength model (2.86), where walls and other obstacles can not be included. Furthermore, to convert measured RSSIs to distances using (2.86), the position $\boldsymbol{\rho}$, transmission power P_0 and path loss exponent γ of all transmitters, must be known. While P_0 and γ can be chosen empirically, the position is crucial, but might be unavailable within some buildings, e.g. due to non-disclosure agreements or data privacy issues. Furthermore, measuring and recording the exact position of all transmitters is a time consuming task. Chintalapudi et al. [CPP10] therefore suggested a big data approach, where all three parameters are assumed
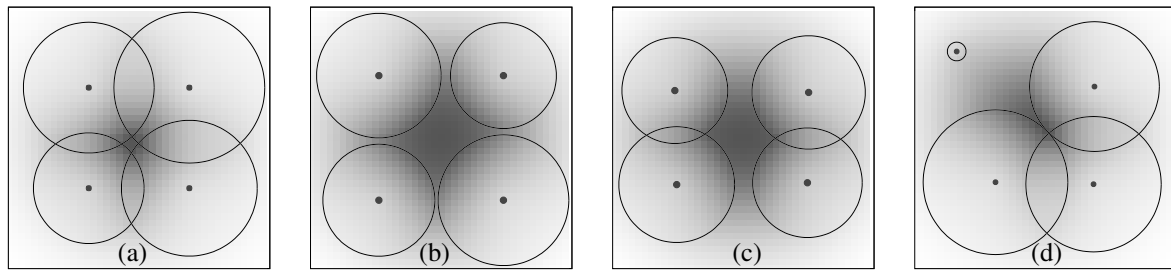
Figure 2.26: Continuous lateration results when using (2.97), but with an exponential distribution, on four reference points (gray) with given distances (circles' radius). The intensity of the background denotes the likelihood to reside at a certain location, where dark areas are more likely.

unknown. Based on signal strengths recorded by pedestrians with smartphones walking through the building, they optimize a model, containing not only each transmitter's parameters ϱ, P_0 and γ, but also the unknown pedestrian locations ϱ at the time of each measurement. To estimate absolute positions with respect to the building, they include information provided by a few GPS fixes, while the pedestrians walked outdoors. If GPS fixes are unavailable, other landmarks, such as stairs or elevators, can be detected, using additional sensors (see section 2.6), and compared against the building's floorplan, providing the same absolute mapping [AY12]. While not requiring prior parameter knowledge, this approach demands for a large dataset of pedestrian measurements, throughout all parts of the building, not easily available. Similar and alternative evaluation and optimization strategies thus are the topic of the following sections.

2.7.5   Location Estimation Using Fingerprints

Depending on the building's architecture, and the multitude of environmental influences discussed in section 2.7.1, simple signal strength prediction models will often be inaccurate. A straightforward, yet time consuming, solution to this problem is given by conducting actual real-world measurements at known locations throughout the whole building, capturing the realities of the signal strength's behavior. Due to architectural influences and other effects, RSSI readings will vary greatly throughout the building, and sometimes even within several meters. This uniqueness is similar to the one of human fingerprints, therefore in literature often referred to as *fingerprinting* [Li+05; CPP10; Lui+11; Pal+11]. Mentioned earlier, it was first introduced by Bahl and Padmanabhan [BP00], even though they named it *offline phase*. Their installation contained three transmitters within an $980 \, \mathrm{m}^2$ office floor, with 70 fingerprints, recorded solely along the floor's hallway, with a varying distance of approximately $2 \, \mathrm{m}$. To achieve a stable estimation, the average of 20 consecutive readings was stored. After recording this database, they performed several walks using the same receiving hardware. Current RSSI measurements were compared with the database, using the Euclidean distance between the three currently received
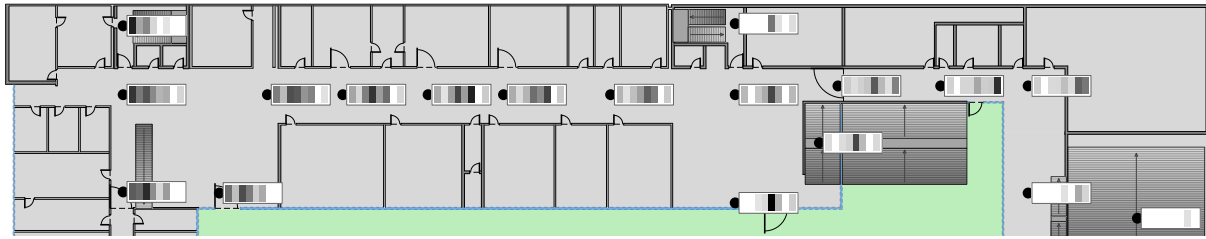
Figure 2.27: Real-world fingerprint example for an area of $110 \times 21\,\mathrm{m}$ and eight nearby access points. At each location, the average RSSI for every of the eight APs is denoted by a bar, where the highest signal strength is black. If a bar is missing (white), the transmitter could not be received at this location.

RSSIs and the three recorded for each fingerprint. The pedestrian's most likely whereabouts are given by the fingerprint with the smallest indicated distance in this *signal space*. Their approach yielded a median distance error of $2.94\,\mathrm{m}$ along all measurements, indicating that Wi-Fi signal strength and fingerprinting is a viable solution for absolute indoor localization.

For an impression of discriminability, figure 2.27 depicts a 110×21 m section of a real-world scenario with eight access points installed along the floor. The eight bars beside each fingerprint denote the RSSI for every of the eight transmitters, where black describes the strongest signal. Even for directly adjacent fingerprints, some variation can be observed. This effect will intensify with the number of installed transmitters, as each is affected by different environmental influences, increasing the chance for separability. Missing bars indicate that the access point's signal was too weak to be detected. Only for one in 18 fingerprints, all transmitters were visible. This represents a crucial aspect for the comparison between the smartphone's current RSSI readings and the fingerprint database. Test arrangements often focus on a small area, where every transmitter can be received at each potential location. For real-world scenarios, like large buildings with multiple floors, this assumption won't hold [BP00]. When the phone receives a transmitter, that is unknown to the database, e.g. due to newly installed hardware or temporal Wi-Fi hot spots, effects are similar. While both cases can be addressed by ignoring the respective entries, doing so discards available information, or might render entries incomparable. If a transmitter was recorded with a weak signal during the time of fingerprinting, it is likely for the smartphone to not receive this transmitter, depending on the sensitivity of the internal Wi-Fi component (see section 2.7.1). If the signal was strong while fingerprinting, it is unlikely for the phone to miss this transmitter. Cheng et al. [Che+05] even suggested using the amount of invisibility of a transmitter as metric, as it strongly depends on the distance from it. This, however, requires several consecutive measurements to provide viable results, adding delays to the localization process. Roos et al. [Roo+02] replace missing entries with a weak dummy signal strength, as the transmitter might just be temporarily unavailable or was recently removed. Yet, without additional prior knowledge, there is no ideal answer to the question of

how to handle missing RSSI readings. Furthermore, it strongly depends on how the actual comparison between current readings from the phone and offline database is conducted, to infer the most likely whereabouts.

Comparison with Euclidean Distances   The simplest approach to determine the best matching fingerprint compares the difference between live and offline RSSI readings by calculating the distance between both, e.g. via the Euclidean or Manhattan distance metric. Intended for geometrical comparisons, it works reasonably well for comparisons in signal space [BP00]. While the phone's current RSSI readings s contain just a single signal strength per nearby transmitter, the fingerprint database contains multiple measurements, to accurately capture the signal's behavior at every location. For comparison, all offline measurements for one transmitter must be combined into a single one. The most common approach uses the average of all consecutive measurements for one transmitter and location [YA05; YMA07]. The comparison between current smartphone readings s and the offline database ς at the fingerprint fp is given by

$$\Delta_{\text{euclid}}\left(s, \varsigma, \text{fp}\right) = \sqrt{\sum_{\text{ap}}\left(s_{\text{ap}} - \mathbb{E}(\varsigma_{\text{fp,ap}})\right)^2}, \tag{2.99}$$

comparing the RSSI s_{ap} of each access point received by the smartphone with its corresponding average from the database

$$\mathbb{E}(\varsigma_{\text{fp,ap}}) = \frac{1}{|\varsigma_{\text{fp,ap}}|}\sum_{n}\varsigma_{\text{fp,ap},n}. \tag{2.100}$$

The number of access points contained within ς_{fp} varies depending on the fingerprint fp, as two fingerprints will probably contain a different number of visible APs (cf. figure 2.27). For fingerprints to be comparable using (2.99), the same number of transmitters must be considered, independent of the fingerprint. All access points present within s, but missing within ς, can safely be omitted, as the offline database has no knowledge about them. Vice versa, if one transmitter is not present within the smartphone readings s, but visible at some fingerprint, this is viable information, and must not be ignored. To ensure that the number of comparisons is the same for every fingerprint, all comparisons consider the entirety of all transmitters known to the offline database. Missing entries, s_{ap} or $\varsigma_{\text{fp,ap}}$, are replaced by a constant. As this value models a missing/invisible transmitter, it can e.g. be chosen to be near the receivers sensitivity, approximately around $\approx -90\,\text{dBm}$ [Roo+02]. Hereafter, all fingerprints, and the smartphone's current readings, contain the same number of entries, equal to the total number of transmitters known to the database, allowing for a valid comparison. The pedestrian's current whereabouts

are then given by the fingerprint that minimizes the distance to the phone's current readings

$$\text{fp}^* = \arg \min_{\text{fp}} \Delta_{\text{euclid}} \left(\boldsymbol{s}, \varsigma, \text{fp} \right) . \tag{2.101}$$

Due to its simplicity, (2.99) suffers from an important drawback. By incorporating fingerprints using just the average of several long-term readings, information on the signal's actual behavior is discarded. This is addressed by using a probabilistic comparison instead.

Comparison with Probability Distributions Depending on architectural surroundings, the RSSI of a transmitter behaves differently throughout a building. Due to multipath propagation, there will often be more than one way for a signal to reach a location. This not only causes its average strength to be different between locations, but also its variance. Depending on paths and architectural influences along them, the variance ranges from small to pronounced. Using all consecutive scans at every location from the offline fingerprint database, the variance can be determined alongside the mean, afterwards allowing for a more profound comparison with the smartphone's readings. Youssef et al. [YA05; YAA05] suggest using the fingerprint measurements to estimate a normal distribution for every access point and location. These are then used to calculate the probability for each of the phone's current readings to match

$$p(s_{\text{ap}} \mid \varsigma_{\text{fp,ap}}) = \mathcal{N} \left(s_{\text{ap}} \mid \mu, \sigma^2 \right)$$
$$\text{with} \quad \mu = \mathbb{E}(\mathcal{X}) , \quad \sigma^2 = \mathbb{E}(\mathcal{X}^2) - (\mathbb{E}(\mathcal{X}))^2 , \quad \mathcal{X} = \{\varsigma_{\text{fp,ap},n} \in \varsigma_{\text{fp,ap}}\} . \tag{2.102}$$

Assuming statistical independence of all access points installed within a building, a fingerprint's overall likelihood is given by the product of the individual probabilities from (2.102)

$$p(\boldsymbol{s} \mid \varsigma, \text{fp}) = \prod_{\text{ap}} p(s_{\text{ap}} \mid \varsigma_{\text{fp,ap}}) . \tag{2.103}$$

When using probabilities for comparing live RSSI readings against fingerprints, transmitters unseen by the phone can be handled in similar ways as described earlier. They can either be replaced with a small constant probability or by a constant, weak RSSI, that is hereafter applied to (2.102), or to a distribution around the weak constant, if the transmitter is unknown to the current fingerprint. While the latter seems to be the most profound solution, as it directly yields the likelihood of measuring a weak signal, the actual value can not be quantified directly.

While using a normal distribution instead of the mean value yields improvements, it is still not ideal. Due to multipath propagation, the mean value might not be the one with the highest probability, and there can be situations where a transmitter's long-term signal for some location denotes multiple peaks, as it arrives via more than one path. In those cases, the signal strength's behavior is non Gaussian, requiring for approaches that support multimodalities.
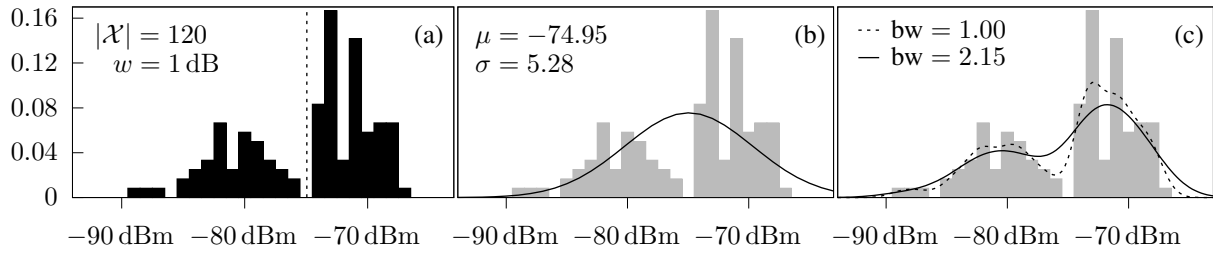
Figure 2.28: Strategies to approximate the signal strength distribution for one access point and location. A histogram of all readings (120) using a bin size $w = 1\,\mathrm{dB}$ (a) with the mean (dashed) for comparison, a normal distribution estimated around this mean (b), and a KDE for two different bandwidths (c), where the dashed bw $= 1.00$ is an empiric choice, and bw $= 2.15$ was determined using (2.108).

Comparison with Histograms   One way to model arbitrary distributions of measurements is the histogram [Pea95; Smi99; YAS03; YA05]. All RSSI readings for one access point at one location from the offline phase are grouped into bins, where each bin denotes the number of measurements contributing to it. After normalizing the sum of all bins to one, each bin directly denotes the likelihood of measuring a signal strength within the range covered by it

$$p(s_{\mathrm{ap}} \mid \varsigma_{\mathrm{fp,ap}}) = \frac{1}{|\varsigma_{\mathrm{fp,ap}}|} \sum_n \begin{cases} 1 & \mathrm{bin}(s_{\mathrm{ap}}) = \mathrm{bin}(\varsigma_{\mathrm{fp,ap},n}) \\ 0 & \mathrm{else} \end{cases}, \quad \mathrm{bin}(x) = \left\lfloor \frac{x}{w} \right\rfloor . \tag{2.104}$$

The bin size w can be chosen freely, but is hard to determine correctly. Using $w = 1\,\mathrm{dBm}$ per bin directly resembles RSSI readings, usually provided as integers by the underlying hardware. This, however, might yield single bins with a count of zero, causing gaps within the histogram (see figure 2.28). Meng et al. [Men+11] therefore suggest using a $w = 2\,\mathrm{dBm}$ to address this problem and simultaneously halve the amount of memory required for storing the histogram. Using larger bins decreases the overall quality, as similar RSSI readings might be treated with the same likelihood. This problem can be addressed by interpolating between adjacent bins, using a weighted result, creating a more continuous output [McC86; DT05].

Besides gaps, the edge-areas of the histogram are difficult to handle as well, as every RSSI not encountered during the offline phase is assigned a probability of zero. This is correct from the histogram's point of view, but does not resemble real-world conditions, where the likelihood is not zero but infinitesimally small. When (2.104) is used within (2.103), a single zero makes the whole fingerprint unlikely. Roos et al. address this issue by pre-assigning a very small constant likelihood to every bin contained within the histogram [Roo+02]. While this mitigates the problem, it is still incorrect, as the probability is not constant, but varying depending on the distance towards the nearest and actually covered bin. Both of the discussed issues can be targeted by *smoothing* the histogram, e.g. by using a kernel density estimation (KDE).

Comparison with Kernel Density Estimation   For a continuous result, the histogram's discrete binning (2.104) is replaced by a continuous function, referred to as *kernel*, e.g. being some narrow distribution, shaped like a smoothed bin. The idea is the same as for aforementioned lateration approaches, including every measured distance by its probability. Therefore, all individual probabilities, given by comparing all offline readings for one location and transmitter with the corresponding one from the smartphone, are summed up, and normalized

$$p(s_{\mathrm{ap}} \mid \varsigma_{\mathrm{fp,ap}}) = \frac{1}{|\varsigma_{\mathrm{fp,ap}}|\mathrm{bw}} \sum_n K\left(\frac{s_{\mathrm{ap}} - \varsigma_{\mathrm{fp,ap},n}}{\mathrm{bw}}\right) . \tag{2.105}$$

While most use cases refer to a Gaussian kernel

$$K_{\mathrm{Gauss}}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} = \mathcal{N}\left(x \mid 0, 1\right) , \tag{2.106}$$

differently shaped functions can be used as well [Roo+02; GCC12]. The overall process is often referred to as kernel density estimation (KDE) or Parzen/Rosenblatt estimation [Par62; Ros56]. For Gaussian kernels, (2.105) can be rewritten, emphasizing its essence

$$p(s_{\mathrm{ap}} \mid \varsigma_{\mathrm{fp,ap}}) = \frac{1}{|\varsigma_{\mathrm{fp,ap}}|} \sum_n \mathcal{N}\left(s_{\mathrm{ap}} \mid \varsigma_{\mathrm{fp,ap},n}, \sigma^2\right) , \quad \sigma = \mathrm{bw} . \tag{2.107}$$

By using the KDE, gaps within the histogram are closed, and multimodalities are preserved. Yet, the result strongly depends on the chosen bw, referred to as *bandwidth*. While small values preserve tiny variations, gaps will remain open. On the other hand, larger values create smoother results, but suppress minor variations. Literature covers a variety of metrics to estimate the best bandwidth, to minimize the error between the KDE and the unknown density function [SS09; BN09]. For many use cases, the *rule-of-thumb* described by Silverman [Sil86] provides viable results, whenever a Gaussian kernel is used. Here, the bandwidth depends on the number and standard deviation of all signal strengths recorded for one access point and location

$$\mathrm{bw}(\varsigma_{\mathrm{fp,ap}}) = \left(\frac{4\sigma^5}{3|\mathcal{X}|}\right)^{\frac{1}{5}}, \quad \sigma = \sqrt{\mathbb{E}(\mathcal{X}^2) - (\mathbb{E}(\mathcal{X}))^2}, \quad \mathcal{X} = \{\varsigma_{\mathrm{fp,ap},n} \in \varsigma_{\mathrm{fp,ap}}\} . \tag{2.108}$$

Figure 2.28 depicts a comparison of all discussed techniques using 120 real-world readings for one transmitter at one location. As can be seen within the histogram 2.28a, the measured RSSIs are split into two large groups. The average is exactly between both of them, indicating major drawbacks for the simple Euclidean distance approach. The same holds true for the normal distribution 2.28b, relying on the same mean value. Using the histogram itself is slightly better, yet there are major gaps throughout the recorded data, yielding near-zero probabilities for some RSSIs. Both issues are addressed by the KDE in 2.28c, where gaps are closed, but still included.

Location Estimation   Previously discussed techniques yield a distance/probability for every fingerprint within the building, based on the current phone RSSIs $\boldsymbol{s}$. The pedestrian's most likely whereabouts are given by the position of the fingerprint which matches best. Similar to lateration described in section 2.7.4, this yields a single, discrete location, even if nearby locations are just as likely. When two, or more, fingerprints are similar, they can easily be confused with each other, independent of the chosen metric, potentially causing large localization errors and jumps [Liu+12]. Just like with lateration, this can be addressed by letting every single fingerprint contribute to the solution, instead of solely focusing on the best one.

Bahl and Padmanabhan [BP00] suggested using the average of the positions of the k nearest fingerprints. While this provides a more continuous result, all k fingerprints are treated equally, even though their Euclidean distance might vary. While the resulting location is not bound to fingerprint positions, it is still a discrete location. Therefore, Youssef et al. [YA05] focused on discussed probability metrics, to hereafter estimate the regional *weighted* average position among the k most likely fingerprints, weighted by their probabilities. While including weights yields more viable results, the output is still a single, discrete position.

As each fingerprint's probability represents a discrete, weighted sample, like a bin within the histogram, the KDE can be used to estimate the unknown continuous density, by combining adjacent fingerprints [TBF05]. In contrast to the KDE applied to signal strengths, a three dimensional kernel is required for estimating a position within the building. However, when assuming the same bandwidth bw for all three dimensions, the kernel can be reduced to a single dimension, using the distance d between each fingerprint's three dimensional position $\mathbf{pos}_{\mathrm{xyz}}\left(\mathrm{fp}\right)$ and a potential location $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right)$ within the building. The evaluation of the pedestrian's current RSSI readings depending on this location is then given by

$$p_{\mathrm{wifiKDE}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \sum_{\mathrm{fp}} \frac{\overbrace{\mathcal{N}\left(d \mid 0, \sigma^2\right)}^{\text{distance kernel}} \overbrace{p(\boldsymbol{s} \mid \boldsymbol{\varsigma}, \mathrm{fp})_{(2.103)}}^{\text{fingerprint probability}}}{\sum_{\mathrm{fp}}\left(1 + p(\boldsymbol{s} \mid \boldsymbol{\varsigma}, \mathrm{fp})_{(2.103)}\right)} \tag{2.109}$$

$$\text{with} \quad d = \left\|\mathbf{pos}_{\mathrm{xyz}}\left(\mathrm{fp}\right) - \mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right)\right\|, \quad \sigma = \mathrm{bw},$$

where the denominator is required for normalization. (2.109) provides viable results if the bandwidth bw is determined correctly, and the building is covered by a decent number of fingerprints. However, this approach relies on multiple nested loops. One for all fingerprints, one for all access points for each fingerprint (2.103), and, dependent on the chosen comparison strategy, all consecutive measurements for every fingerprint and access point (2.104) or (2.105). This yields $O(n^3)$ as worst case scenario for every single location evaluation, potentially causing issues for embedded use when multiple locations are evaluated, and many fingerprints must be examined.

2.7.6 Location Estimation Using Propagation Models

The approaches from section 2.7.4 and 2.7.5 come with different benefits and drawbacks.

Lateration solely requires the positions of all access points installed within the buildings and some empirically chosen parameters for P_0 and γ. Yet, it only supports simple invertible signal strength prediction models, thus tendentially providing poor results, especially under the influence of heavily attenuating concrete walls, which can not be included individually.

Fingerprinting addresses this problem via real-world signal strength measurements. However, the time needed for the initial setup, creating hundreds of fingerprints, each with several consecutive measurements, is tremendous. Recommendations for those repetitions range between 20 [BP00] and 1000 [Men+11], being almost unfeasible for large buildings like airports, or other multi-level architecture. Palaniappan et al. [Pal+11] therefore suggest using robots to perform required measurements automatically. While this can reduce setup/maintenance time and costs, necessary efforts are still notable. Furthermore, fingerprints are discrete and require for costly computations, such as the kernel density estimation, to infer continuous results. While techniques to perform rapid KDE approximations exist [Bul+18], required computational power is still a concern for recent smartphone CPUs/GPUs, especially when considering the third dimension as well. Also, fingerprinting needs special treatment of invisible transmitters for valid comparisons with smartphone readings. Thus, neither of the two approaches represents an ideal choice for an indoor localization system that is accurate, fast to set up and easy to maintain.

A compromise between both is utilizing advanced signal strength prediction models. As they usually are not invertible, they can not be used to convert a phone's measured RSSI into a distance estimation, like performed in section 2.7.4. Instead, the models are used to estimate each transmitter's signal strength, that is, predicting what fingerprints would look like [ElK+10]. Thus, the strategies presented in section 2.7.5 are applicable, comparing smartphone readings with model predictions, instead of fingerprints. However, in contrast to the latter, model predictions are calculable for every single location within a building. Thus, transmitters are never invisible to the model, and this approach is completely continuous. This reduces computational costs, not requiring any interpolation steps, such as the KDE. To use this strategy, the parameters required by a chosen model, like γ and P_0, must be determined in some way. Depending on available prior knowledge, like known transmitter locations, and the chosen model, these parameters can either be determined empirically, or must be estimated. This can e.g. be performed by training, based on a few reference measurements. While these measurements are similar to fingerprinting, they are much more sparse, and thus significantly faster to set up and maintain. For this approach, setup and maintenance times are reduced at the cost of accuracy, depending on the chosen model and reference measurements [Ebn+14; Ebn+15; Ebn+17].

The following discussions focus on comparing RSSIs received by the phone with model predictions, and how to determine necessary model parameters. They are provided in a general form, independent of a chosen model. That is, they apply to all signal strength prediction models, including the three representatives (log-distance model, extended log-distance model, ray tracing) discussed in section 2.7.2. Implementation examples are provided afterwards.

Comparing RSSIs With Model Predictions   In the following, a model's signal strength prediction for a location $\boldsymbol{\rho}$ and access point ap is referred to as $P_{\mathrm{mdl}}\left(\boldsymbol{\rho}, \psi_{\mathrm{ap}}\right)$. It is based on the parameters ψ_{ap} required for a transmitter, containing e.g. its location (cf. section 2.7.2). The probability for residing at $\boldsymbol{\rho}$ is then given by comparing s_{ap} measured by the phone with the model's prediction (cf. section 2.7.5). For example, by using a probability density function (PDF), including the model's uncertainty, and the signal's multipath behavior at this location

$$
p(s_{\mathrm{ap}} \mid \boldsymbol{\rho}, \psi_{\mathrm{ap}}) = \underbrace{p\Big(s_{\mathrm{ap}} \mid \overbrace{P_{\mathrm{mdl}}\left(\boldsymbol{\rho}, \psi_{\mathrm{ap}}\right)}^{\text{model prediction}}, \overbrace{\dots}^{\text{PDF dep.}}\Big)}_{\text{PDF for model/multipath uncertainty}}, \quad \underbrace{\boldsymbol{\rho}_{\mathrm{ap}}, \overbrace{\dots}^{\text{model dep.}} \in \psi_{\mathrm{ap}}}_{\text{model parameters for one AP}}. \tag{2.110}
$$

Type, shape and additional parameters of the PDF strongly depend on the chosen model. The log-distance model, and its extended version, predict the *mean* signal strength given one location, with the uncertainty being a zero mean random variable $\mathcal{X} \sim \mathcal{N}(0, \sigma_{\mathrm{rssi}}^2)$ (cf. section 2.7.2). Here, the PDF from (2.110) can be chosen to be a normal distribution. Its σ_{rssi} is either determined empirically, or estimated, e.g. by comparing the model predictions with reference measurements. However, for more advanced prediction models, supporting multipath propagation, such as ray tracing, more complex PDFs are required, including support for multimodalities [Ebn+14]. Every single ray reaching the location in question could e.g. be added to a histogram, hereafter applying a KDE to estimate the signal's unknown distribution for this location. Again, this represents a tradeoff between quality, and memory/computational complexity.

Assuming statistical independence of all access points, the evaluation of the phone's current RSSI values $\boldsymbol{s}$ against a state $\boldsymbol{q}_t$ is given by the product of the individual probabilities (2.110)

$$
p_{\mathrm{wifiMdl}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right) = \prod_{\mathrm{ap}} p\left(s_{\mathrm{ap}} \mid \mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right), \psi_{\mathrm{ap}}\right)_{(2.110)}
$$
$$
\langle \boldsymbol{q} \rangle_t = \langle (x, y, z, \dots) \rangle_t, \quad \langle \boldsymbol{o} \rangle_t = \langle (\boldsymbol{s}, \dots) \rangle_t. \tag{2.111}
$$

As mentioned, compared to fingerprinting, prediction models are always able to provide a signal strength estimation, even if the resulting value is far below the sensitivity of the receiving hardware. This reduces the problem of invisible transmitters solely to the smartphone side, where unknown entries can simply be ignored, using only transmitters that are both, currently seen by the phone and known to the model [Ebn+14].

Model Parameter Estimation   To calculate $P_{\mathrm{mdl}}\left(\boldsymbol{\varrho}, \boldsymbol{\psi}_{\mathrm{ap}}\right)$ in (2.110), the parameters $\boldsymbol{\psi}_{\mathrm{ap}}$, for a certain model and transmitter, must be known. Shown earlier, most models require the transmitter's location $\boldsymbol{\varrho}_{\mathrm{ap}}$, to infer the distance towards it, and to perform intersection tests. Besides, they e.g. require values for the signal's attenuation within free space, or for encountered obstacles, discussed in section 2.7.2. The following discussions examine potential estimation strategies.

For simple models it is possible to use the positions of all transmitters installed within a building, and using empirical choices for the remaining parameters [Ebn+14; Ebn+15]. However, for larger public buildings it can be hard to gain access to documents containing these positions, if available. Also, physical access to the transmitting hardware can be prohibited, or the number of installed transmitters is too large for this approach to be worthwhile [Tor+17].

Alternatively, by conducting several reference measurements at known locations throughout the walkable area, numerical optimization can be used to estimate all required parameters. Goal of the optimization is to determine $\boldsymbol{\psi}_{\mathrm{ap}}$ (e.g. $\boldsymbol{\varrho}$, P_0, γ or ϕ), so that the model's predictions hereafter match with real-world signal strength behavior. Being similar to recording a few fingerprints, reference measurements are also referred to as fp, stored within the database ς.

When referring to the extended log-distance model, every transmitter has its own location $\boldsymbol{\varrho}_{\mathrm{ap}}$ and transmission power $P_{0\mathrm{ap}}$. In theory, the remaining parameters γ and ϕ, are identical among all transmitters, and could thus be optimized together. However, this model represents a vague approximation of the signal's real behavior, omitting many physical effects. Estimating parameters *per transmitter* instead of globally will thus often increase its prediction quality [Ebn+17]. For more realistic models, such as ray tracing, a global parameter estimation can be suitable, as it increases the amount of available training data, and allows for a more robust estimation, e.g. of the individual attenuation factors ϕ, also reducing the risk of overfitting. With the required equations being marginally different, the following focuses on the *per transmitter* optimization only, without loss of generality.

Independent of the chosen model, its predictions must match real-world conditions as closely as possible. This is achieved by comparing predictions with conducted reference measurements, determining their *difference*. The to-be-optimized target function for a single transmitter's model parameters $\boldsymbol{\psi}_{\mathrm{ap}}$ is defined as the sum of squared errors between each reference measurement at position $\mathbf{pos}_{\mathrm{xyz}}\left(\mathrm{fp}\right)$ for this transmitter, and the corresponding model prediction

$$
\varepsilon_{\mathrm{ap}}\left(\varsigma, \mathrm{ap}, \boldsymbol{\psi}_{\mathrm{ap}}\right) = \sum_{\mathrm{fp}} \overbrace{\sum_{n}}^{\text{repetitions}} \left(\overbrace{\varsigma_{\mathrm{fp,ap},n}}^{\text{measured}} - \overbrace{P_{\mathrm{mdl}}\left(\mathbf{pos}_{\mathrm{xyz}}\left(\mathrm{fp}\right), \boldsymbol{\psi}_{\mathrm{ap}}\right)}^{\text{model prediction, e.g. (2.87)}} \right)^2 . \tag{2.112}
$$

As can be verified, this error should be as small as possible. For a quantification of resulting values, it makes sense to also determine the Root Mean Square Error (RMSE), dividing (2.112)

by the total number N of the two nested summations, and extracting the square root

$$\mathrm{RMSE}_{\mathrm{ap}}\left(\varsigma, \mathrm{ap}, \boldsymbol{\psi}_{\mathrm{ap}}\right) = \sqrt{\frac{1}{N}\, \varepsilon_{\mathrm{ap}}\left(\varsigma, \mathrm{ap}, \boldsymbol{\psi}_{\mathrm{ap}}\right)}\,. \tag{2.113}$$

Concerning optimization, there is no difference between (2.112) and (2.113), as the general be-havior remains the same, and actual values are unimportant, as long as the minimum is reached. However, due to requiring less mathematical operations, optimizing (2.112) is faster. Thus, the best model parameters $\boldsymbol{\psi}_{\mathrm{ap}}^{*}$ for one access point are the ones that minimize (2.112) or (2.113)

$$\boldsymbol{\psi}_{\mathrm{ap}}^{*} = \underset{\boldsymbol{\psi}_{\mathrm{ap}}}{\arg\min}\; \varepsilon_{\mathrm{ap}}\left(\varsigma, \mathrm{ap}, \boldsymbol{\psi}_{\mathrm{ap}}\right) = \underset{\boldsymbol{\psi}_{\mathrm{ap}}}{\arg\min}\; \mathrm{RMSE}_{\mathrm{ap}}\left(\varsigma, \mathrm{ap}, \boldsymbol{\psi}_{\mathrm{ap}}\right)\,. \tag{2.114}$$

(2.114) is solved by using the previously mentioned optimization algorithms. Which one per-forms best, strongly depends on the chosen signal strength prediction model. While simple models can denote continuous convex target functions, advanced variants often show discontin-uous behavior, with many local minima. Details are now discussed by using a synthetic example for the log-distance and the extended log-distance model.

Optimizing the Log-Distance Model   When using the log-distance model (2.87) as imple-mentation for $P_{\mathrm{mdl}}\left(\boldsymbol{\rho}_{\bullet}, \boldsymbol{\psi}_{\mathrm{ap}}\right)$, five model parameters $\boldsymbol{\psi}_{\mathrm{ap}}$ must be optimized for every transmitter

$$\boldsymbol{\rho}_{\mathrm{ap}}, P_{0\mathrm{ap}}, \gamma_{\mathrm{ap}} \in \boldsymbol{\psi}_{\mathrm{ap}}\,, \quad \boldsymbol{\rho}_{\mathrm{ap}} = (x, y, z)^{T}\,. \tag{2.115}$$

Figure 2.29 depicts an example setup, optimizing these parameters based on 14 reference mea-surements, shown in the upper left corner. The 14 values were generated by the ray tracing simulation, depicted in figure 2.24. Minimizing (2.113), minimizes the difference between these 14 reference measurements and corresponding model predictions, by adjusting $\boldsymbol{\psi}_{\mathrm{ap}}$. The optimal result $\boldsymbol{\psi}_{\mathrm{ap}}^{*}$ corresponds to the one with the smallest error. Resulting differences in dB, between reference measurements and predictions based on $\boldsymbol{\psi}_{\mathrm{ap}}^{*}$, are shown in the upper right. The floorplan is drawn transparently, to emphasize that the log-distance model is unaware of any obstacles. The RMSE of $0.6\,\mathrm{dB}$ can be verified by squaring each of the 14 differences, sum-ming the squared values, and taking the sum's square root. As can be seen, the error between model predictions and reference measurements is negligible, and the black cross, indicating the estimated access point position $\boldsymbol{\rho}_{\mathrm{ap}}$, almost matches the actual location. The same holds true for $P_{0\mathrm{ap}}$ and γ_{ap}, both being close to values used for generating the reference from figure 2.24. The five plots below denote the change in error (2.113) when using $\boldsymbol{\psi}_{\mathrm{ap}}^{*}$, but modifying one of the five parameters. That is, keeping all values as determined by the optimization process, but e.g. moving the access point left or right. Denoted by the first three plots, changing the transmitter's
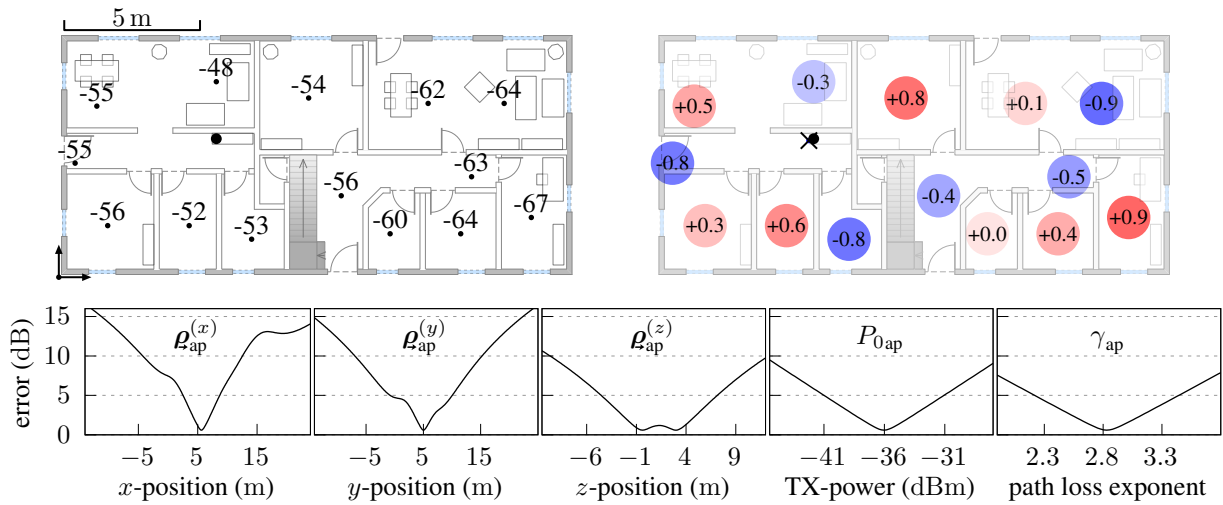
Figure 2.29: Optimizing the log-distance model (2.87), by minimizing (2.113) to match the 14 reference measurements shown in upper left (in dBm), with the two axes denoting $(0, 0)$. Resulting model parameters ψ^*_{ap} were $\rho_{\text{ap}} = (5.6, 5.0, -0.4)$, $P_{0\text{ap}} = -36.0$ and $\gamma_{\text{ap}} = 2.8$, yielding a RMSE of $0.6\,\text{dB}$. All individual errors are depicted in the upper right (in dB), where blue denotes a model prediction lower, and red higher than the reference measurement. The five plots below depict the change in RMSE (2.113) when manually adjusting a single parameter from ψ^*_{ap}.

location can yield non-convex behavior of the error function, with causes shown in figure 2.22. Due to the nonlinear behavior of the log-distance model, individual errors behave in the same way. Depending on the location of each reference measurement, the change in error between prediction and reference is different when moving the transmitter, yielding non-convex behavior. In contrast, changing $P_{0\text{ap}}$ or γ_{ap} denotes a convex output, as they represent an addition and a multiplication, affecting the signal strength prediction model (2.87) linearly.

While results are encouraging, there is a major caveat, as the presented setup referred to a drywall-only environment. Typically, drywalls attenuate radio signals only slightly, approximately matching with the continuous behavior of the log-distance model. Within most buildings, other materials, such as concrete and metallized glass, significantly attenuate radio propagation, causing discontinuous behavior. Repercussions are determined by adding four additional reference measurements, residing behind concrete walls, shown in figure 2.30. The four new reference measurements are strongly attenuated by the concrete, yielding a rapid depletion in signal strength. This behavior does not match with the floorplan-agnostic log-distance model. The impact is shown within the upper right part of the figure, depicting the differences between predictions and reference measurements, where the three leftmost entries clearly deviate from their reference measurements. While the center one ($-3.0\,\text{dB}$) is within the line of sight to the transmitter, the upper ($+3.8\,\text{dB}$) and lower ($+3.6\,\text{dB}$) ones are occluded by concrete. To mitigate this impact, the optimization process chose the path loss exponent to be $\gamma_{\text{ap}} = 8.1$, which
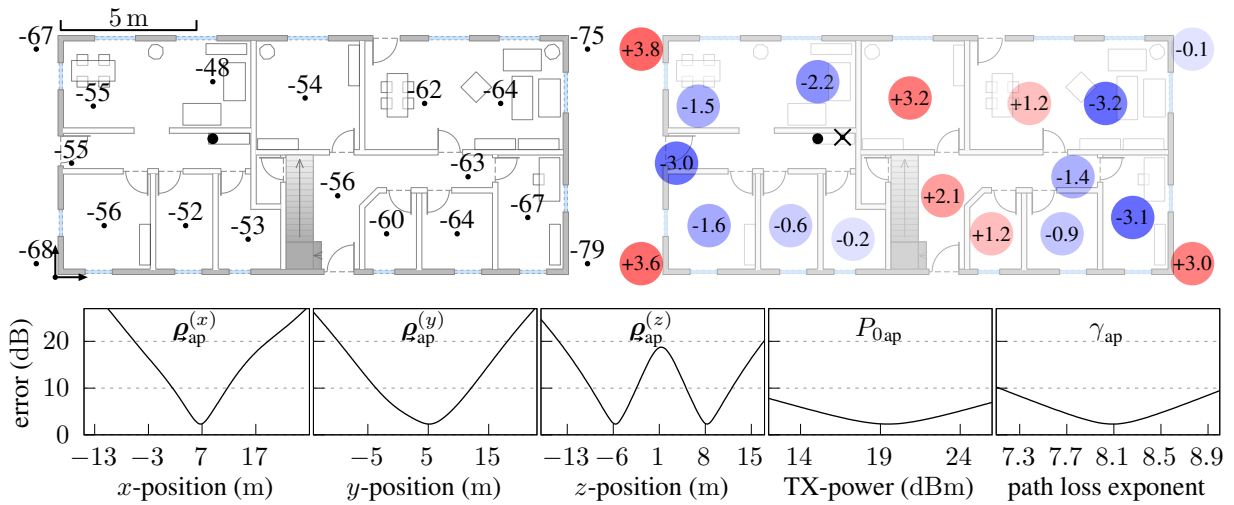
Figure 2.30: Optimizing the log-distance model (2.87), by minimizing (2.113) to match the 18 reference measurements, shown in upper left (in dBm). Resulting model parameters ψ_{ap}^* were $\boldsymbol{\rho}_{\mathrm{ap}} = (6.7, 5.1, -5.7)$, $P_{0\mathrm{ap}} = 19.$ and $\gamma_{\mathrm{ap}} = 8.1$, yielding a RMSE of 2.3 dB. The four new reference measurements cause significant errors, due to the attenuating concrete.

is unnaturally high, and causes a more rapid drop in signal strength with increasing distance. Additionally, the process placed the transmitter $5.7\,\mathrm{m}$ below ground, increasing the distance towards all reference measurements. In doing so, the optimization tries to increase the distance/path loss towards the four exterior reference measurements, to match their lower RSSIs. By increasing the distance towards all reference measurements, the process works within an almost linear range of the log-distance model, explaining why changing the transmitter's x or y now looks more convex than earlier. To compensate for the large γ_{ap} and increased distances, the optimization determined the transmission power to be $P_{0\mathrm{ap}} = 19\,\mathrm{dBm}$, which is numerous times the typical value of $\approx -40\,\mathrm{dBm}$, indicating an overfitting to the given problem. When reference measurements on additional floors above and below are introduced, the situation changes, as the z-coordinate can not be used for compensation. With γ_{ap} being the only remaining parameter, the model will yield unsatisfying results for most multistory buildings [Ebn+17].

Optimizing the Extended Log-Distance Model   When using the extended log-distance model (2.88) for $P_{\mathrm{mdl}}\left(\boldsymbol{\rho}, \boldsymbol{\psi}_{\mathrm{ap}}\right)$, γ_{ap} mainly models free space attenuation. Walls and ceilings along the line of sight are included as separate attenuations based on their material. Within the following, six materials are distinguished

$$\boldsymbol{\phi} = \left(\phi_{\mathrm{ceiling}}, \phi_{\mathrm{concrete}}, \phi_{\mathrm{drywall}}, \phi_{\mathrm{glass}}, \phi_{\mathrm{metalGlass}}, \phi_{\mathrm{door}}\right), \tag{2.116}$$

where ϕ_{door} is only required when doors are considered, and assumed to be closed. The corresponding optimization process estimates the same parameters as for the log-distance model, but

additionally includes the attenuation factors ϕ

$$\boldsymbol{\rho}_{\mathrm{ap}}, P_{0\mathrm{ap}}, \gamma_{\mathrm{ap}}, \boldsymbol{\phi} \in \boldsymbol{\psi}_{\mathrm{ap}}, \tag{2.117}$$

yielding a highly multidimensional optimization problem. As discussed earlier, this model represents a discontinuous function. When the line of sight intersects an obstacle from the floorplan, its attenuation is added, yielding a jump of the predicted signal strength, also affecting the to-be-optimized target function. Previously used optimization algorithms will rarely converge when dealing with discontinuous functions. Yet, brute force approaches might not converge either, due to the "*curse of dimensionality*" [Nie83].

Another option is given by choosing optimization algorithms that are better suited for discontinuous and non-convex problems. Typical representatives are all types of genetic algorithms [Rec73; Sch77; HNG94], where optimization is performed by creating a population of several entities. Every entity is e.g. initialized by drawing a random value for each to-be-optimized model parameter. Hereafter, the whole population is sorted by its *fitness*, e.g. using (2.113), determining the quality of the parameters within each entity. The best entities, with the lowest errors, are randomly combined with each other, e.g. by exchanging, averaging or mutating their values, hopefully using the best parts of both, hereafter replacing unfit entities. Sorting and combining is repeated several times, until a limit is reached. Due to the random approach, the probability of finding a better, global minimum is increased, yet, not guaranteed to converge.

The chance for convergence can be increased by a few adjustments. For the discussed target function/model, the range of the individual parameters is approximately known beforehand. This knowledge can be included during the process, by randomly creating the initial population based on the provided min/max range for every parameter of $\boldsymbol{\psi}_{\mathrm{ap}}$. The access points can e.g. be assumed to reside somewhere within the building's bounding box. Similarly, typical values for P_0, γ and the attenuations ϕ were presented in section 2.7.2. After sorting the population by its fitness (2.114), some % of best entries are kept as they are, in order to not lose good results. The remaining population is replaced by randomly combining fit entities, and/or creating a copy of a random fit entity, with slightly mutated values. In order to find a stable minimum, the amount allowed for these changes starts with a fraction of the known parameter range, and is further reduced over time, inspired by *cooling* from simulated annealing [KGV83]. Using such random adjustments of promising starting values needs some time to converge, but often yields viable results for hard optimization problems, such as the presented one [Ebn+17]. The same strategy can be applied to other complex signal strength prediction models, like ray tracing.

Figure 2.31 depicts the result of using (2.88) in (2.113), with the reference measurements from figure 2.30. The RMSE of $0.3\,\mathrm{dB}$ denotes a viable estimation, confirmed by the 18 indi-
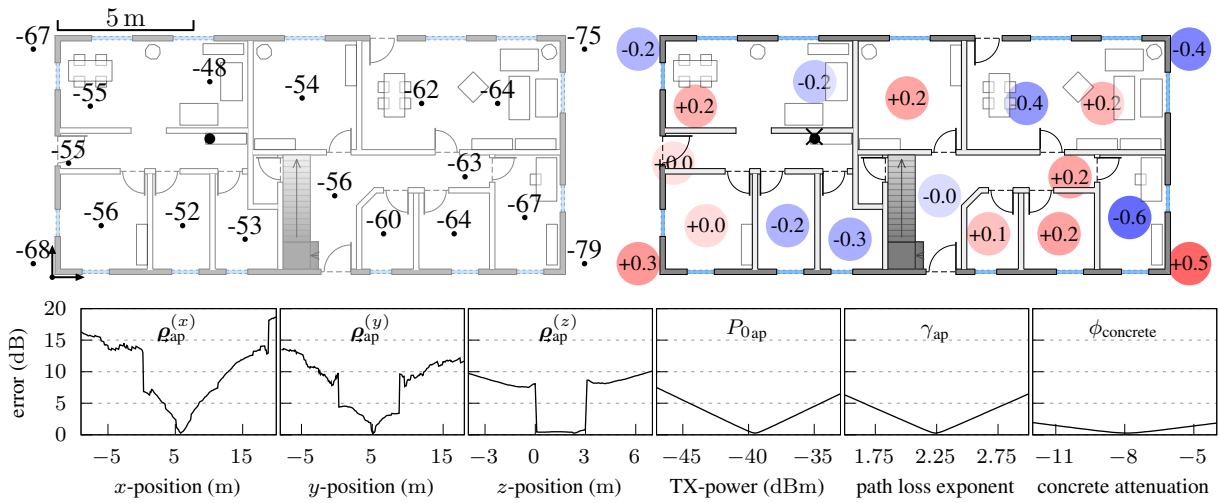
Figure 2.31: Optimizing the extended log-distance model (2.88), by minimizing (2.113) to match the 18 reference measurements, shown in upper left (in dBm). Resulting model parameters ψ_{ap}^* were $\boldsymbol{\rho}_{\mathrm{ap}} = (5.8, 5.2, 2.3)$, $P_{0\mathrm{ap}} = -39.5$, $\gamma_{\mathrm{ap}} = 2.2$, $\phi_{\mathrm{concrete}} = -7.9$ and $\phi_{\mathrm{drywall}} = -0.9$, yielding a RMSE of $0.3\,\mathrm{dB}$. Changing the transmitter's position yields highly discontinuous behavior, clearly denoting the location of the exterior concrete wall, ground-floor and ceiling.

vidual errors within the upper right of the figure. Furthermore, all resulting parameters are close to the ones used for the ray tracing that generated the synthetic reference measurements (cf. figure 2.24). Shown in the plots below, the behavior when changing the transmitter's location is highly discontinuous. The first two plots modify its x and y, clearly indicating the location of the exterior concrete wall as an abrupt change in error by ϕ_{concrete}. Similarly, adjusting z denotes the location of ground-floor and ceiling. As earlier, changing γ_{ap} and $P_{0\mathrm{ap}}$ yields a linear impact, as does changing the attenuations ϕ, which represent an additive factor, defined in (2.89).

While the presented optimization will consume significant amounts of time for larger buildings and numerous transmitters, it needs to be performed only once. During the localization of the pedestrian's smartphone, however, many intersection tests must be performed, in order to determine all attenuations induced by obstacles. Depending on the building's architecture, these intersection tests can be costly. Furthermore, for being used within the prediction model, the floorplan should be detailed, and must include information on each obstacle's material. For real-world scenarios, and use on smartphones, a compromise between both, the log-distance model and the extended log-distance model, should be taken into account.

Regional Log-Distance Model   To be suited for use on smartphones, the complexity of the extended log-distance model must be reduced, e.g. by lowering the number of needed intersection tests. One way of doing so is focusing only on obstacles imposing major attenuations, like ceilings, concrete walls and metallized glass. The latter are mainly found along a building's exterior, affecting localization outdoors, where GPS is available, and thus might be irrelevant,
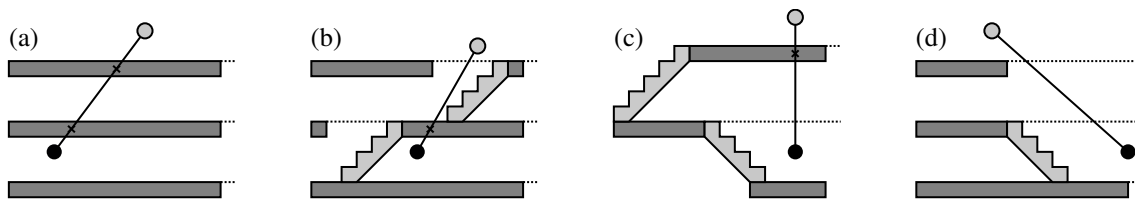
Figure 2.32: Potential encounters with floors/ceilings between a transmitter and some other location, in multi-level buildings. Several floors of equidistant height (a), cutouts for stairs or elevators (b), adjacent, shifted floors with different heights (c), atriums or galleries (d).

depending on the use case and building. Also, many modern buildings use steel-reinforced concrete in a few regions for structural purposes, and can thus be omitted as well. If the intersection problem is reduced to just ceilings, computational complexity can be decreased even further.

For buildings with stacked and similarly sized floors, ceilings can be expected everywhere, except a few small cutouts for stairs, and elevators. One option for modeling the effect of floor/ceiling attenuation is then given by artificially scaling the z-distance within the model. By increasing the distance towards the transmitter, the attenuation is increased as well. Doing so yields a target function similar to the one presented for the log-distance model, allowing for typical optimization algorithms. However, besides other drawbacks, such as invalid predictions along stairs, this approach is only suitable for buildings with equidistant floor heights.

Alternatively, the intersection test is replaced by discretely counting the number of floors between the transmitter's z-coordinate, and the one of the location in question. This also supports varying floor heights. Within older buildings, however, floors often are not evenly stacked, and multiple floors of different altitudes are *adjacent* to each other [Fet+18]. Here, the number of floors in between is (x, y) location dependent, requiring a different approach. The same holds true when floors contain larger cutouts, yielding a direct line of sight towards other levels, often encountered within modern architecture, with atriums, galleries, light wells, and observation platforms. When near such regions, counting the number of floors/ceilings will yield results differing from an actual intersection test [Ebn+17], with the signal strength estimation of the model far below real-world readings, due to invalid attenuations. The discussed cases are shown in figure 2.32, where counting the number of floors (dashed lines) between transmitter (black dot) and location in question (gray dot) is not always equal to an intersection test (cross).

One way to address aforementioned issues is given by splitting the building into sections, and to estimate one model for each such region. This is achieved by defining several bounding boxes, e.g. one per floor. All reference measurements that belong to one bounding box are used to optimize one log-distance model for every transmitter receivable within this bounding box. For evaluating the pedestrian's potential whereabouts based on current smartphone readings, the models that belong to the bounding box containing the location in question are compared
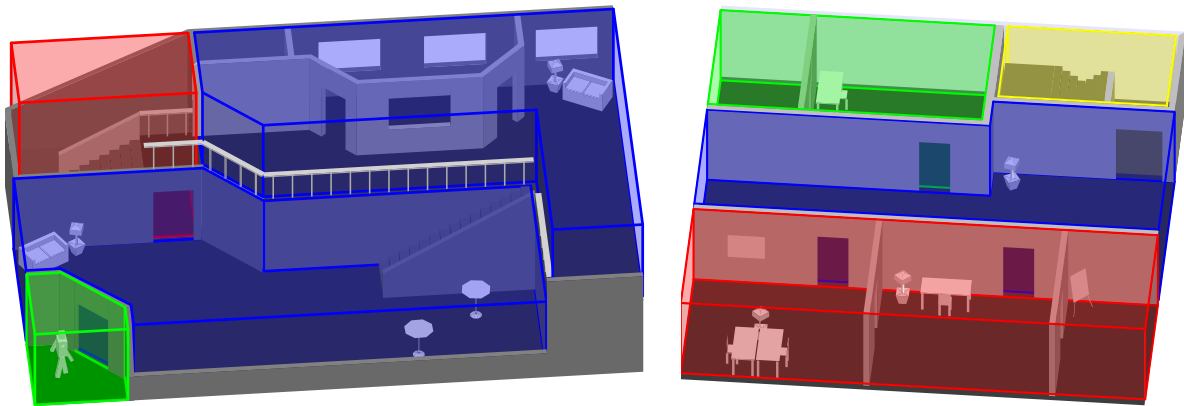
Figure 2.33: Example separation for a regional signal strength prediction model. By dividing a building's area into several regions, signal strength prediction models can focus on special needs within every section, thus reducing prediction errors.

with the phone's readings. As the check, whether a bounding box contains some point, is cheap, especially for axis-aligned bounding boxes, this approach yields major performance improvements compared to intersection tests. Furthermore, it is possible to use additional bounding boxes for sensitive areas, where the optimization process indicates major model errors, e.g. within stairwells, surrounded by massive concrete walls, shown in figure 2.33. However, every single bounding box needs at decent number of reference measurements for the optimization to converge, and to prevent overfitting [Ebn+17]. Also, similar to fingerprinting, invisible access points must be considered, as not all transmitters are visible within every bounding box, thus not yielding a model that provides predictions for every transmitter and location.

Bluetooth Beacons While aforementioned discussions were solely focused on RSSIs from Wi-Fi, they also apply to different radio hardware available for pedestrian smartphone localization, such as Bluetooth beacons. Independent of varying transmission protocols, they use similar frequencies, and are thus influenced by the same effects. One major difference is given by the transmission powers P and P_0. Like with access points, P is configurable for beacons. Yet, P_0 is also broadcasted by the protocol, making it visible to receiving clients. In theory, this value can thus be omitted from the presented optimization processes. However, as previous discussions have shown, models represent an approximation of real-world conditions, and parameters resulting from optimization are not necessarily equal to real-world parameters. Furthermore, beacons often are battery powered, thus using lower transmission powers than access points, also ceasing gradually. Thus, they are mainly suited for coarse location based services, or as supplement, used in locations, where Wi-Fi reception is poor [Ebn+15; Ebn+16].

2.7.7   Error Compensation

All of the discussed Wi-Fi localization approaches suffer from similar types of errors and uncertainties. Many of them could be addressed by using specialized hardware, which is not (yet) available within commodity smartphones. While current development is promising, eventually providing new components that increase the accuracy of smartphone-based location estimation [Ibr+18; Dvo+19; BSA16], it will take time for them to become widely available. The following paragraphs thus focus on the most common issues encountered for signal strength-based location estimation, and how to mitigate them.

Effects on RSSI   As mentioned in section 2.7.1, different kinds of antennae strongly affect the RSSI measurable by the smartphone. They are encountered as an additive gain $s_{\mathrm{ap}} = s'_{\mathrm{ap}} + G$, present within all readings received by the phone (cf. (2.84) and figure 2.18), affecting all evaluations presented earlier. Chintalapudi et al. [CPP10] address this problem by treating the indoor localization as crowd-based global optimization problem, where all unknowns, pedestrian locations and model parameters, are optimized together via genetic algorithms. While they also compensate for individual antenna gains, this is a big data approach with huge computational complexity, mainly intended for offline use. Wang et al. [Wan+11b] propose a more versatile solution. As antenna gain is a constant factor, all readings from a phone are affected in the same way. By evaluating relative instead of absolute signal strengths, the additive gain is removed

$$s_{\mathrm{ap}_1} = s'_{\mathrm{ap}_1} + G\,, \quad s_{\mathrm{ap}_2} = s'_{\mathrm{ap}_2} + G$$
$$s_{\mathrm{ap}_2} - s_{\mathrm{ap}_1} = (s'_{\mathrm{ap}_2} + G) - (s'_{\mathrm{ap}_1} + G) = s'_{\mathrm{ap}_2} - s'_{\mathrm{ap}_1}\,. \tag{2.118}$$

To use relative RSSIs, a common basis must be defined. Wang et al. suggest using the strongest transmitter received by the smartphone as reference, where all of the RSSIs within s are converted to be relative to the strongest one. Depending on the setup, the same step is applied to each fingerprint, or the values returned from signal strength prediction models. They are converted to be relative to the transmitter that was the strongest within the phone's readings. Discussed evaluations $p_{\mathrm{wifi}}(o_t \mid q_t)$ can hereafter be performed analogously [Ebn+14]. While this approach will often work as expected, the influence of the pedestrian shadowing all transmitters behind the phone can cause severe estimation errors. However, the same effect is also noticeable when using absolute readings [Zha+11].

Human Influence   As water strongly affects propagation of radio waves, and human bodies mainly consist of water, they strongly absorb Wi-Fi signals. When a pedestrian carries the phone upfront (cf. figure 2.5), all transmitters behind the pedestrian are attenuated, causing errors within the fingerprint or model comparison. If the current position and absolute walking

direction are known, e.g. from a previous location estimation or the phone's eCompass, and the pedestrian is assumed to carry the phone upfront, the measured RSSIs for all transmitters behind the pedestrian can be artificially increased, to compensate the effect of the human body.

Grouping Virtual Access Points   Another major drawback is presented by signal strength fluctuation due to environmental effects, such as multipath propagation, influences between nearby transmitters, and other pedestrians moving within the building, absorbing radio waves [YAS03]. While this can be addressed by low-pass filtering incoming RSSI readings, this causes already discussed side-effects, such as increased delays, affecting responsiveness [Ebn+14].

Network infrastructure within public buildings is often separated into domains with varying access rights. While staff members log into Wi-Fi A, guests use Wi-Fi B. To not require two identical installations for such scenarios, most access points support *virtual access point*s (VAPs). Here, the same hardware supplies several networks, usually on the same channel [Kit06]. This fact can be exploited while scanning for nearby transmitters, as each VAP will send its own response to a phone's scan request. VAPs can usually be identified by analyzing the MAC address, where only the last byte is different. This yields several RSSIs for the same physical hardware, sent with a minuscule time delay between them. By using the measurements' mean or median, the RSSI can be stabilized without introducing notable delays [Ebn+17].

This strategy especially applies to passive scans. If the receiving hardware supports *promiscuous mode*, which captures every single packet on one Wi-Fi channel, *each* data frame sent by a transmitter can be used as RSSI measurement. Within public buildings, where a constant amount of traffic can be expected, this provides dozens of additional signal strength indications, on each currently examined channel.

Varying Infrastructure   A major problem for all discussed Wi-Fi localization approaches are changes in infrastructure. When new transmitters are installed, or existing ones are (re)moved, new fingerprints or reference measurements must be conducted, or models must be adjusted. While new installations do not directly affect the location estimation, the system will not benefit from them until they are added. The impact of removing transmitters strongly depends on how invisible access points are handled within comparisons (cf. section 2.7.6). When existing transmitters are relocated, the evaluation becomes inoperable, as future smartphone RSSI readings do not match with fingerprints or model predictions. Similarly, within most buildings, there are more Wi-Fi transmitters besides the permanently installed infrastructure, such as smart TVs, digital projectors, medical equipment, personal hot spots, etc. Such non-stationary devices must be omitted within fingerprints or reference measurements, to ensure system operability. Besides consulting the building's IT Service Center to get an explicit list containing all permanent transmitters, manual approaches can be used as well. Usually, most administrators

will tend to install a single brand of devices for the Wi-Fi network architecture. By using the MAC address, the hardware vendor of a transmitter can often be identified.

Setup and Maintenance Times For new installations, or major changes of the Wi-Fi infrastructure, fingerprints and reference measurements must be (re)created. In case of fingerprinting, robots can be used to automate and speed up the process [Pal+11]. Instead of conducting individual measurements at fixed locations, walks along several ground truth points can be used to reduce the amount of time required for reference measurements [Tor+17]. Here, a few locations within the building are selected, and connected to denote a path. A person carrying a smartphone walks along this path at a constant speed, meanwhile recording the current signal strength readings and confirming whenever a ground truth point is reached. The constant walking speed allows for linear interpolation between adjacent ground truth locations. Combined with the timings of whenever such a location was reached, the pedestrian's position during the time of any RSSI measurement recorded by the phone can be determined [Gui+16]. This is similar to individual reference measurements, and can hereafter be used for model optimizations.

Overfitting When optimizing models based on training data, there is always a risk of overfitting. Thus, a decent number of reference measurements is required, especially when optimizing regional models for smaller fractions of the building. Another way to prevent such issues is given by including prior knowledge within the optimization process [SS09; GI10]. Applied to the presented target functions, the unknown parameters can be limited to a range, which is expected to be sane for real-world values. Whenever one of the parameters is outside of its sane range, the output of the target function is artificially increased, e.g. by adding a large *penalty*. The penalty should not be introduced abruptly but e.g. in a linear fashion, depending on how far the parameters are out-of-range, to not render the target function unnecessarily discontinuous.

Other Metrics Predicting signal strengths often is erroneous. Complex models, detailed floorplans, and reference measurements are required to achieve good results. Even if the models produce accurate results, or fingerprints are used, the pedestrian carrying the smartphone and the phone's antennae affect the RSSIs. This leads to the question of alternative metrics besides RSSI, that are more stable, less affected by architecture, and require reduced setup efforts.

Sen et al. [Sen+12] proposed a fingerprint-based approach, including additional information from the hardware layer of the Wi-Fi component. By using the phase and magnitude of multiple signal subcarriers instead of the RSSI, they achieved an accuracy of $\approx 1\,\mathrm{m}$. While the used hardware components were typical consumer products, the required metric is neither available by every commodity hardware, nor exported by smartphone operating systems.

Ancient seafarers and surveyors often used visible landmarks at known positions, such as special buildings, to estimate the current position. As the distance towards them was unknown,

they referred to angles, drawing lines, cutting each landmark at the observed angle, hereafter revealing the own position where all drawn lines intersect, known as *triangulation*. Applied to radio waves, multiple antennae, or antenna arrays, can be used to estimate the angle a signal is arriving from. Due to discussed multipath effects, however, there often might be more than one angle, or the required line of sight angle is not observed due to attenuation [DLK07]. Furthermore, smartphones and/or transmitters would require special antennae, potentially affecting their cost or normal use of Wi-Fi data transmission [HBS09].

Another complex thus focuses on the time needed for the signal to travel from the transmitter towards the receiver, referred to as *time of arrival* (TOA). Slight variations use the *time difference of arrival* (TDOA), where the timing difference between several signals is used instead, known from the GPS. The signal transit time is less affected by obstacles than the RSSI and thus more reliable [GH05]. Yet, multipath effects still play an important role [Ibr+18; McC+00]. Due to the speed of light, radio waves just need $\approx 3.3\,\mathrm{ns}$ to travel $1\,\mathrm{m}$. Distance measurements based on such timings thus require complex and fast dedicated hardware components, suffering from drifting over time, demanding for compensation techniques [McC+00; KAO08]. Günther and Hoene [GH05] suggested an approach that is intended to work with consumer hardware, measuring the round trip time between transmitter and receiver on the hardware level, using a certain part of the Wi-Fi standard, where the receiver is required to immediately respond to a received packet. As their hardware only contained timers with $1\,\mu\mathrm{s}$ accuracy ($\approx 300\,\mathrm{m}$), they utilized several statistical effects to estimate more accurate timings by combining many individual measurements. They point out that a viable distance estimation from a transmitter is possible after approximately 1000 measurements, or $\approx 1\,\mathrm{s}$. With at least three distances required for a 2D location estimation, the resulting delays are too substantial for indoor navigation.

Lately, distance measurements based on the signal's time of flight (TOF) have been added to the Wi-Fi standard [IEE16], referred to as *fine timing measurement* (FTM). Hardware components supporting this standard are able to infer their distance towards each other, using the TOA. By adding this to the standard, upcoming smartphones and corresponding operating systems will, or already do include this technique [Gooa], yielding completely new attempts for smartphone based indoor localization. While Ibrahim et al. [Ibr+18] have recently shown that this approach works in general, they still observed large errors due to multipath and similar effects. Nevertheless, this approach is expected to be refined over time, offering new fields of research, increasing the accuracy of upcoming Wi-Fi-based localization techniques.

2.8 Summary

Throughout this chapter, sensors available within commodity smartphones were examined, concerning their suitability for indoor localization and navigation. Therefore, sensor errors were discussed, distinguishing between accuracy and precision. Hereafter, these errors and indoor localization were brought together, by giving a probabilistic problem formulation, introducing the concept of evaluation and transition. Both relate to an unknown state, which is problem and sensor dependent, and e.g. denotes the pedestrian's unknown whereabouts, and the walking direction. Then, the general functionality of every sensor was presented, including required equations, providing an impression on repercussions, advantages and disadvantages. Thereby, a new probabilistic model was derived for every component, implementing the evaluation, denoting the probability for certain sensor observations, given some potential state.

Unavailable indoors, the global positioning system was mentioned for reference, and to discuss the concept of multilateration, which applies to other sensors as well. The location information provided by this sensor often is not zero mean. Therefore, a new probability density function was introduced, similar to a normal distribution, but without a clear mean value. This density is applicable to other sensors and components as well.

After introducing (pedestrian) dead reckoning, the smartphone's IMU was presented. Its accelerometer was used to perform a step-detection, by analyzing the behavior of the measurable gravity. To enhance detection results, sensor noise was suppressed using digital filters, which were introduced briefly. With steps being not always clearly measurable, the concept of probabilistic steps was introduced. Afterwards, step-detection was accompanied by turn-detection, using the gyroscope to estimate the pedestrian's turning behavior. For this to work, the readings from the gyroscope must be projected into what they would look like if the phone was placed parallel to the ground. This tilt compensation was given by the accelerometer's gravity reading, which was improved by the complementary filter, including readings from the gyroscope. However, due to the nature of the gyroscope, the probabilistic evaluations developed hereafter, suffer from cumulating drifts and increasing uncertainties.

The magnetometer was then presented as a source for absolute heading indications. Similar to the gyroscope, it required a prior projection, based on aforementioned concepts. Hereafter, probabilistic evaluations were developed and presented. With the sensor being subject to environmental effects, causing offsets, the density introduced for the GPS was applied to this evaluations as well. While not providing accurate results, the sensor can be used in conjunction with the gyroscope to suppress cumulating drifts.

Afterwards, the barometer was introduced as absolute altitude estimator. However, based on brief experiments, the influence of ambient conditions was depicted. Relative-pressure strate-

gies were introduced, to mitigate some of the resulting offsets. Yet, results are only valid as long as ambient conditions remain stable. Therefore, a relative probabilistic evaluation was developed besides the absolute variant.

With the barometer being mainly suited for short-term information, the field of activity-detection was briefly discussed. Here, the barometer was e.g. used to detect short-term altitude changes, to infer whether the pedestrian is currently taking stairs or walking along ground. The developed probabilistic evaluation provided a coarse absolute location estimation, for example by limiting potential whereabouts to staircases.

With neither of the previous sensors providing true 3D location estimations, Wi-Fi was introduced as a potential source for 3D localization indoors. After discussing physical effects on radio waves, the relation between signal strength and distance became clear. Hereafter, signal strength prediction models were briefly introduced. One of them was used to convert measured signal strengths into distances, allowing for lateration, as introduced for the GPS. With the resulting accuracy being too coarse, and results limited to a single result, the concept of fingerprinting was introduced. Based on numerous real-world measurements, they allow for an accurate, yet discrete, probabilistic localization. A compromise was developed, using advanced signal strength prediction models, predicting what fingerprints should look like. This allowed for a continuous and probabilistic location estimation, but required several model parameters to be estimated. The latter was performed using a few reference measurements, and numerical optimization. Hereafter, several workarounds were developed, improving the localization result.

Based on the presented models, potential pedestrian movements and whereabouts can now be evaluated in a probabilistic manner. However, it is yet unclear how to combine the information of all individual sensors. Furthermore, with only Wi-Fi providing 3D localization, additional information and constraints are required, like from the building's floorplan. Similarly, for navigation to be performed, this mapping information is required as well. These aspects represent the topics of the two following chapters.

Chapter 3

Probabilistic Movement Models

One of the main problems with previously discussed sensors is measurement noise, even if their uncertainty is known beforehand and modeled via probability densities. The location estimation given e.g. by a GPS sensor can span several meters, even under good conditions. While weather has only a minor effect on accuracy, occlusion by trees and tall buildings affects performance, reducing the number of usable satellites [Oga11]. However, no commercial navigation system displays a virtual car jumping around the screen, while waiting at a red traffic light, surrounded by skyscrapers. For the described situation, additional information, such as the car's current velocity, might be used to suppress movement due to GPS measurement noise, e.g. by using the presented complementary filter [Bon+01; AP99; SND99]. While this approach will work for a non-moving vehicle, it is only a partial solution when moving, as the known velocity can only be used to limit the distance of the movement, not to constrain its direction.

As cars can be expected to reside on a road most of the time, mapping information provides additional constraints. Therefore, navigation systems e.g. snap the virtual car onto the street nearest to the estimated position, using some sort of heuristic or geometric rules [JSZ04]. Yet, if there is more than one street nearby, like for trunks and intersections, this approach can yield incorrect results, snapping the estimation onto the wrong street (cf. figure 3.1a). Furthermore, while this ensures the virtual car to be located on some street, measurement noise can still cause leaps along, or jumps between adjacent roads. As with the sensor estimations from chapter 2, such discrete approaches often do not provide a satisfying solution to the problem. Instead, the derived continuous probabilities for sensor readings should be used, and combined with the map. To suppress invalid locations on a probabilistic basis, leaving only those that match with the surroundings, shown in figure 3.1c. When additional information, such as the current driving direction is available, potential whereabouts can be constrained even further. In figure 3.1d, a single road remains likely, with the car's location uncertainty distributed alongside.
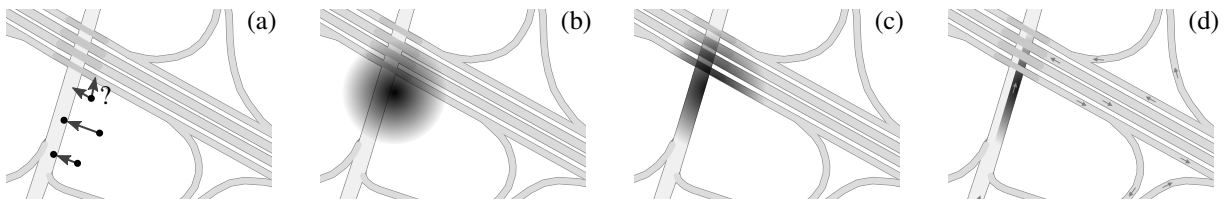
Figure 3.1: Example combinations of sensor readings and map data for car navigation. Snapping discrete GPS indications to the nearest street (a), using probabilities without map knowledge (b), combining map and probability information (c), additionally including movement direction information (d).

The described approach does not only improve the current location estimation, but can also be used to refine subsequent steps. Dependent on the receiver, the GPS provides the next location reading between one and ten times per second [TY09]. Hereafter, the car's new location is not only constrained by recent sensor values and the map. New whereabouts also depend on the previous location, the elapsed timeframe, current driving speed, surroundings, like the type of street (highway, farm road), nearby intersections, one way streets, and similar. The road map, and some set of rules on typical car movements, can be used to create a *model*, statistically predicting the car's *potential movements*, incorporating additional information, such as speed limits, the size of the vehicle, whether it has to slow down to take a certain road with a narrow turn, the driver's desired destination, and many more. Instead of simply snapping a single discrete measurement onto the nearest street, the probabilistic prediction of the model is combined with recent measurements, including their uncertainty. New potential whereabouts are then given where both, the model and the sensor readings, agree.

3.1   Probabilistic Problem Formulation

The same applies to the scenario of indoor localization and navigation, where the model describes potential pedestrian movements within a certain timeframe. Instead of road maps, the building's floorplan is used to limit potential walks and whereabouts. This prevents noisy sensor data from yielding unlikely or impossible movements, by including walls, obstacles, stairs, and elevators. Describing the change, or transition, from one location, or state, to a new one within some timeframe, it is referred to as *transition model*. It defines the probability of a new state $\boldsymbol{q}_t$ after some time Δt, given an old state $\boldsymbol{q}_{t-1}$. The probability to *"move"* from one state to another during some time Δt is written as

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}). \tag{3.1}$$

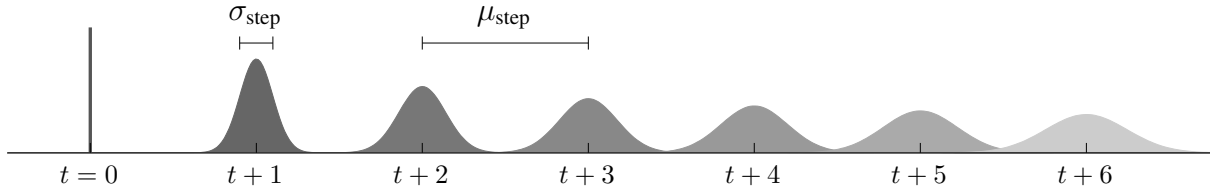A simple 1D transition model is depicted in figure 3.2. The pedestrian is assumed to walk to the

Figure 3.2: Several one-dimensional transitions using (3.2), starting from a known position at time $t = 0$. Each transition makes one step to the right, using an average step size μ_{step} with some uncertainty σ_{step}. The uncertainty of the potential whereabouts increases with every additional transition.

right, with an average step-size μ_{step}, and uncertainty σ_{step}, modeled by a Gaussian. Walking is limited solely to the x-dimension, and initial whereabouts $q_0^{(x)}$ at time $t = 0$ are known exactly

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) = \mathcal{N}\left(q_t^{(x)} \;\middle|\; q_{t-1}^{(x)} + \mu_{\text{step}}, \sigma_{\text{step}}^2\right), \quad \langle \boldsymbol{q} \rangle_t = \langle (x) \rangle_t. \tag{3.2}$$

Regarding probabilities, the exactly known initial position $q_0^{(x)}$ is denoted by a single spike. A transition from this position is similar to the pedestrian taking one step, introducing some uncertainty, which is equal to σ_{step} after the first step. Intuitively, this uncertainty increases with every subsequent transition, similarly to the cumulative sensor errors discussed in section 2.4.

For predicting actual pedestrian movements indoors, the state is required to contain at least the current 2D location for single floors, or 3D for multistory buildings. The actual formulation strongly depends on available knowledge. Based on the initial car navigation example, a floor-plan, the current walking speed, and heading represent a valuable contribution. The following discussions gradually include prior knowledge, starting without using sensor information at all.

The model's suggestion for new whereabouts is constrained by the distance a pedestrian is able to walk within a certain timeframe. Depending on age and gender, the average walking speed ranges around $1.4\,\text{m/s}$ with a deviation of approximately $0.1\,\text{m/s}$ [KPN96; Bro+06], varying with the current surroundings. Regarding pedestrian walking behavior near outdoor street crossings, or indoors when taking stairs, the deviation in walking speed among several study participants was as large as $0.25\,\text{m/s}$ [GB15; FT04]. In case of stairwells, the taken step size directly depends on the tread length of the stair [TG91]. If the model is able to distinguish between situations, such as walking outdoors, indoors or along stairs, it could simply rely on average speeds from literature. However, the pedestrian might not be moving at all, resting at the current position. This behavior could be described by a model including both cases, where some transitions use the average velocity, and others do not move at all. Alternatively, the velocity could be completely randomized. Yet, this mainly yields movement predictions not resembling real-world conditions [Ebn+14; KGD14].

While walking speeds can be included using values from literature, this does not apply to the current heading and its changes. Even if the previous heading is given, the amount of change within a certain timeframe is unknown for pedestrians. Unlike with cars, where the driving direction can be assumed to follow the course of the road until some intersection is reached, indoor navigation is often open-spaced, providing fewer limitations on potential movements. While there are experimental analysis of human walking behaviors, e.g. for pedestrian groups leaving narrow passages conducted by Daamen and Hoogendoorn [DH03], it usually does not make sense to assume a fixed heading deviation for the pedestrian, as directional adjustments are based on desired destination, obstacles, and building architecture. While simply using an empiric best-fit choice as heading deviation is possible, a single value does not model all potential walks, as the pedestrian might turn at any point in time. Like with walking and resting, there are several options to address this issue. Assuming a large heading deviation per second, using a small deviation but allowing multiple directions (forward, left, right, turning backwards), or omitting the heading information completely, to allow walking into any reachable direction. Especially the latter reveals an important aspect on the behavior of transition models.

To examine the impact, (3.2) is altered, omitting the known movement direction towards the right, by replacing $+\mu_{\text{step}}$ with $\pm\mu_{\text{step}}$. For the newly created model there now is a major difference between one prediction of a $5\,\text{s}$ timeframe and five consecutive predictions, with $1\,\text{s}$ each. For the first case, the velocity is simply increased by a factor of 5, with the result denoting two peaks. One to the left, one to the right, both at a distance equal to the adjusted velocity. For the second case, the behavior is far more complex. After the first of five steps, two peaks with their distance equal to $1\,\text{s}$ of walking are created. The second transition starts from this result, again into an unknown direction, yielding *two new peaks for each of the previous ones*. After five steps, this produces a highly complex, non Gaussian, and possibly unexpected result.

This lack of prediction quality, resulting from the unknown heading, can e.g. be improved by information from turn-detection or the eCompass (cf. section 2.4) [Ebn+14; Ebn+15]. To include such observed sensor data within the transition in general, (3.1) is adjusted, yielding the same notation as (2.6), previously discussed in chapter 2

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})\,. \tag{3.3}$$

Depending on the information available, the prediction of the model will look completely different. Furthermore, its behavior depends on the required dimensionality. In a two dimensional setup, sufficient for single floor buildings, possible movements can be described using velocity and a heading angle Θ, restricted by walls or obstacles. Referring to three dimensional setups, implementation strongly depends on the surroundings. For mid-air airplanes, for example,
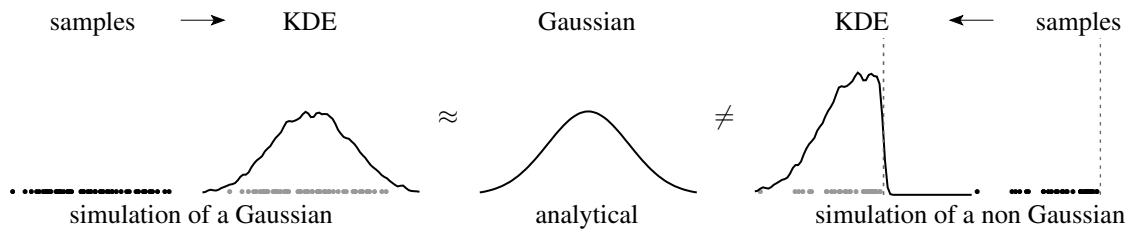
Figure 3.3: Brief comparison between analytical and simulated probability density estimation. When using random samples and a KDE, arbitrary non-analytical densities can be modeled. Shown in the right, this e.g. allows for obstacles (dashed-line) to be considered within predictions.

there is no limitation on potential movements in 3D space, described by a velocity and, at least, two angles, or a 3D heading vector. In pedestrian indoor localization, however, only stairs, escalators and elevators allow for changes along the z-axis. Most of the time, movements are restricted to the (x, y)-plane, and the possibility of z-changes depends on the underlying architecture. This drops the requirement for a second angle, and the floorplan represents the most important information to constrain potential walks.

As walking through walls is impossible, and movements in the z-direction require stairs, escalators or elevators, the floorplan adds discrete constraints on potential transitions. A wall within figure 3.2 would prevent movements, causing an abrupt drop in probability for all locations behind it. Due to this discontinuous impact, analytical solutions, such as (3.2), are often unavailable, and the density can neither be calculated nor visualized directly.

Complex transition models are thus statistically approximated by *simulation*. For every input state q_{t-1}, potential output samples q_t are created by a random process, approximating $p(q_t \mid q_{t-1})$ or $p(q_t \mid q_{t-1}, o_{t-1})$, with theoretical backgrounds examined in chapter 4. The resulting density is described by a *set of multiple samples*. Similar to the discussions on fingerprinting (see section 2.7), a continuous representation can hereafter be derived, e.g. by applying a KDE to these samples. A brief comparison between a 1D analytical and simulated representation is shown in figure 3.3, where a wall abruptly limits density propagation. The number of samples required for such an approximation strongly depends on the to-be-approximated density, where broader densities require more samples for a viable result. However, due to computational costs, this number is a crucial aspect, concerning use on embedded devices.

This chapter focuses on presenting different types of movement models, with and without additional sensor information, ranging from simple analytical setups without prior knowledge, towards complex discontinuous approaches based on a floorplan. Though not explicitly mentioned within above notation, all models can utilize this floorplan. For now, it is assumed to be given, including all semantic details, potentially required for a specific model, shown in figure 3.4. This mainly covers individual floors including their walls, obstacles, doors, stairs, and
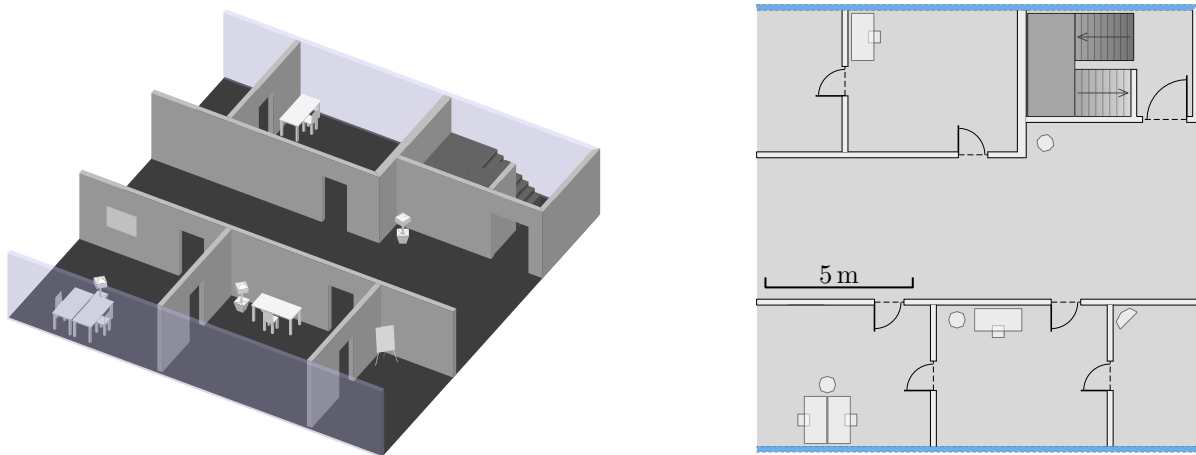
Figure 3.4: Example showing a 3D and top-view of a single floor with a corridor and adjacent rooms. Besides walls, there can be other types of obstacles. The upper floors can be reached via a stairwell.

elevators, which are relevant to localization and navigation. The actual creation of the floorplan is discussed later in chapter 5.

Due to the work's focus on smartphone-based pedestrian indoor localization and navigation, computational complexity and memory constraints are examined as well. This not only affects the complexity of the possible movement models, but also the complexity of included prior information, such as the floorplan. Depending on the actual use case, simple analytical models without architectural knowledge might be sufficient, and allow for fast calculations with small memory footprint. For fully featured navigation, more sophisticated setups, including the actual architecture, are required. Resulting constraints, advantages, and disadvantages of the individual approaches will also be part of the following discussions. Furthermore, the different requirements between single-floor 2D and multi-floor 3D setups are examined as well.

3.2   Simple Models without Floorplan Information

Similar to sensors in chapter 2, models for potential movements should include uncertainty expectations, as depicted for the unconstrained one dimensional walk in figure 3.2. Instead of using a single scalar value to describe new whereabouts, a distribution models the likelihood of nearby locations as well. The most simple setup uses a normal distribution to describe this uncertainty around the mean, or expected value. This section introduces simple transition models, based on such analytical distributions, and examines the required steps to adapt them for 2D and 3D localization setups. While being both, computationally efficient and continuous, environmental information, such as a floorplan, can not be considered. Due to the continuous nature, the distribution propagates through walls and other obstacles, reducing the number of use cases.

Similar to the 1D case from figure 3.2, for a simple setup with no obstacles nearby, a 2D/3D variant is given by the multivariate normal distribution, describing the uncertainty around a mean value $\boldsymbol{\mu}$ using a matrix $\boldsymbol{\Sigma}$, resulting in a multidimensional Gaussian:

$$p(\boldsymbol{q}) = \mathcal{N}\left(\boldsymbol{q} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)^d \det\left(\boldsymbol{\Sigma}\right)}} \exp\left(-\frac{1}{2}(\boldsymbol{q} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{q} - \boldsymbol{\mu})\right) . \tag{3.4}$$

In (3.4), $\boldsymbol{q}$ and $\boldsymbol{\mu}$ e.g. denote a 2D location $(x, y)^T$ or 3D location $(x, y, z)^T$. The *covariance* $\boldsymbol{\Sigma}$ describes the uncertainty around $\boldsymbol{\mu}$. For the 1D case, the uncertainty σ distributes equally to either side of the mean. For the multidimensional case, it distributes into as many directions as the problem has dimensions. While the distribution is still symmetric around $\boldsymbol{\mu}$, it is neither distributed with the same amount into each dimension, nor are the dimensions required to be aligned with the axes of the coordinate system, as long as they are orthogonal. This can easily be verified when looking at pedestrian movements. Depending on current heading and speed, x, y and z will be affected differently. The amounts and directions of uncertainty distributions are modeled within the covariance matrix $\boldsymbol{\Sigma}$ [BC12; HTF09].

Assuming a 2D localization problem, where the pedestrian's initial position $\boldsymbol{q}_0 = (x, y)^T$ at time $t = 0$ is known with some uncertainty, and no additional information is available. The initial whereabouts $\boldsymbol{q}_0$ can be described using a two dimensional normal distribution, with the mean equal to the known position, and the covariance as diagonal matrix with the same uncertainty in x and y, and no dependency between both

$$p(\boldsymbol{q}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \quad \boldsymbol{q}_0 = \boldsymbol{\mu}_0 = (x, y)^T, \quad \boldsymbol{\Sigma}_0 = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} . \tag{3.5}$$

The 1σ contour of (3.5) denotes a circle around the mean that ensures with $\approx 68\,\%$ confidence that the pedestrian resides within. If neither the current heading nor the velocity are known, this uncertainty is expected to grow around the initial center over time. This is achieved by keeping $\boldsymbol{\mu}$ as-is, and increasing σ, e.g. by the average walking speed of $1.4\,\mathrm{m/s}$, every second. The resulting 1σ contour still contains the pedestrian with a confidence of $68\,\%$, when not walking faster than $1.4\,\mathrm{m/s}$ on average

$$p(\boldsymbol{q}_t) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_t), \quad \boldsymbol{\Sigma}_t = \begin{pmatrix} \left((t+1)\sigma\right)^2 & 0 \\ 0 & \left((t+1)\sigma\right)^2 \end{pmatrix} = (t+1)^2 \begin{pmatrix} 1.4^2 & 0 \\ 0 & 1.4^2 \end{pmatrix} . \tag{3.6}$$

As (3.6) describes σ linearly increasing with t, visualizing the 1σ contour for several t yields a set of circles with their radius increasing by a constant factor, shown in figure 3.5. This absolute notation, describing the uncertainty at time t, can be converted to the initially introduced relative

notation (3.1), describing the change in uncertainty between two timesteps, by using the linear increment that must occur for every such step

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) = \mathcal{N}\left(d_1 \mid \zeta d_2, \sigma^2\right) , \quad \sigma \to 0$$

$$d_1 = \mathrm{dist}_{\mathrm{xy}}\left(\boldsymbol{q}_0, \boldsymbol{q}_t\right) , \quad d_2 = \mathrm{dist}_{\mathrm{xy}}\left(\boldsymbol{q}_0, \boldsymbol{q}_{t-1}\right) , \quad \langle \boldsymbol{q} \rangle_t = \langle (x,y) \rangle_t , \quad \zeta = \frac{t+2}{t+1} . \tag{3.7}$$

Assuming $p(\boldsymbol{q}_0)$ was well defined, e.g. by using (3.5), (3.7) ensures that all *subsequent* steps propagate in the same manner. If $\boldsymbol{q}_{t-1}$ was d away from the initial $\boldsymbol{q}_0$, $\boldsymbol{q}_t$ must be ζd away from it, where ζ enforces the linear growth from (3.6). When not insisting on this growth to be linear, just allowing some variation within every timestep, the relative update from (3.7) can be reduced to a significantly simpler, straightforward representation [Ebn+15]:

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) = \mathcal{N}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{\Sigma}\right) , \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{xy}^2 & 0 & 0 \\ 0 & \sigma_{xy}^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{pmatrix} , \quad \langle \boldsymbol{q} \rangle_t = \langle (x,y,z)^T \rangle_t . \tag{3.8}$$

(3.8) enforces $\boldsymbol{q}_t$ to be within the vicinity of $\boldsymbol{q}_{t-1}$, and also applies to the 3D case, allowing for a different variation along the z axis, which is usually smaller than for x and y [Ebn+15].

While (3.6) is an absolute analytical model that is directly calculable, (3.7) and (3.8) are not. As they are relative, they depend on the previous density, to perform the adjustments. The coherence between both is given by the *law of total probability* [BC12, pp. 69]:

$$p(\boldsymbol{q}_t) = \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) \, p(\boldsymbol{q}_{t-1}) \, \mathrm{d}\boldsymbol{q}_{t-1} . \tag{3.9}$$

As can be seen, (3.9) denotes a recursive chain that propagates until the terminal $p(\boldsymbol{q}_0)$ is reached. Whether it can be calculated in an analytical way, depends on the used probability density functions (PDFs). While (3.7) can be calculated analytically by (3.6), most transition models presented within this chapter can not, and are approximated by simulation. To derive $p(\boldsymbol{q}_t)$, the simulation starts with the set of samples for $p(\boldsymbol{q}_0)$, and applies (3.9) recursively, where each $\boldsymbol{q}_{t-1}$ leads to a new sample set for $p(\boldsymbol{q}_t)$. To accurately approximate the resulting density, a decent number of samples is required, strongly dependent on its size and shape. An example is shown in Figure 3.5, containing both, the analytical 1σ contours shown as circles, and simulated samples shown as dots. In the right, the expected continuous output after applying a KDE to the samples from the left, is shown as dark background shading. While the 1σ contour increases constantly over time, the figure clearly shows that major parts of the probability mass remain centered around the mean value, typical for a single Gaussian. Intuitively, this setup does not represent true expectations on pedestrian movements indoors.
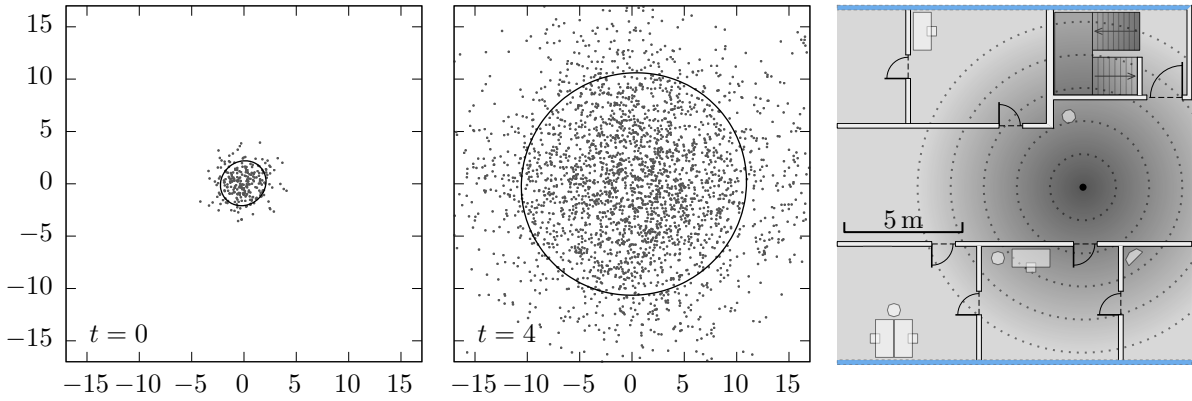
Figure 3.5: Behavior of (3.6)/(3.7) during five timesteps, with corresponding 1σ contours as circles. The two left images denote the simulated sample sets and contours for $t = 0$ and $t = 4$. In the right, the contours for $t \in \{0, \ldots, 4\}$ are shown together with $p(\boldsymbol{q}_t)$ at $t = 4$ denoted as dark shading.

A slightly more realistic model is given by the following assumption: At any point in time t, a pedestrian could either be walking with a velocity μ_{walk}, or resting, with the information e.g. provided by an observed activity $o_{t-1}^{(\Omega)}$ (see section 2.6):

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) = \begin{cases} \mathcal{N}\left(d \mid \mu_{\text{walk}}, \sigma_{\text{walk}}^2\right) & o_{t-1}^{(\Omega)} = \texttt{walking} \\ \mathcal{N}\left(d \mid 0, \sigma_{\text{stand}}^2\right) & o_{t-1}^{(\Omega)} = \texttt{standing} \end{cases} \tag{3.10}$$

$$d = \text{dist}_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right), \quad \langle \boldsymbol{q} \rangle_t = \langle (x, y, \ldots) \rangle_t, \quad \langle \boldsymbol{o} \rangle_t = \langle (\Omega, \ldots) \rangle_t.$$

The *standing* case of (3.10) is similar to the previous model, and describes an increase in uncertainty, around the previous location estimation $\boldsymbol{q}_{t-1}$. The *walking* case, however, is completely different, and models the next whereabouts to reside along the circumference of a circle, with the radius defined by the expected walking speed μ_{walk}. Even though the walking case is described using a normal distribution, the resulting likelihood denotes a non Gaussian distribution (cf. figure 3.6). If the information, whether the pedestrian is walking or resting, is unavailable, heuristic assumptions can be made instead [Ebn+15]:

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) = \kappa_{\text{w/s}} \mathcal{N}\left(d \mid \mu_{\text{walk}}, \sigma_{\text{walk}}^2\right) + (1 - \kappa_{\text{w/s}}) \mathcal{N}\left(d \mid 0, \sigma_{\text{stand}}^2\right). \tag{3.11}$$

Using $\kappa_{\text{w/s}} \in [0, 1]$, a general empiric probability for walking or resting can be used to mix both cases. Figure 3.6 shows a simulation for $\kappa_{\text{w/s}} = 0.9$, with the chance of standing to be 10%, and a small uncertainty $\sigma_{\text{walk}} = \sigma_{\text{stand}} = 0.05$ for visualization reasons. After one timestep, the result is as expected. One inner spot that kept the previous position, and a circle $\mu_{\text{walk}} = 1.4\,\text{m}$ apart. The results of the followings steps are rather unexpected. As (3.10) and (3.11) do not contain any constraints on the walking direction, all movements from (b) to (c) are completely
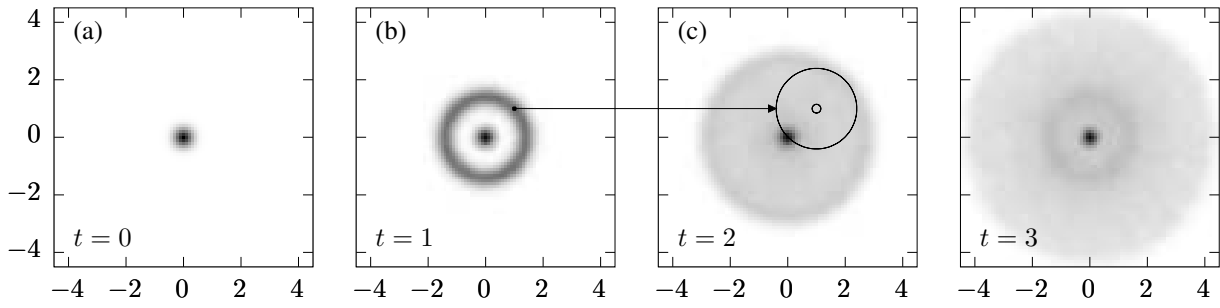
Figure 3.6: Simulated behavior of (3.11) using $\mu_{\text{walk}} = 1.4\,\text{m}$ and $\sigma_{\text{walk}} = \sigma_{\text{stand}} = 0.05$. While the first step is as expected, all following steps behave differently. This is due to (3.11) not constraining the walking direction, just staying at or departing from the *previous* location by a given distance. All samples from (b) end up in (c) somewhere nearby (when resting), or are located along a circle (when walking).

undirected, and solely enforced to either rest with a chance of $10\,\%$, or move by approximately $1.4\,\text{m}$. This yields each simulated sample from (b), to end up in (c) with the same spot/circle pattern, as from (a) to (b). An unrestricted heading for every subsequent step within a small timeframe does not resemble human walking behavior. As within chapter 2, considering the heading in some way seems mandatory.

Again, this can be addressed by adding a heading information Θ to the unknown state $\boldsymbol{q}$. If no prior information is available, the initial heading $q_0^{(\Theta)}$ at time $t = 0$ remains unknown, and thus $q_0^{(\Theta)} \sim \mathcal{U}(0, 2\pi)$. When simulating $p(\boldsymbol{q}_0)$, this yields a set of samples, with uniformly distributed heading. All subsequent transitions use this heading, and allow only slight deviations from it, modeling a pedestrian walking almost straight from the starting position. Assuming statistical independence, this is achieved by multiplying another probability as constraint

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) = \mathcal{N}\left(\alpha \mid 0, \sigma_{\text{turn}}^2\right) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})_{(3.10)}$$

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) = \mathcal{N}\left(\alpha \mid 0, \sigma_{\text{turn}}^2\right) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})_{(3.11)} \qquad (3.12)$$

$$\alpha = \angle_\Delta\left(q_{t-1}^{(\Theta)}, \angle_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)\right), \qquad \langle \boldsymbol{q}\rangle_t = \langle(x, y, \Theta, \ldots)\rangle_t.$$

The additional constraint enforces the state's heading to be kept approximately, yielding almost straight walks. This is achieved by comparing the expected direction $q_{t-1}^{(\Theta)}$ against every potential heading, given by the angle between previous and new whereabouts $\angle_{\text{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)$ (2.52). The difference $\angle_\Delta(\alpha, \beta)$ between the expected and the potential angle is provided by (2.53). This angular difference should be rather small, and is thus applied to a zero mean normal distribution with some uncertainty, to allow slight variations. Again, other PDFs, such as the von Mises distribution [Mis18], are possible as well. The simulated result of (3.12) is shown in figure 3.7, and resembles water drops, with several equidistant circles, denoting the potential whereabouts after consecutive transitions.
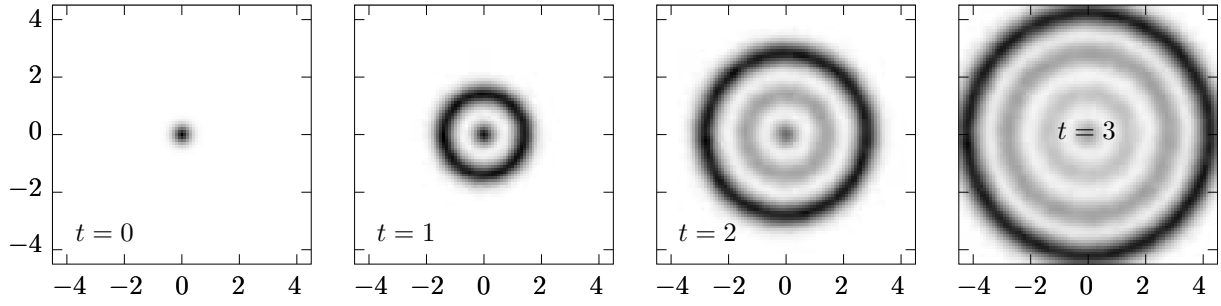
Figure 3.7: Simulated behavior of (3.12) with uniform initial heading $q_0^{(\Theta)} \sim \mathcal{U}(0, 2\pi)$, walking speed $\mu_{\text{walk}} = 1.4\,\text{m/s}$, and uncertainties $\sigma_{\text{walk}} = \sigma_{\text{stand}} = \sigma_{\text{turn}} = 0.1$. The result looks similar to sea waves, where the equidistant circles denote potential whereabouts, created by the underlying model.

(3.12) is versatile, and also applies when the pedestrian's initial heading is known, e.g. from an eCompass, or prior knowledge. However, it does not define how this assumed heading might change between $q_{t-1}^{(\Theta)}$ and $q_t^{(\Theta)}$, e.g. when the pedestrian is turning. Therefore, another component is added. It describes the probability of the new state's heading, based on observed changes (2.50), absolute indications (2.51) and (2.60), or when unobserved by sensors

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) = p_{\text{head}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}\right) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) \tag{3.12}$$

$$p_{\text{head}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}\right) = \begin{cases} \mathcal{N}\left(q_t^{(\Theta)} \mid q_{t-1}^{(\Theta)} + o_{t-1}^{(\theta)}, \sigma_{\text{turn}}^2\right), & \langle \boldsymbol{o} \rangle_t = \langle (\theta, \ldots) \rangle_t & \text{relative} \\ \mathcal{N}\left(q_t^{(\Theta)} \mid o_{t-1}^{(\Theta)}, \sigma_{\text{turn}}^2\right), & \langle \boldsymbol{o} \rangle_t = \langle (\Theta, \ldots) \rangle_t & \text{absolute} \\ \mathcal{N}\left(q_t^{(\Theta)} \mid q_{t-1}^{(\Theta)}, \sigma_{\text{turn}}^2\right) & & \text{unobserved}. \end{cases}$$
$$\tag{3.13}$$

A simulation of (3.13) for a synthetic scenario, where the pedestrian's initial whereabouts and heading are approximately known, and updated relatively hereafter, is shown in figure 3.8. Up until (c), the behavior is similar to the one from figure 3.7, except that the whereabouts only denote a *fraction* of a circle, due to the initially known heading. As soon as a sensor indicates relative changes, this rotation affects the shape of the whole probability density. After several $45°$ turns, the result in (h) does not resemble an arc, but looks similar to a Gaussian. This effect occurs after consecutive larger turns, increasing the uncertainty in multiple directions, based on σ_{turn}. For longer straight walks, the density will often be shaped as seen in (b) and (c).

When not distinguishing between walking and resting, or this information is provided by some sensor, it is possible to approximate aforementioned model analytically. Using a single normal distribution with a relative update function, the model becomes directly calculable, and does not require simulation. The current covariance, omitting time indices for readability, can be derived by using an uncertainty for the walking speed, and one for the heading. This diagonal covariance matrix $\boldsymbol{\Sigma}$ is hereafter transformed by a 2D rotation matrix $\boldsymbol{R}$ (see section 2.4.2),

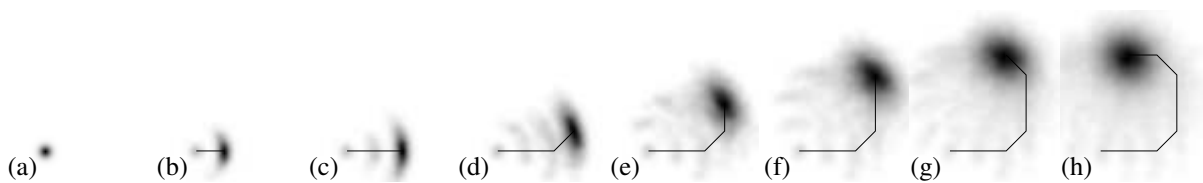(a)   (b)   (c)   (d)   (e)   (f)   (g)   (h)

Figure 3.8: Simulated behavior of (3.13) with narrow initial heading $q_0^{(\Theta)} \sim \mathcal{N}(0, 0.1)$, walking speed $\mu_{\text{walk}} = 1.4\,\text{m/s}$, and uncertainties $\sigma_{\text{walk}} = \sigma_{\text{stand}} = 0.05$ for speed, and $\sigma_{\text{turn}} = 0.25$ for heading. Visible in (b) and (c), the approximately known initial heading yields only a fraction of the circle from figure 3.7. When the heading starts to change between (c) and (d), the shape of potential whereabouts is affected, and rotated as well. After several $45°$ turns, the result in (h) is similar to a Gaussian.

defined by the current heading $q_t^{(\Theta)}$. $\boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T$ (see [CK08]) rotates the uncertainties in speed and direction, defined in $\boldsymbol{\Sigma}$, to match the current heading. The covariance at time t can be combined with the previous one from $t-1$, adjusting its shape and size over time. The new mean value $\boldsymbol{\mu}$ at time t is given, by adding a vector denoting the current walking direction $(\cos q_t^{(\Theta)}, \sin q_t^{(\Theta)})^T$ times the estimated walking speed μ_{step}, to the previous mean from $t-1$. Even though being an approximation of the unique shapes from figure 3.8, such basic analytical setups, also like (3.8), can already be sufficient for certain use cases.

However, neither of the presented methods considered actual floorplan information when deriving new whereabouts. Shown in figure 3.5, this creates potential issues, with predictions crossing walls. Including obstacles is non-trivial, as it results in discontinuous behavior. Deriving movement models with support for such information will be the topic of the next section.

3.3   Simple Models with 2D Floorplan

To accurately predict potential whereabouts after a certain amount of time, constraints given by the surroundings must be considered. Similar to car navigation systems, which limit potential movements based on mapping data. Within indoor environments, walls and other obstacles limit the pedestrian's potential movements. When currently residing in front of a wall, the possibility of being located on the opposite side after a certain amount of time is rather small, and strongly depends on the presence of nearby doors. Including such constraints in an analytical way, where one equation describes a probability density function based on the current whereabouts, considering obstacles, is not feasible. Literature suggests several strategies, which are mainly based on aforementioned approximation, by simulating several potential movements, each considering the ambient surroundings.

Proceeding from one of the previous relative movement models, such as (3.13). A simple solution for considering obstacles is then given by adding a new probability, testing whether the
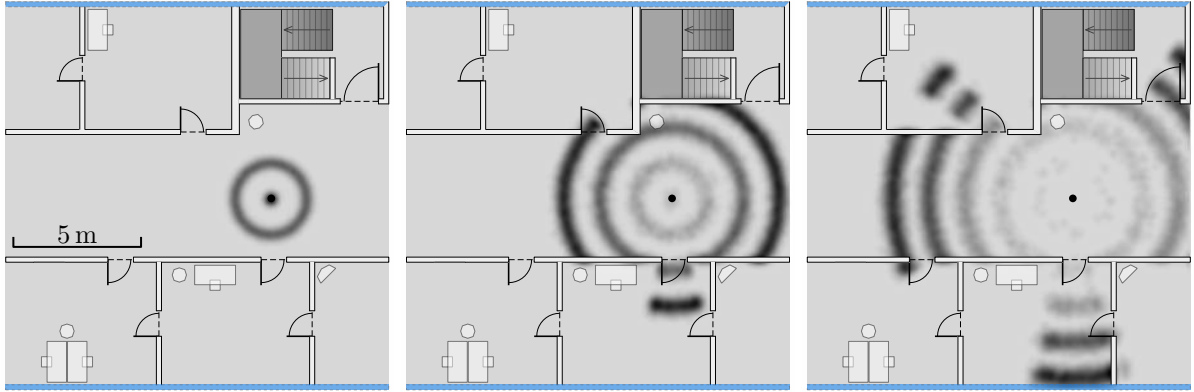
Figure 3.9: Potential whereabouts (3.14) after 1, 3 and 5 consecutive transitions, starting from the black dot, walking straight with $\mu_{\text{walk}} = 1.4\,\text{m/s}$ and $\sigma_{\text{walk}} = 0.1$, into an unknown direction $q_0^{(\Theta)} = \mathcal{U}(0, 2\pi)$. Darker shadings denote an increased likelihood. After $1\,$s, whereabouts are placed along a circle, similar to figure 3.7. As soon as obstacles are encountered, the potential walk is blocked. After 5 steps, the circular shape is still visible, split into several fractions, where movements were not blocked by obstacles.

potential movement from location $\mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_{t-1}\right)$ towards $\mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_t\right)$ is blocked by an obstacle, and, if so, assign this movement a zero or near-zero probability [Ebn+14]:

$$p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) = p_{\text{free}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}\right) p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})_{(3.13)}$$

$$p_{\text{free}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}\right) = \begin{cases} 1.0 & \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t \text{ free} \\ 0.0 & \boldsymbol{q}_{t-1} \to \boldsymbol{q}_t \text{ blocked} . \end{cases} \tag{3.14}$$

A simulation example for $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})_{(3.14)}$ is shown in figure 3.9. It starts from the position indicated by the black dot, with an initially unknown heading. Each heading is kept approximately, using $\sigma_{\text{turn}} = 0.05$. The mean walking distance μ_{walk} is $1.4\,$m with an uncertainty of $\sigma_{\text{walk}} = 0.1$. The three figures denote the resulting density $p(\boldsymbol{q}_t)$ after 1, 3 and 5 consecutive simulations. When no obstacles are blocking the movement, results are similar to figure 3.7. In case of obstacles, walking is prevented, and rooms can only be entered using doors.

How to determine whether there is an obstacle between $\boldsymbol{q}_{t-1}$ and $\boldsymbol{q}_t$, depends on the way the floorplan is modeled. For two dimensional setups, walls could e.g. be given as lines between two points, allowing for a simple geometric intersection test [Ber17; Sch17] of the line between $\mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_{t-1}\right)$ and $\mathbf{pos}_{\text{xy}}\left(\boldsymbol{q}_t\right)$, and each wall from the floorplan. However, depending on the number of required samples, and the number of obstacles within the floorplan, the amount of required intersection tests can be significant. Especially when used on smartphones, realtime usage is limited [Ebn+14].

Besides describing obstacles by lines, there are other options for storing the building's floorplan, such as the *occupancy grid* [Mor88], often used for mobile robots and *simultaneous local-*

ization and mapping (SLAM) [GSB05; Elf89; Thr03]. Here, a discrete grid with many small cells is used to remember whether a location contains an obstacle (is occupied), or not. This allows for quickly testing whether a known location is blocked. Furthermore, it corresponds with sensors measuring distances, often used for mobile robots, such as the LIDAR [Hes+16]. They can be used to populate the grid, and to compare measurements against it. This storage strategy affects the intersection testing between the potential walk from q_{t-1} to q_t and known obstacles. The way from A to B is blocked, if there is one occupied grid cell along it. Performance thus directly depends on the distance between both points, scaling linearly with the number of cells to be checked for occupancy. The provided result is an approximation, depending on the size of each grid cell, used for discretization. Furthermore, depending on the size of the floorplan and the chosen cell size, the grid might require large amounts of memory for storage, demanding for special data structures to reduce this overhead [JGB14].

The grid's concept of locality can be applied to the line-based representation, reducing the number of required intersection tests, by only examining walls that are of interest when moving from q_{t-1} towards q_t. That is, all obstacles which are near the potential walk, and thus might affect it. This question can quickly be answered using data structures such as the *octree* [Mea80] or *k-d tree* [Ben75], both dividing multidimensional space into smaller sections, storing all objects belonging to them. This allows for fast lookups of all obstacles that are near $\mathbf{pos}_{xy}(q_{t-1})$ and $\mathbf{pos}_{xy}(q_t)$, and potentially block this transition. The strategy is also applicable to three dimensional setups, where walls and obstacles are defined as 3D objects. By adding them to a spatial hierarchy, fast spatial lookups are possible. This is well-known from ray tracing in computer graphics, where data structures, such as the *bounding volume hierarchy*, are used to speed up required intersection tests [Shi03].

Nurminen et al. [NRP16] propose a different solution to prevent time consuming intersection tests at runtime, by moving them into an offline phase. Therefore, they divide the walkable area into a grid of $0.5\,\mathrm{m}$ sized cells. For every such cell an intersection test is performed, starting from the cell's center into each possible direction $0°$ to $360°$ using $5°$ steps. For the resulting 72 rays, the distance to the nearest intersection is determined and remembered. This approach yields a database containing the nearest obstacle, depending on the current position and walking direction, with some degree of uncertainty due to the discretization to $0.5\,\mathrm{m}$ in location, and $5°$ in direction. This database not only allows fast lookups whether a potential movement is restricted by architecture, but also favoring movements into more open spaces. As it is less likely for pedestrians to walk near or to approach obstacles, directions with no nearby intersections are more likely to be taken. Therefore, Nurminen et al. converted their database into a PDF, describing the likelihood for walking into some direction, based on the distances of

surrounding obstacles. Due to the $0.5\,\mathrm{m}$ spacing, memory consumption is moderate, and only critical for large multistory buildings.

Though simple in implementation, this approach shows an important drawback. $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ and $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$ are small, when the movement is blocked by an obstacle. Even if there is a door, or other type of entrance, nearby. Thus, this approach will not provide viable predictions, whenever an obstacle can be avoided, or circumvented. Yet, as each detour takes additional time, this flaw becomes mainly relevant when predicting movements for larger timeframes. It is unlikely for a pedestrian to circumvent a wall with a door nearby during one footstep or one second. After five or more seconds, however, chances are much more likely.

Thus, for all following discussions, the timeframe used for the prediction will be considered, as most models work better for small timeframes up to a few seconds. Similarly, the visual representation of a navigation system is more appealing to the user, when updates occur more often. This can be verified by comparing today's car navigation systems with older ones. Recent devices update the location estimation several times per second. Older ones only every one or two seconds. However, while increasing accuracy, more updates require more calculations, and thus easily exceed the limits of embedded computation power and battery life [Ebn+15].

To address the problem of including potential detours within the movement prediction process, a different approach is required. Previous models were based on the assumption of the pedestrian walking straight most of the time. In case of obstacles, this is incorrect. Most likely, the pedestrian uses the shortest way possible to circumvent the obstacle. That is, the shortest distance required to reach any arbitrary location from the starting point must be considered. This still matches with the previous discussions, as walking straight yields the shortest distance towards a target location. Potential whereabouts after a certain timeframe are given by locations, where expected walking distance μ_{walk}, and the smallest distance $\mathrm{dist}^*_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)$ required for reaching, are similar:

$$p_{\mathrm{dist}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}\right) = \mathcal{N}\left(d^* \mid \mu_{\mathrm{walk}}, \sigma^2_{\mathrm{walk}}\right), \quad d^* = \mathrm{dist}^*_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right) \tag{3.15}$$

Within this example, the probability of the pedestrian currently resting instead of walking, and the current heading, are omitted, to focus solely on the impact of detours. To determine the shortest path $\mathrm{dist}^*_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)$ between two locations $\boldsymbol{q}_{t-1}$ and $\boldsymbol{q}_t$, algorithms such as *Dijkstra* [Dij59], or its modification A^* [HNR68], can be used. Both require a graph-based data structure, using vertices connected by edges, to derive the shortest possible connection between two vertices, if there is one. To apply this algorithm to navigation indoors, a graph must be built based on the building's floorplan beforehand. Discussed in detail later, vertices are placed throughout the whole walkable area, e.g. using an equidistant spacing as placement pattern, similar to the
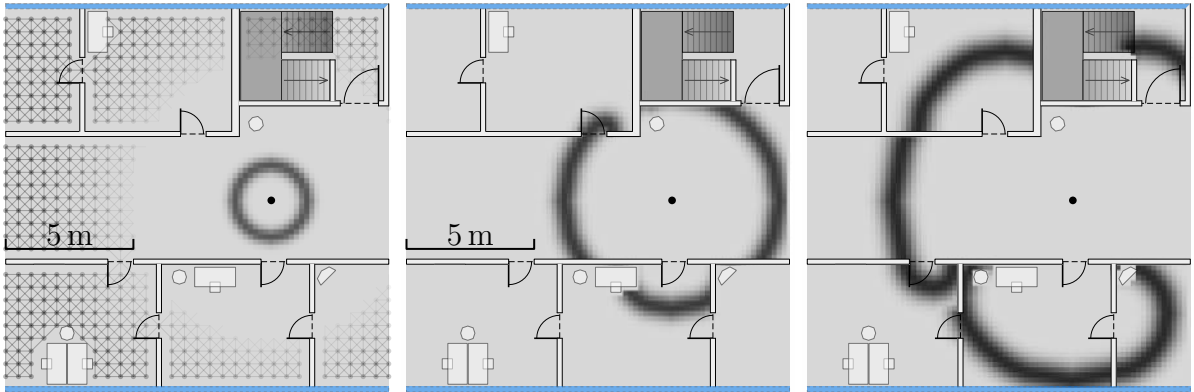
Figure 3.10: Simulation of (3.15) for a $1\,\mathrm{s}$, $3\,\mathrm{s}$ and $5\,\mathrm{s}$ timeframe, assuming the pedestrian to use the shortest way possible to reach a location, starting from the black dot, walking with $\approx 1.4\,\mathrm{m/s}$ into an unknown direction. The shortest path is estimated using a graph based data structure, partially shown on the left. After $1\,\mathrm{s}$, whereabouts are placed along a circle, denoting straight walks from the center. When obstacles prevent straight movements, the shortest detour is used. For the $5\,\mathrm{s}$ timeframe, the circular shape vanishes and is split into several arcs, e.g. after entering a room through a narrow door.

approach presented by [NRP16]. Hereafter, adjacent vertices are connected by a bidirectional edge, if there is no obstacle in between [Ebn+15]. An exemplary fraction of such a graph is shown in the left of figure 3.10. This data structure is able to determine the shortest possible way $\mathrm{dist}^*_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)$ between two locations $\boldsymbol{q}_{t-1}$ and $\boldsymbol{q}_t$ within the building, by following the created edges. As the state $\boldsymbol{q}$ contains a continuous 2D or 3D location, determining the vertices corresponding to $\boldsymbol{q}_{t-1}$ and $\boldsymbol{q}_t$ involves numerical rounding of the coordinates. If there is no direct correspondence in location between the state and a vertex, the nearest vertex can e.g. be used instead. The quality of the returned result thus directly depends on the distance chosen for placing the vertices, where smaller distances, that is, more vertices, yield better results. Again, the simulated timeframe must be considered. For longer timeframes, the approximation imposed by larger vertex distances is barely noticeable. For shorter ones, like $1\,\mathrm{s}$ or $1.4\,\mathrm{m}$, the impact of a larger vertex spacing is significant. This correlation results in a tradeoff between required precision, resulting memory footprint, and computation time [Ebn+16; Hil+14].

A simulated result of (3.15) is shown in figure 3.10, using a $20\,\mathrm{cm}$ distance between the vertices. The left shows an excerpt of the underlying graph, using $50\,\mathrm{cm}$ vertex spacing for visualization. Potential whereabouts, when starting from $\boldsymbol{q}_0$, are simulated for walking durations of $1\,\mathrm{s}$, $3\,\mathrm{s}$ and $5\,\mathrm{s}$. All resulting locations $\boldsymbol{q}_t$ are placed directly on the vertices, that is, all resulting coordinates are multiples of $20\,\mathrm{cm}$. Within $1\,\mathrm{s}$, no obstacles are encountered, and the whereabouts predicted by (3.15) are equal to the ones from (3.12), denoting a circle. For longer simulation timeframes, μ_{walk} and its uncertainty σ_{walk} were adjusted accordingly. As soon as obstacles are encountered, the overall circular shape vanishes, and smaller circles start

to form immediately after narrow passages, such as doors, visible within the room in the lower right corner. Compared to figure 3.9, figure 3.10 indicates potential whereabouts near the ones within the line of sight from the starting point, yielding more realistic results.

One important drawback of this approach is computation time. Mentioned earlier, multiple samples are required to approximate $p_{\text{dist}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}\right)$ by simulation. That is, the shortest path algorithm must be executed for every unique $\boldsymbol{q}_{t-1}$, to determine which $\boldsymbol{q}_t$ are reachable. Depending on the actual implementation, Dijkstra's algorithm has a time complexity between $O(|V| \log |V| + |E|)$ and $O(|V|^2 + |E|)$ for the number $|V|$ of vertices and $|E|$ edges within the graph [Cor09]. Together, this prevents realtime use on embedded devices. Even though the A^* variant is faster, computational requirements are still significant.

Similar to the aforementioned enhancements from Nurminen et al., Köping et al. address this issue by pre-calculating all costly operations once during an offline phase, yielding a data structure that provides efficient lookups [Köp+12; KGD14]. However, a data structure containing the shortest distance from every single vertex towards all other vertices within a building, will consume large amounts of memory, and does not scale for larger floorplans [ARC12].

Furthermore, (3.15) does not consider prior knowledge on the current heading, introduced in (3.12). This absence will yield a similar behavior as shown in figure 3.6. For every consecutive simulation, starting from a previous result, the walking direction is completely randomized, causing unexpected behavior. Yet, it is unclear how to correctly combine the shortest path approach with heuristic heading constraints, such as (3.12). Especially when longer walks are simulated and detours due to obstacles are required, as taking a detour does not match with the assumption of the pedestrian walking straight. For small timeframes, however, a movement prediction based on shortest paths requires a very dense graph-based data structure, to prevent errors, introduced by rounding. Furthermore, when the simulated timeframe is decently small, shortest path calculations can often be completely omitted, as detours rarely occur within short timeframes. The shortest path approach is thus mainly suited for smaller buildings, and for longer simulation timeframes, e.g. stabilized by additional sensor observations.

One important drawback that concerns most models discussed within this section, is their applicability to three dimensional setups. For the 2D case, movements in (x, y) are sufficient. For true three dimensional approaches, a continuous z-coordinate is required as well, to model movements along stairs, escalators and elevators. Assuming the floorplan contains the required semantic information, some sort of data structure is needed for the transition model. It denotes, when the z-component must remain static, due to walking along ground floor, and when it requires adjustment, due to using stairs, escalators or elevators.

The additional z-component also affects intersection tests discussed for (3.14). Simple 2D line-line intersections will not work, when the pedestrian is taking a stair, escalator or elevator,

due to insufficient information on the third dimension. For most floors, the (x, y) coordinate of a 3D setup can be assumed to be unique, as walking below stairs is rarely possible or required, and typical stairs reach the next floor with $\leq 180°$ turns, that is, the stair does not overlap itself in (x, y) within the same floor. While the discussed shortest paths, based on a grid, are close to being suited for 3D, they need adjustments as well, to properly include the additional z-component, and to correctly model potential and invalid movements in all three dimensions.

To prevent requiring large amounts of memory for 3D representations, many approaches found in literature revert to a tradeoff, keeping each floor continuous in (x, y), but using discrete values for z, simply representing the current floor level [GF06; Hil+14; WH08]. Those 2.5D approaches allow for using all of the techniques discussed above, and discretely change to another floorlevel, when needed. The current level can be determined e.g. by using current barometer readings [EBS16; Ebn+15], or Wi-Fi measurements [Zha+18b; Ebn+17], representing a discretization of the approaches discussed in chapter 2. While this is a viable solution in terms of data structures and implementation, user experience is reduced [Ebn+16].

Referring to user experience, additional aspects are important for indoor localization and navigation. For wheelchair users, semantic information whether some area is passable, or the elevator is suited, represents crucial information. Partially blind pedestrians might benefit from restricting the walkable area, to proactively avoid nearby obstacles for a safer navigation. Within security aware areas, access rights to certain doors or elevators might be viable as well. Again, those requirements clearly denote that a single Gaussian and most other analytical representations are unable to model movement predictions, considering all environmental constraints.

The following sections will focus on movement models allowing for true three dimensional setups with respect to the building's floorplan, including a continuous z-component for stairs, escalators and elevators, providing realistic predictions with support for prior information, such as the current heading, still being computationally efficient, suited for smartphone use.

3.4   Overview on Spatial Models for Indoor Floorplans

To calculate realistic movement predictions within indoor environments, the building's floorplan must be considered in some way [AY12; Fro+13; Ebn+14]. Literature provides an extensive list of strategies for creating required *spatial models*, supporting for different types of lookups needed for localization and navigation. A brief example is shown in figure 3.11. Afyouni et al. [ARC12] and Yanbing [Yan06] conducted a survey on different options, mainly related to the field of *geographic information systems* (GIS), yet also applicable to indoor use cases. Among all presented variations, two major groups can be identified, either using a *regular* or *irregular* way of representing the underlying architecture.
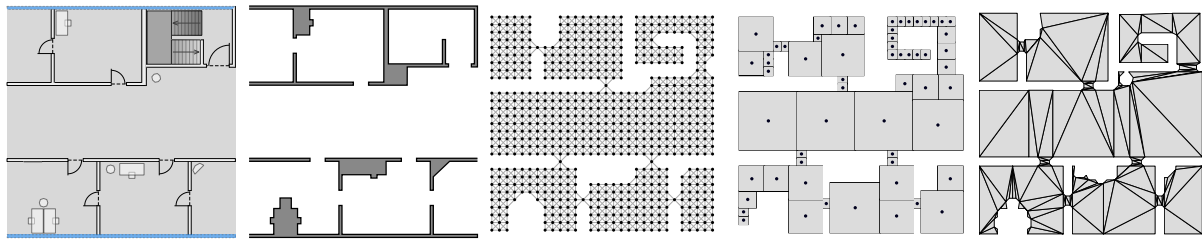
Figure 3.11: Brief overview on various spatial representations for the single floor in the left.

The group of regular patterns can be summarized as a sort of equidistant rasterization of a floor's or building's architecture. Either in 2D or 3D space, dividing the architecture e.g. into several, equally sized rectangles (2D) or cubes (3D). This group is often related to autonomous mobile robots, having to navigate within their surroundings [Sie04; Sch+12]. One simple implementation are 2D or 3D arrays, where each index corresponds with a real-world location. The values stored in such an array e.g. denotes whether an index' location, and its vicinity, belong to free space, or represent a blocking obstacle. This type of representation is thus often referred to as occupancy grid [Mor88]. The size of the array's dimensions depends on the size of the modeled architecture, as well as the required resolution. More entries correspond to more locations with smaller vicinities within real world, and thus a higher spatial resolution.

This information can also be provided by irregular patterns. However, when e.g. using dynamically sized rectangles or cubes, it is not directly clear what the best size for each of them might be. Each object should be sized in correspondence with its neighbors, to accurately describe the building without gaps (cf. figure 3.11). Therefore, most irregular approaches refer to other geometrical structures or primitives. One example is known from computer graphics, where 3D objects are often described using differently sized and connected polygons, referred to as meshes [KSS17]. Applied to architectural use cases, the walkable area can e.g. be *tesselated*, converting it into arbitrary geometric objects, connected to denote the surface [Led06]. Size, shape and placement of those geometric objects strongly depend on the chosen algorithm. For most real-world architecture, this results in irregular sizing and placement, to correctly describe local architectural details [Wu10]. It can be thought of as describing indoor environments using a mesh of connected primitives, such as triangles or polygons. Each primitive denotes a part of the walkable surface, such as a floor or stair. The irregular nature allows for an accurate representation of wide open areas, as well as narrow passages, such as doors, stairwells or elevators [Fet+18]. In contrast to cubes, these meshes describe a surface, instead of a 3D volume. However, for indoor localization and navigation this is sufficient, as the pedestrian can only walk along ground. Whereas underwater or mid-air use cases require the whole volume to be modeled.

Concerning indoor localization, navigation, and movement prediction models, both groups come with benefits and drawbacks that relate to lookup strategies for different types of queries, memory requirements affecting embedded use, and accuracy of the architectural representation. [Wu10] mentions different data structures and options, including underlying storage strategy for efficient lookups, that can be used for regular or irregular representations. Similarly, Yanbing [Yan06] provides a list of various real-world examples using different models and types of geometrical shapes for their respective use cases, also with corresponding data structures, used to improve lookups, as well as resulting advantages and disadvantages of each approach. Besides the already mentioned requirements, there can be additional constraints, dependent on the actual use case, such as semantic information to support navigation for the visually impaired, or wheelchair users. Afyouni et al. [ARC12] list several requirements concerning spatial models for indoor environments in general, and give an extensive overview on potential candidates for different groups of models, dependent on the use case. They also mention the importance of realtime capabilities, support for semantic information, the tightly coupled memory concerns, and maintenance options, often requiring a compromise between all aspects.

Independent of the chosen representation, the to-be-modeled floorplan must be provided, which is a broad field of research on its own. It e.g. covers reconstruction based on panorama images [CF14], or an automatic conversion of scanned blueprints to semantic vector graphics including rooms, walls, doors and obstacles [Liu+17]. 3D scans from depth cameras [Zha+15], or laser scanners [SCI13], can also be used to estimate geometry and obstacles. For example, by using autonomous mapping-robots equipped with LIDAR scanners to derive a map [Hes+16]. But even crowd-based movement data from pedestrians can be used to estimate rooms, and corridors [AY12]. Within this work, the floorplan is assumed to be a given, including all the required semantic information to derive a corresponding spatial model. Details on a tool-assisted creation of the floorplan from blueprints are omitted for now and briefly discussed later on.

Based on the existing floorplan, the following two sections focus on two different spatial models, viable for representing indoor architecture, compatible with the discussed transition models, and suitable for smartphone use. One of them belongs to the group of regular models, the other one is irregular. Each of which comes with certain advantages and disadvantages, that are tightly coupled to the explicit use case scenario. Both are derived from the floorplan automatically, ensuring a fast setup time for the overall system.

3.5   Regular Spatial Models for 3D Movement Prediction

Afyouni et al. identified *grid-based models* to be one of five groups of geometric models, suitable to spatially model indoor environments. Grids, which belong to the type of *regular* repre-

sentations, are able to accurately describe complex architecture and obstacles, are well suited for different types of lookups, and support adding semantic information. The authors also mention that this type of representation requires high amounts of memory when large buildings are modeled by a dense grid. However, their perception of *grid-based* is mainly related to aforementioned occupancy grids [Mor88], where each floor is stored within something similar to a 2D array or an image, with each pixel's color denoting whether the corresponding real-world location is part of free space, or occupied. Occupancy grids are often created dynamically, e.g. by combining several noisy sensor measurements, to distinguish between free and occupied space [ME85]. Thus, the occupancy information is rarely stored as a binary indication, but as a probability, increasing the memory requirements. Depending on the ratio between one pixel and the real-world area covered by it (resolution), as well as the size and number of floors of the building, memory requirements can be significant. This is also due to the image representation being dense, covering a rectangular area, using one pixel for every location, even if it is not part of the building's area. This effect increases if three dimensional environments are to be modeled, requiring several stacked images per floor. The resulting data structure is referred to as *voxel grid*, as it divides the volume of the building into many small cubes, called voxels. As mentioned, volumes are not required for pedestrian indoor localization, and are better suited for aerial or nautical use cases [DMX10; LJ14]. Concerning pedestrians, only stairs, escalators and elevators provide a walkable connection between adjacent floors. The surrounding areas might not be blocked by obstacles, but are still unreachable, as they reside in mid-air. Within voxel grids, those areas require large amounts of unnecessary data that could safely be omitted.

Furthermore, if the grid is crafted based on an actual, exact blueprint of a building's floorplan, probabilities can be omitted and replaced by discrete free/occupied entries, requiring a single bit for each location, significantly reducing the memory footprint. However, if semantic information is to be added for each location, memory requirements are increased by the amount of this information, again consuming unnecessary data for impassable locations. As discussed, especially for 3D grids, this effect will severely affect the applicability for embedded use.

Sparse storage strategies address this issue. Instead of e.g. dense bitmaps, a *sparse list of walkable locations* can be used, dropping the requirement to store regions that are impassable by pedestrians. For the three dimensional case, this reduces the number of stored elements significantly. Furthermore, when using a sparse data structure, all *occupied* locations can be omitted as well, as it is unimportant whether a location is unreachable due to obstacles, or any other reason. This results in a list of all walkable locations within the building, each with corresponding semantic information, if required. To still provide efficient lookups, like within occupancy grids, an appropriate data structure is required [ARC12; Yan06]. For discrete queries, determining whether some location belongs to the list, and thus the walkable surface, *trees* or *hash-sets*
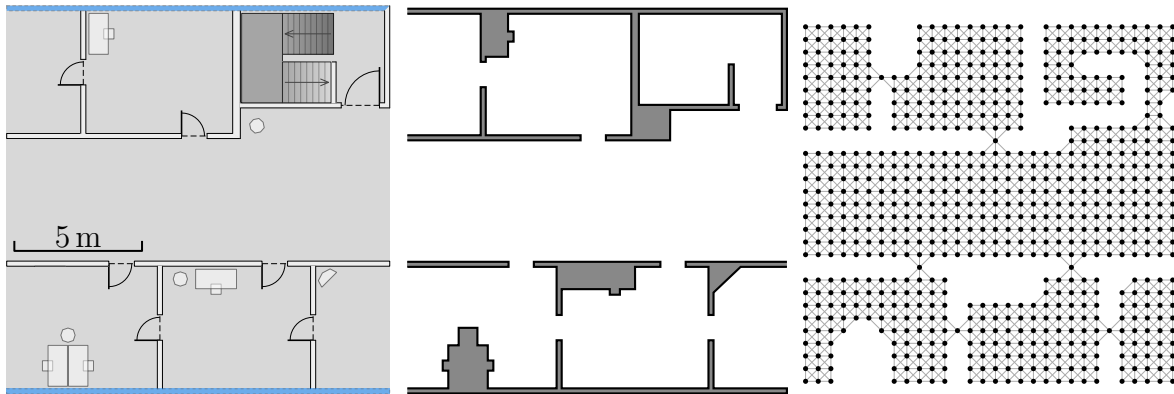
Figure 3.12: Two regular spatial models for an example floorplan (left). The occupancy grid (center) shows walkable areas in white, obstacles in black, and unknown regions, which might either be free or blocked, in gray. The graph on the right describes the walkable area using equidistant (here: $50\,\mathrm{cm}$) vertices (circles), which are connected by edges (lines), if walking between them is physically possible.

[GD18] are a viable choice. For spatial or range-based queries, octrees or k-d trees are a suitable alternative [Mea80; Ben75].

Again, to determine whether walking from one location to another is possible, it must be checked whether both are part of the list, and connected by entries in between. However, the latter information is not yet part of the list-based representation described above. Being sparse, there is no indication whether two walkable locations are adjacent, and reachable from one another. Thus, this connectivity information must be added explicitly. While this increases memory requirements, these connections also allow for adding semantic information concerning the corresponding movement, and provide additional benefits for previous transition models.

The resulting data structure represents a *graph*, with its *vertices* denoting the walkable area, connected by *edges*, if walking between them is physically possible [Ebn+15; Ebn+16]. For the 2D case, it will consume more memory than dense occupancy grids, due to additionally storing individual edges. For the 3D case, however, memory consumption can be assumed to be less than for a dense voxel grid, due to the sparse nature of the graph. Actual requirements strongly depend on the building's architecture, the amount of semantic information to be stored, and the needed resolution, which is given by the distance between two adjacent vertices. However, besides memory overhead, the described representation offers several benefits, that are especially useful for indoor localization and navigation, discussed within the following.

Figure 3.12 depicts a comparison between a dense occupancy grid and a sparse graph-based representation for an example floorplan. The grid stores a probability for each pixel being free (white), occupied (black) or unknown (gray). The graph places vertices only along the actually walkable area, connecting adjacent ones by edges. This section focuses on the graph-based representation of building floorplans, as they are suited for indoor localization on smartphones.

The first part examines the detailed representation of the graph and how to derive it from various types of existing blueprints or floorplans. Hereafter, actual movement prediction models $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$ are discussed in detail, referring to the requirements and issues discussed for previous transition models.

3.5.1 Generation Based on an Existing Floorplan

To use a graph-based data structure as transition model, it must be created based on the buildings architecture. The graph $G = (V, E)$ uses vertices $v \in V$ to denote all the reachable locations within the building. The edges $e \in E$ model the reachability between two vertices. Each vertex v_i stores at least its own 3D position within the building, and optionally, additional semantic information, thus $v_i = \{x, y, z, \ldots\}$. Two vertices v_i and v_j are connected if there is an edge $e_{i,j} \in E$, which is also able to store semantic information $e_{i,j} = \{\ldots\}$. Differing from typical notations for graphs, like the ones used in [Tit11; Aig15], edges do not contain their starting and ending vertex, due to the way the data structure will be kept in memory. All required steps and the final result are shown in figure 3.13, and will now be discussed in detail.

Within this work, deriving the graph requires the floorplan to be readily available, and to contain all required semantic information. Concerning a single floor, this e.g. refers to knowledge on whether some location is walkable, and if the line of sight between two locations is blocked by an obstacle. The described creation process is independent of the format the floorplan is provided with, as long as it contains the required information. Potential formats cover scans from blueprints, 2D vector representations, or complex 3D meshes of the building.

Using this information, the first step places vertices throughout the walkable ground of each single floor. Depending on the floorplan's representation, this area is e.g. given by a connected group of white pixels within a scanned blueprint. Within this step, only the two dimensions x and y are considered, and three dimensional architectural objects within each floor are omitted. The z-value of all newly created vertices is assumed constant, and equal to the height the current floor resides at, e.g. relative to the building's main entrance.

Concerning the vertex placement strategy, the two previously presented options are applicable. Either using a *regular* placement pattern, similar to the occupancy grid, or using an *irregular* pattern, adjusting to the shape of the architecture. The latter can provide a better approximation of the underlying architecture, e.g. in case of curves or other non-axis-aligned architectural components [Hil+14]. A regular pattern is deterministic, as potentially walkable locations are well defined by the pattern itself, and can thus easily be retrieved from the data structure [Ebn+15]. The following discussions thus focus on the regular placement only.
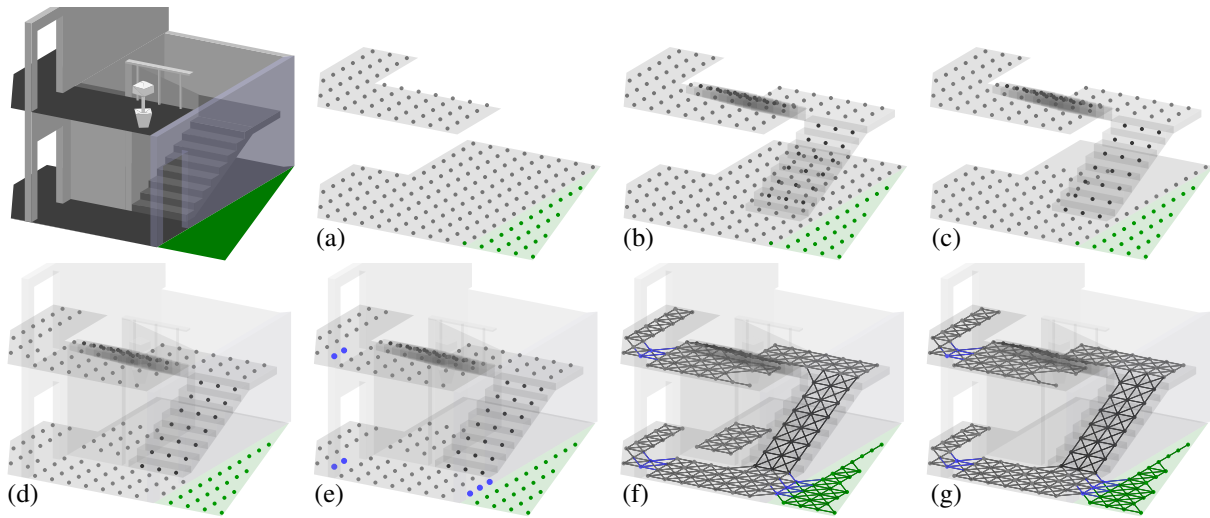
Figure 3.13: Individual steps to derive a walkable 3D graph based on an existing floorplan: Add vertices along every floor using (3.16) with gs $= 50\,\mathrm{cm}$ (a), add vertices along stairs, escalators, and elevators (b), remove impassable vertices above and below stairs (c), remove vertices blocked by obstacles (d), add semantic information (e.g. doors) (e), three-dimensionally connect adjacent vertices using (3.19) if they are physically connected (f), remove small isolated areas (g).

All vertices are placed with a constant distance in between, referred to as grid size gs

$$\forall v \in V : v^{(x)}, v^{(y)} = n\,\mathrm{gs}\,, \quad n \in \mathbb{N}\,. \tag{3.16}$$

Due to this equidistant spacing, each vertex denotes a rectangle, centered at the vertex' position $(v^{(x)}, v^{(y)})$, and its width and height equal to gs, referred to as *navigation grid*.

Depending on the use case, semantic information, whether a vertex belongs to a specific region, can be valuable. Vertices near doors can e.g. be handled differently during movement predictions [Ebn+16]. Similarly, the information whether a vertex belongs to an outdoor or indoor region can e.g. be used to determine whether the GPS is trustworthy at its location [Ebn+17]. If this information is not given by the floorplan, it can sometimes be estimated based on the created navigation grid. Whether a vertex belongs to a door can e.g. be determined by considering all vertices within the vicinity that were *omitted* due to obstacles. If such omitted vertices exist, their locations can be used to calculate a 2D covariance matrix. When the two corresponding eigenvalues greatly differ in size, and the covariance is rather elliptic than circular, a narrow passage, thus potentially a door, has been found [Ebn+16].

Hereafter, adjacent vertices are connected by edges, to model all physically possible moves. Visualized in figure 3.13, depending on the vertex placement, and the floorplan, there can be situations where adjacent vertices are not reachable from one another, separated by an obstacle. In such cases, not placing an edge between both denotes their physical separation. Due to the

placement pattern (3.16), only eight potential neighbors need to be considered for potential connections. The locations of all eight neighbors are well defined. If such a neighbor exists

$$\forall e_{i,j} \in E : (i \neq j) \wedge (v_i \in V \wedge v_j \in V) \wedge (v_j^{(x)} = v_i^{(x)} + d) \wedge (v_j^{(y)} = v_i^{(y)} + d)$$
$$d \in \{- \text{gs}, 0, + \text{gs}\} \ , \tag{3.17}$$

and there is no obstacle in between, the vertices v_i and v_j are connected by an edge $e_{i,j}$. This algorithm yields a graph for one floor of the building. Individual floors are yet to be connected.

The simplest solution just needs the information on starting and ending positions of stairs, escalators and elevators, and directly adds a single edge between them. Such approaches are often referred to as 2.5D, as changes along the z-axis are discontinuous, and based on discrete floor levels [EBS16; GF06]. While working reasonably well for straight stairs, escalators and elevators, it represents a harsh approximation for stairs that include turns (see figure 3.13), present within the stairwells of most larger buildings. Furthermore, this discretization imposes issues during the movement prediction. The single edge has a long distance between the starting and ending point, and strongly deviates from the regular gs. If potential movements are evaluated *per-edge*, and constrained by sensor readings (see chapter 2), model and the sensor's observations might not match. A barometer, for example, denotes continuous pressure changes while walking upstairs. The single edge between two floors, however, denotes an abrupt and large change in z. An evaluation of the barometer's indicated pressures changes does not match with moving directly from one floor to another, and floor changes can be missed [Ebn+15].

True 3D connections, as used in figure 3.13, thus represent the desired way of modeling, to accurately resemble actual architecture, and to match with expected sensor readings. In case of straight stairs, escalators, and elevators, this is e.g. accomplished by replacing aforementioned single edge with multiple vertices and edges along its way, using interpolation. Every new edge represents a small segment, and thus a small change in z. For complex stairs, e.g. including turns, more sophisticated approaches are needed, and the provided floorplan must contain semantic information on size, shape and inclination of individual stair segments. Such information could e.g. be given by describing each stair as 3D mesh. Here, the stair's vertices can be created through samples from the mesh's surface by using inclusion checks [NS80; Gla90], and are simultaneously satisfying (3.16). For typical stairs in buildings, with all vertices' (x, y) pairs being unique throughout a single floor, rasterization can be reduced to a 2D problem. First, the z-component of all stair primitives is omitted, sampling 2D raster points in the (x, y) plane, similar to the previously discussed rasterization of the floor itself. Hereafter, the omitted z-values are re-calculated via interpolation. This approach is well known from computer graphics, where barycentric coordinates are used for arbitrary interpolations within triangles [Vin17]. A similar approach is suitable for quadrilaterals, where a *bilinear interpolation* can be applied

after projecting the quad to a unit rectangle, using an affine transformation [SGM14]. In case of polygons, a prior triangulation is required [Ber+08]. The described process will often create stair vertices having the same (x, y, z) as an existing floor vertex. This is addressed by simply omitting duplicates

$$\forall v_i, v_j \in V : (v_i^{(x)} \neq v_j^{(x)}) \vee (v_i^{(y)} \neq v_j^{(y)}) \vee (v_i^{(z)} \neq v_j^{(z)}) . \tag{3.18}$$

The described creation process can be thought of projecting the whole stair onto the 2D plane of the floor it starts at, rasterizing vertices (x, y) within the area of the projected stair, hereafter undoing the projection by interpolating the correct z-value. Whether the resulting z-values need rounding to multiples of gs to match the regular pattern from (3.16), is not directly clear. Stairs come with various inclinations that are usually well below $45°$ to be walkable [RRF02]. A potential gs_z for z should thus be smaller than the gs used for x and y, to allow for such angles, and a smooth linear z-transition between adjacent vertices. Furthermore, for stairs, the z-value is the result of undoing the projection. Increasing or decreasing a potential gs_z will not alter the number of vertices required for each stair, but only the smoothness of their placement. It thus makes sense to keep the z-value as-is, and only round it within the data-structure used for lookups. An actual grid spacing gs_z for z is only important for elevators, where the vertical placement requires sampling along the z-axis [Ebn+15]. Yet, this neither affects the amount of required memory to hold the graph data structure, as there will not be many elevator vertices, nor will the accuracy of elevators increase when choosing values $\text{gs}_z \ll \text{gs}$.

To ensure proper modeling of stairwells, the algorithm hereafter removes all vertices that are directly above or below a stair. That is, vertices sharing the same (x, y) as a stair vertex, with their distance in z below a certain threshold. Put another way, this step removes impassable vertices above or below stairs, where the pedestrian would e.g. have to crouch, in order to fit beneath. This deletion is shown in figure 3.13c.

Typical stairs, with one plateau in between, consist of two parts. One with treads, and one without, parallel to the ground (cf. figure 3.13). Mentioned earlier, for both, the pedestrian's walking speed will be different. The same holds true for elevators and escalators, where the pedestrian might move without necessarily walking. To be considered during movement predictions, the vertices must carry appropriate semantic information.

To finally connect all floors, (3.17) is altered, in order to also search for neighbors that are slightly above or below. In doing so, vertices related to stairs, escalators, and elevators are connected as well as the ground floor

$$\begin{aligned} \forall e_{i,j} \in E : (i \neq j) \wedge (v_i \in V \wedge v_j \in V) \\ \wedge (v_j^{(x)} = v_i^{(x)} + d) \wedge (v_j^{(y)} = v_i^{(y)} + d) \wedge (v_j^{(z)} = v_i^{(z)} + d) . \end{aligned} \tag{3.19}$$

(3.19) yields a maximum of 26 connections per vertex, nine above, nine below, and eight directly around it. However, for real-world setups, ten neighbors are sufficient, e.g. for elevators, where all eight (x, y) neighbors are present, and two additional, for moving up and down.

Based on the building's architecture, the described algorithm can cause small groups of adjacent vertices that are disconnected from the vast majority of the others. Such isolated regions can safely be removed, to reduce the amount of required memory. This is performed by searching for the largest connected set, and deleting all vertices and edges that are not part of it. Such a disconnected group, and its deletion, is shown in figure 3.13f.

After the algorithm used for creating, the navigation grid's data storage is briefly discussed. As mentioned, vertices are stored within a sequential list, where each vertex can uniquely be identified by its index. Besides variable semantic information, each vertex contains, at least, its position (x, y, z) within the building. Due to the regular, discrete placement, this position can e.g. be stored as integer value in cm. Even when the grid size is as small as $\mathrm{gs} = 1\,\mathrm{cm}$, $16\,\mathrm{bit}$ are sufficient for $2^{16}\,\mathrm{cm} \approx 650\,\mathrm{m}$ in each dimension. A *hash-map* provides fast lookups, using a combination of x, y, and z as hash value, and returns the vertex' index within the list.

For directed graphs without additional semantic information, edges are typically modeled by an *adjacency matrix* [Tit11; Aig15]. This is a square matrix, where each cell denotes the number of edges between the vertex identified by the row index towards the vertex identified by the column index. For the presented use case, this would yield a large matrix with many entries equal to zero, and thus is impractical. As vertices are uniquely identified by their index, each vertex can store a list of indices, pointing to its connected neighbors. While this is typically done using some sort of dynamic data structure [GD18], a fixed-size list is possible as well. Discussed earlier, the largest number of neighbors is assumed to be ten, for elevator vertices. While the smallest number of neighbors is one, most vertices will belong to open spaces, with an average of eight neighbors. Using a fixed-size list for up to ten neighbors, storing their index numbers, thus is a viable choice. Furthermore, this keeps the neighbor-information close to the vertex' details, which allows for caching [BD13]. However, when edges are additionally equipped with semantic information, a dynamic data structure will be more suited, and reduces the memory overhead by not storing unused placeholders.

To summarize, the resulting navigation grid describes the building's walkable area as small rectangles, defined by vertices. They are connected to neighboring tiles, if there is no physical object in between. Due to each vertex holding a list of its adjacent neighbors, they can quickly be identified. As will be shown, this setup allows for various movement prediction strategies, and for incorporating prior knowledge, such as the pedestrians desired destination when navigating through a building.

3.5.2   Random Walks on Graphs

As within section 3.3, the navigation grid is used to estimate the probabilities $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ and $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$ of potential pedestrian movements during a certain timeframe. Initial discussions will focus on $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$, where no additional prior knowledge $\boldsymbol{o}_{t-1}$ is available.

Previous techniques suffered from various drawbacks, such as producing poor results, being computationally complex, or limited to two dimensional predictions only. Based on the created navigation grid, a completely different strategy is used for the movement prediction. Instead of deriving a calculable representation like (3.15), $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ is estimated by simulation, using a technique referred to as *random walks* on graphs [GS97; DS84; Gri18; Tia+02]. In contrast to section 3.3, this approach is computationally efficient, and allows for 3D movement predictions, based on the building's floorplan, discussed within the following.

As with all previous simulations, the starting setup for time $t = 0$ is given by $\boldsymbol{q}_0$. The term $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ denotes one simulation of a certain timeframe, given by two points in time, $t - 1$ and t. Each random walk is initialized by looking up the vertex corresponding to its start, denoted by $\boldsymbol{q}_{t-1}$. This is the vertex v_i with its position

$$\mathbf{pos}_{\mathrm{xyz}}(v_i) = \left(v_i^{(x)}, v_i^{(y)}, v_i^{(z)} \right)^T , \tag{3.20}$$

equal to the position $\mathbf{pos}_{\mathrm{xyz}}(\boldsymbol{q}_{t-1})$ stored within the starting state $\boldsymbol{q}_{t-1}$, after rounding it to multiples of gs. This vertex can be determined instantly, using aforementioned hash-map. The algorithm then *randomly* walks along one of the edges connected to v_i, to reach another vertex v_j. Hereafter, the process continues from the newly reached vertex in the same way. Thus, at every vertex, a new edge is taken by chance. For every such edge, its individual distance

$$||e_{i,j}|| = \mathrm{dist}_{\mathrm{xyz}}(v_i, v_j) = ||\mathbf{pos}_{\mathrm{xyz}}(v_i) - \mathbf{pos}_{\mathrm{xyz}}(v_j)|| \tag{3.21}$$

is added to a cumulated value. The simulation stops, as soon as this value reaches a chosen distance d_{walk}

$$\sum ||e_{i,j}|| \le d_{\mathrm{walk}} . \tag{3.22}$$

The vertex, this simulation ends at, is remembered. As earlier, the probability $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ of reaching $\boldsymbol{q}_t$ from $\boldsymbol{q}_{t-1}$ could be derived by repeating this process several times, creating multiple samples. By applying a KDE, a continuous approximation of this probability is derived. Alternatively, the sampled result could be used to determined how often $\boldsymbol{q}_t$ was reached among all samples, also resulting in aforementioned probability. Similar discussions, using random walks for simulating electric networks, can be found in [DS84]. Which strategy serves best, is discussed within the following.

Performing multiple random walks to derive the probability for a single destination, is similar in complexity to calculating shortest paths. Yet, random walks provide several advantages, like considering additional information, such as semantics attached to vertices and edges. Furthermore, as within section 3.2 and 3.3, the number of simulations required for a stable result strongly depends on the behavior of the underlying density. The same holds true for the timeframe of the prediction. Smaller timeframes result in less scattering, thus narrower densities, and less samples required. The underlying model, assumed for pedestrian movements, affects the size and shape of the density in a similar way: If d_{walk} is the only constraint, and the heading is completely flexible, the resulting density resembles a Gaussian with increasing size, as in figure 3.5. When the model also distinguishes between walking and standing still, the density loses its Gaussian shape, as in figure 3.6. If the pedestrian is assumed to walk straight, potential whereabouts denote a circle, similar to figure 3.7. When the heading is approximately known, only a small fraction of this circle remains likely, as depicted in figure 3.8. As can be seen, depending on available constraints, the number of required samples can be reduced considerably. Due to the navigation grid being a discrete spatial representation, the number of likely vertices is even smaller. Depending on the model and resulting density, down to a few. To be suited for smartphone use, this number should be as low as possible, yet, high enough to accurately describe the resulting density throughout the whole state space [Ebn+16].

Within the following, several movement models with increasing complexity and amount of used prior knowledge are discussed. Regarding strategies, the ideas from section 3.2 and section 3.3 are considered, and applied during the random walk, where edges are picked based on a probabilistic metric. For every single step between two vertices, each outgoing edge $e_{i,j}$ of a source vertex v_i is assigned its own probability $p(e_{i,j} \mid \boldsymbol{q}_{t-1})$, which is equivalent to

$$p(e_{i,j} \mid \boldsymbol{q}_{t-1}) = p(v_j \mid v_i,\, \boldsymbol{q}_{t-1})\,. \tag{3.23}$$

If additional prior knowledge $\boldsymbol{o}_{t-1}$ is available, $p(e_{i,j} \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$ is used instead. After this assignment, one of the up to ten edges is randomly selected, with respect to the calculated probability. Edges with higher probabilities are thus selected more often than ones with a lower probability. Actual algorithms for this kind of biased random selection process are topic of chapter 4, and, for now, assumed to be given.

For comparison, a truly random, and thus unbiased, strategy assumes all outgoing edges of one vertex to have the same, constant probability

$$p_{\mathrm{const}}\left(e_{i,j} \mid \boldsymbol{q}_{t-1}\right) = \frac{1}{|\,\mathrm{neighbors}(v_i)|}\,, \tag{3.24}$$

which depends on the total number $|\operatorname{neighbors}(v_i)|$ for each vertex v_i. This is similar to using a random number generator [HD62], to draw a uniformly distributed random index, deciding which edge (neighbor) to take. After every random selection, the process continues from the newly reached vertex v_j, until the walking distance is reached. As earlier strategies, (3.24) is inaccurate in several ways. It allows for walking back and forth between two vertices, which is unlikely for actual pedestrian movements. Furthermore, the number of potential whereabouts is expected to be fairly large, requiring many samples for a stable result.

According to the findings from section 3.2 and section 3.3, pedestrians tend to keep their walking direction most of the time. The recent heading should thus be considered, when selecting edges within the random process. As earlier, the heading is modeled as part of the unknown state using $q^{(\Theta)}$. Even if the initial heading is unknown, and thus $q_0^{(\Theta)} \sim \mathcal{U}(0, 2\pi)$, it is assumed to be fairly constant throughout a random walk. At least, for short timeframes. In case of existing observations, it can be updated between consecutive transitions, using (3.13)

$$
p_{\text{head}}\left(e_{i,j} \mid \boldsymbol{q}_{t-1}\right) = \mathcal{N}\left(\alpha \mid 0, \sigma_{\text{turn}}^2\right), \quad \alpha = \angle_{\Delta}\left(\angle_{\text{xy}}\left(e_{i,j}\right), q_{t-1}^{(\Theta)}\right)
$$
$$
q_t^{(\Theta)} \sim p_{\text{head}}\left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}\right)_{(3.13)}, \quad \langle \boldsymbol{q}\rangle_t = \langle(\Theta, \ldots)\rangle_t. \tag{3.25}
$$

(3.25) determines a potential edge's 2D angle $\angle_{\text{xy}}\left(e_{i,j}\right)$ in the (x, y) plane, relative to the x-axis

$$
\angle_{\text{xy}}\left(e_{i,j}\right) = \operatorname{atan2}\left(v_j^{(y)} - v_i^{(y)}, v_j^{(x)} - v_i^{(x)}\right), \tag{3.26}
$$

and compares it against $q_{t-1}^{(\Theta)}$. The difference between both should be close to $0°$, thus using a zero mean Gaussian, or comparable von Mises, as PDF. Here, one important aspect must be considered. Due to the regular placement of vertices, and each vertex being connected only to adjacent neighbors, all edge angles $\angle_{\text{xy}}\left(e_{i,j}\right)$ are multiples of $45°$. This would require σ_{turn} from (3.25) to be relatively large, for other edges to be considered at all, which does not match previous findings on straight walks. When using (3.25) as-is, an absolute heading, such as $q_{t-1}^{(\Theta)} = 5°$, can not be modeled at all. For small σ_{turn}, the random process will almost always choose the same edge, in this case, the one pointing towards the positive x axis having $\angle_{\text{xy}}\left(e_{i,j}\right) = 0°$.

This drawback can be addressed by modifying (3.25) to include an error value $\varepsilon_{\text{head}}$ as part of the state, remembering the difference between the desired heading $q_{t-1}^{(\Theta)}$, and every actually walked heading $\angle_{\text{xy}}\left(e_{i,j}\right)$. The angle used for comparison within (3.25) is then adjusted by this error, and approximates the desired angle using zig-zack walks along the navigation grid. This approach resembles Bresenham's line-drawing algorithm [Bre65], known from 2D graphics.

Similar to the angular error, rounding vertex locations to multiples of gs yields an error in walking distance. Adjacent vertices are either gs or $\sqrt{2}$ gs apart. Walking exactly $1.4\,\text{m}$, for ex-

ample, is impossible for $gs = 30\,cm$. This can be addressed in the same way, by storing another error ε_{dist}, carrying the previous overshoot compared to the requested walking distance. Consecutive random walks, continuing from q_t, consider this overshoot, and reduce the requested walking distance. For several walks in sequence, the desired distance is met approximately.

```
1: function WALK(d_walk, q_{t-1})
2:      d' ← q_{t-1}^{(ε_dist)}
3:      ε_head ← q_{t-1}^{(ε_head)}
4:      v_i = FINDVERTEXFORPOS(pos_xyz(q_{t-1}))
5:      while d' < d_walk do
6:          Θ ← q_{t-1}^{(Θ)} + ε_head
7:          v_j ← PICKRANDOMNEIGHBOR(v_i, Θ)
8:          ε_head ← ε_head + ∠_Δ(q_{t-1}^{(Θ)}, ∠_xy(e_{i,j}))
9:          d' ← d' + ||e_{i,j}||
10:         v_i ← v_j
11:     end while
12:     ε_dist ← d' − d_walk
13:     q_t ← (pos_xyz(v_i), q_{t-1}^{(Θ)}, ε_dist, ε_head)
14:     return q_t
15: end function
```

Algorithm 1: Directed random walk, including distance and heading error compensation.

The overall process is best explained using the pseudo code, shown in algorithm 1. First, distance and heading errors from previous runs, if any, are retrieved in 1.2 and 1.3. These values are considered when limiting the walk's length in 1.5, and when randomly selecting an edge in 1.7 that matches the requested heading, after applying the compensation in 1.6. For every taken edge, 1.8 adjusts the heading error by the difference between requested and taken angle. The error in walking distance is adjusted in 1.12, after the walk is completed. The reached location, and both errors, are finally assigned to the returned q_t in 1.13. That is, the errors are available, when another walk is started from this result.

A schematic example of six consecutive random walks using algorithm 1 denotes the impact of distance and heading error compensation, and is shown in figure 3.14. The requested heading was $q_0^{(\Theta)} = 5°$, and the distance $d_{walk} = 1.4\,m$. Most of the time, $0°$ edges are used, as they are closest to $5°$. Due to the cumulating heading error, eventually a $45°$ edge is taken for correction. Similarly, the length of each of the six random walks is adjusted, dependent on the distance error of the previous one. For several consecutive walks, $d_{walk} = 1.4\,m$ is met on average.

The impact of heading error compensation, and varying parameters for σ_{turn}, are shown in figure 3.15. Based on a navigation grid with $gs = 20\,cm$ and no obstacles, four different setups were simulated. Each performed 1000 random walks to estimate $p(q_t \mid q_{t-1})$, starting from q_{t-1}
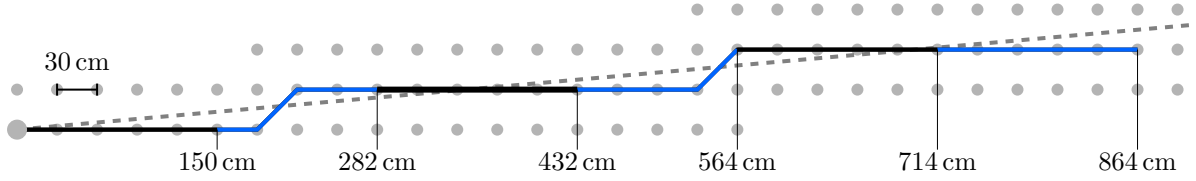
Figure 3.14: Schematic visualization of six consecutive $1.4\,\mathrm{m}$ long random walks into $q_0^{(\Theta)} = 5°$ along a graph using $\mathrm{gs} = 30\,\mathrm{cm}$, algorithm 1 with strategy (3.25) and σ_{turn}^2 near zero. Most of the time, $0°$ edges are taken as they are closest to $5°$. When the cumulated difference between requested and taken angles becomes large, a $45°$ edge is taken instead. The same holds true for the distance. If the previous walk was too long, the next one is shorter, approximating the requested distance throughout several walks.

located in the center, walking into an unknown direction $q_{t-1}^{(\Theta)} = \mathcal{U}(0, 2\pi)$. The dots denote the distinct vertices, reached after the repetitions. Due to the discrete rasterization, the number of different vertices is far below 1000. To visualize a continuous PDF, a KDE is applied to all resulting samples, shown as gray background. When not using the heading error compensation, and enforcing a small $\sigma_{\mathrm{turn}} = 0.01$, as shown in 3.15a, the random walk always picks the direction closest to the initial $q_{t-1}^{(\Theta)}$. As all edges are multiples of $45°$, the result is given by eight distinct vertices, and the corresponding KDE denotes eight smaller blobs. Increasing the allowed heading deviation to $\sigma_{\mathrm{turn}} = 0.1$, the number of distinct vertices, shown in (b), increases rapidly. The KDE, however, barely changes, and is as within (a). This is due to the majority of the resulting samples still being the same. Only a few random walks used other directions, enabled by the increased σ. When the allowed deviation is further increased to $\sigma_{\mathrm{turn}} = 0.5$, shown in (c), the number of distinct vertices advances again. This time, the KDE is affected as well, starting to resemble a circle. Setting the allowed deviation back to $\sigma_{\mathrm{turn}} = 0.1$ from (b), and enabling the heading error compensation, the result of (d) is generated. Here, distinct vertices are as within (b), but the KDE output is similar to (c). This is due to the vertices now being uniformly distributed as they should be, based on $q_{t-1}^{(\Theta)} = \mathcal{U}(0, 2\pi)$.

The main intentions behind the graph-based approach are to avoid costly intersection tests with the floorplan, introduce semantics, and to enable three dimensional setups. For the random walk to be comparable against the continuous variant with floorplan intersection tests from (3.14) and figure 3.9, two changes are required. Pedestrian's can not be expected to walk exactly $1.4\,\mathrm{m/s}$, thus, a slight variation of the walking distance must be included. Furthermore, the pedestrian might not be walking at all, but is currently resting [Ebn+16]. Both adjustments can be included by modifying the walking distance d_{walk}, before every execution of algorithm 1

$$d_{\mathrm{walk}} = \begin{cases} \mathcal{X}_{\mathrm{stand}} & \mathcal{Y} \leq \kappa_{\mathrm{w/s}} \\ \mathcal{X}_{\mathrm{walk}} + d_{\mathrm{walk}} & \mathcal{Y} > \kappa_{\mathrm{w/s}} \end{cases}, \quad \begin{array}{l} \mathcal{X}_{\mathrm{stand}} \sim \mathcal{N}(0, \sigma_{\mathrm{stand}}^2) \\ \mathcal{X}_{\mathrm{walk}} \sim \mathcal{N}(0, \sigma_{\mathrm{walk}}^2) \end{array}, \quad \mathcal{Y} \sim \mathcal{U}(0, 1) . \quad (3.27)$$

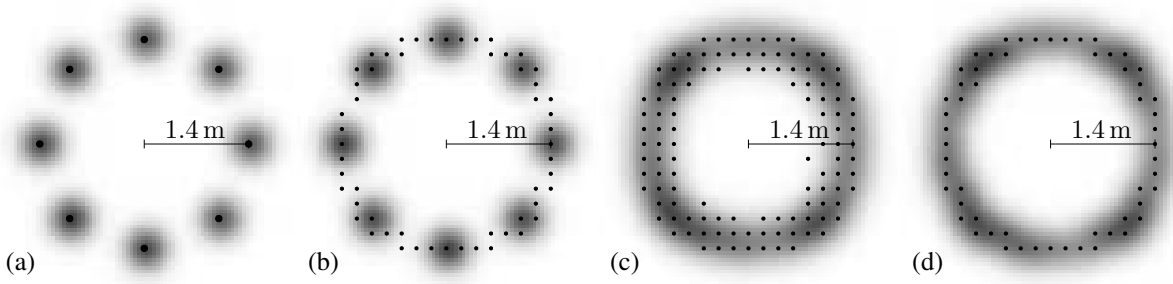Figure 3.15: Result of a random walk (3.25), starting from the center, into an unknown direction $q_{t-1}^{(\Theta)} = \mathcal{U}(0, 2\pi)$. Simulated 1000 times, and with four different parameter sets, along a graph of gs $= 20\,\mathrm{cm}$ with no obstacles. The first three results do not use heading error compensation, and $\sigma_{\mathrm{turn}} = 0.01$ (a), $\sigma_{\mathrm{turn}} = 0.1$ (b), $\sigma_{\mathrm{turn}} = 0.5$ (c). In (d), heading error compensation is used with $\sigma_{\mathrm{turn}} = 0.1$. Dots denote reached target vertices. The gray background is the result of a KDE on top of these vertices.

Both random variables $\mathcal{X}$ are used to adjust the requested walking distance d_{walk} by a zero mean Gaussian with some uncertainty. Additionally, the uniformly distributed $\mathcal{Y}$ and the threshold $\kappa_{\mathrm{w/s}}$ are used to decide, whether walking or standing behavior is used for a random walk. $\kappa_{\mathrm{w/s}} = 0.1$ equals a $10\,\%$ chance of standing, and $90\,\%$ chance of the simulation to be walking. Results after 1, 3 and 5 *consecutive* random walks with algorithm 1 are shown in figure 3.16. As each yields *a single* potential destination, the depicted densities were approximated by 5000 repetitions, generating 5000 potential destinations. The results are comparable with figure 3.9. The three densities are less continuous, and expected circles appear as octagons. Furthermore, the result after 5 consecutive walks looks notably different than the one from figure 3.9. This is due to a disparity between the graph-based simulation, and the intersection tests from section 3.3. For the latter, all movements that crossed an obstacle were completely omitted from the resulting density. For random walks, this is different. As soon as a walk encounters a wall, it has reached a vertex with no neighbors pointing into the desired walking direction. The closest possible direction is chosen instead, letting the random walk move alongside the wall. Due to the heading error compensation, the algorithm alternates between the possible directions, trying to keep the heading. Large obstacles prevent further movements, and the random walk gets stuck in front. The density shown in figure 3.16 thus contains high probabilities along the corridor walls, as many simulations ended there, and could not move on.

To prevent such cases, several options can be taken into account. The walk's uncertainties can e.g. be redrawn, randomly selecting a new σ_{walk} and σ_{turn}, hopefully not encountering the obstacle when restarting with the changed parameters. Or, the result is simply omitted. The latter assumes that it is not possible, or likely, to start walking from $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}\right)$ into the direction given by $q_{t-1}^{(\Theta)}$, as it will encounter an obstacle, a pedestrian would avoid. Further details on strategies and implementations will be topic of the following two chapters.
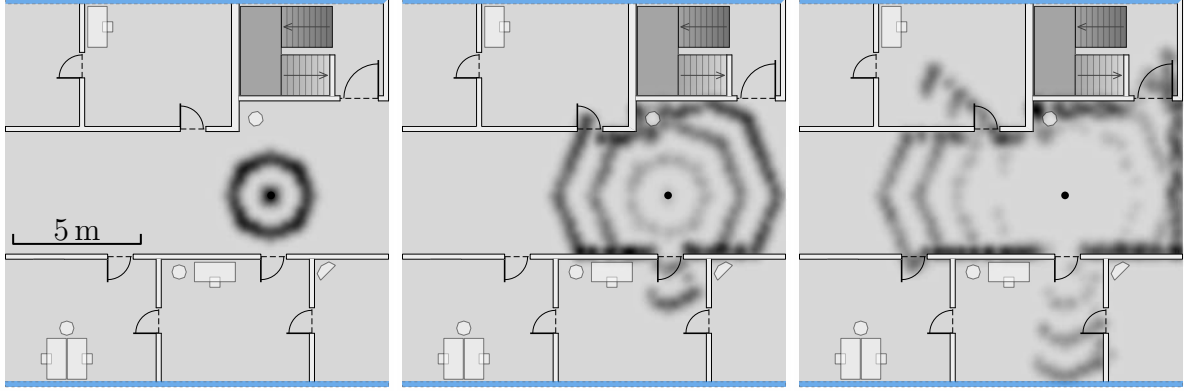
Figure 3.16: Result after 1, 3 and 5 consecutive random walks from the dot in the center using algorithm 1 based on (3.25) and (3.27) with $q_0^{(\Theta)} = \mathcal{U}(0, 2\pi)$, $d_{\text{walk}} = 1.4\,\text{m}$ and $\sigma_{\text{walk}} = 0.1$, approximated by 5000 samples. Darker shadings denote more likely regions. The result is directly comparable against figure 3.9. Due to the regular placement, the density's shape is more octagonal than circular. Furthermore, many simulations stopped at the walls thus yielding a different density than figure 3.9.

By replacing costly intersection tests with random walks, the computational complexity is reduced significantly. Furthermore, additional benefits are enabled. One of which are flexible walking speeds. Mentioned earlier, when taking stairs, the typical pedestrian walking speed of $\approx 1.4\,\text{m/s}$ is reduced significantly, ranging somewhere around $0.55\,\text{m/s}$ [FT04; TG91], dependent on the layout of the stair's treads [SJP13]. The most simple solution uses semantic data, to determine whether the starting vertex belongs to a stair-part with treads (see figure 3.13), and, if so, adjusts the requested walking distance d_{walk} beforehand. This, however, yields issues for larger timeframes, where a single walk covers both, vertices that belong to stairs, and ones that do not. Here, a flexible walking speed is required, achieved by modifying algorithm 1. Instead of cumulating the walking distance by each edge's physical length $||e_{i,j}||$ in 1.9, this value is artificially increased by 2.5, when the edge points upstairs or downstairs. Effectively, this reduces the requested walking distance d_{walk} by a factor of 2.5, whenever taking stairs.

Furthermore, the semantic information on stairs is also well suited for the discussions from section 2.6. Detected activities $o_{t-1}^{(\Omega)}$ can e.g. be used to prioritize certain edges [Fet+16]:

$$p_{\text{act}}\left(e_{i,j} \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}\right) = \begin{cases} \kappa_{\text{act}} & o_{t-1}^{(\Omega)} = \texttt{stair}\uparrow \ \wedge \ \left(v_j^{(z)} - v_i^{(z)}\right) > 0 \\ \kappa_{\text{act}} & o_{t-1}^{(\Omega)} = \texttt{stair}\downarrow \ \wedge \ \left(v_j^{(z)} - v_i^{(z)}\right) < 0 \\ \dots & \\ \left(1 - \kappa_{\text{act}}\right) & \text{else}\,. \end{cases} \quad (3.28)$$

(3.28) represents a brief example of favoring edges that match a detected activity, like walking stairs, by using an empiric value $\kappa_{\text{act}} > 0.5$ as importance heuristic. Assuming statistical in-

dependence, (3.28) can be combined with (3.25) by multiplication, improving the movement prediction model by including observed activities. Other observations, such as barometer readings discussed in section 2.5, can be included similarly, favoring edges that match with the readings from the sensors. As can be seen, the random walk allows for a powerful integration of arbitrary metrics, to favor potential movements during the prediction process. Alongside sensor observations, prior knowledge can be included in a similar way.

3.5.3 Navigation

Besides localization, navigation indoors also represents a crucial aspect, especially within large public buildings like airports, hospitals, museums and similar [Fet+18]. As mentioned in section 3.3, graph data structures are ideal for the required shortest path calculations. Previously, they were used to estimate the shortest distance between two locations, to determine the probability for certain movements. With the number of potential locations being numerous and constantly changing, this was inefficient, and computationally too complex. However, concerning navigation, an efficient implementation for smartphone use is possible.

Typically, the user selects a destination, and the system determines the shortest path for reaching it, once. As long as location estimations are stable, and the user follows the calculated route, no additional calculations are required. Based on the discussions from chapter 2, the location estimation can be assumed to often be unstable, and the pedestrian can not always be expected to strictly adhere to a calculated route. Furthermore, the walkable area of a building is more open than e.g. streets for car navigation. To provide a solution that is able to cope with uncertain location estimations, and the pedestrian deviating from the shortest path, while still being suited for smartphone use, the implementation should be different than for typical outdoor navigation systems, which only consider a required fraction of the map [SA11].

After selecting the desired destination $\rho_{\text{dest}} = (x, y, z)$, the corresponding vertex v_{dest} within the navigation grid is determined. Counter-intuitively, this vertex is used as *starting* point for Dijkstra's algorithm [Dij59]. By not specifying a *target*, the algorithm executes until all vertices were visited. The required edge-weight is given by the three dimensional distance (3.21)

$$\text{w}_{\text{dijkstra}}\left(e_{i,j}\right) = \text{w}_{\text{dijkstra}}\left(v_j \mid v_i\right) = \text{dist}_{\text{xyz}}\left(v_i, v_j\right)_{(3.21)}. \qquad (3.29)$$

After this calculation, every single vertex v_i within the navigation grid knows its shortest physical distance $\text{dist}^*_{\text{xyz}}\left(v_i, v_{\text{dest}}\right)$ towards the destination v_{dest} [Ebn+16]. Depending on the building's size and gs, the calculation requires several seconds. However, it is valid as long as the pedestrian's *destination* remains as-is. A corresponding result within a two story example building is shown in figure 3.17. The heat map denotes each vertex' distance towards the desired
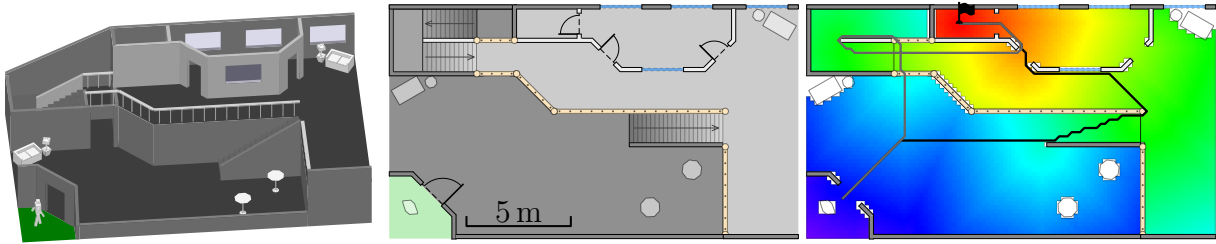
Figure 3.17: Shortest path and distance example within a fraction of a two story building, using gs = 20 cm and (3.29). The requested destination is within a small room on the second floor, denoted by a flag. A heat map visualizes the walking distance from this point towards all other locations within the building. Warm colors (red) denote a short, and cold colors (blue) a long distance. If the pedestrian were to reside in the lower left corner, two potential routes (black, gray) lead to the destination.

destination, which is within a small room on the upper side, denoted by a flag. Warm colors indicate being near this destination, cold colors are farther away from it. To determine what this means in terms of actual routing, two example paths are visualized. Assuming the pedestrian currently resides at the lower left corner, two stairs are available for reaching the upper floor. The black path, 25 m in length, uses the right stair, and is the shortest path for reaching the destination from the pedestrian's current location. When the right stair is unavailable, e.g. due to construction work, the left one can be used instead, shown by the gray path, 28 m in length. While the presented algorithm estimates the shortest path for every vertex within the building, neither of the two depicted paths is realistic. Both stick unnaturally close to walls and obstacles. To predict realistic pedestrian walking paths, a few modifications are introduced.

To avoid walking near walls and obstacles, exterior vertices must be treated specially during the shortest path calculation. A simple solution is to artificially increase the *weight* between two vertices, if one of both is near an obstacle. When using this adjusted weight within the shortest path algorithm, walks along obstacles are made artificially longer than they physically are. If the increase in distance is longer than the savings from sticking close to walls, resulting paths are farther away from obstacles and more realistic [Ebn+16]. This is achieved by adding a new semantic attribute $v_i^{(\iota)}$ to every vertex, denoting the likelihood for being stepped onto by a pedestrian. To reduce this likelihood for vertices that are close to obstacles, the distance towards the closest obstacle is determined. If this distance is small, an obstacle is nearby, and the vertex is less likely to be considered by the pedestrian. To determine this distance, the vertices themselves can be used. Mentioned earlier, most of the vertices have eight neighbors. If a vertex is directly adjacent to an obstacle, this number is reduced. Thus, all vertices with less than eight neighbors potentially belong to the exterior, close to obstacles. The distance towards the closest obstacle is thus given by the smallest distance towards any vertex with less than eight neighbors. This value can quickly be determined using data structures for *nearest-*
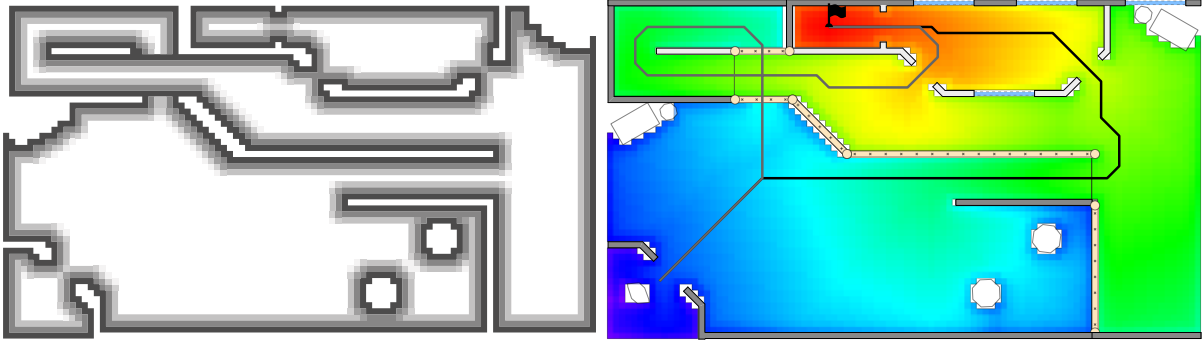
Figure 3.18: Importance factors $v_i^{(\iota)}$ from (3.30), calculated for every vertex, when using $\sigma_{\text{wall}} = 0.5\,\text{m}$ and $\text{gs} = 20\,\text{cm}$ (left). Dark elements denote vertices near obstacles, which have a lower importance. White elements are farther away from obstacles, resulting in an importance of ≈ 1. The right depicts the shortest path and distance calculation towards the black flag, when including the importance factors by using (3.31). Compared to figure 3.17, the two resulting paths are notably more realistic.

neighbor problems [Cla83; Ary+98]. To provide a smooth transition for the likelihood $v_i^{(\iota)}$, a continuous metric is used. It is designed to start at 1, decreasing when approaching an obstacle. $1 - \mathcal{N}\left(d \mid 0, \sigma_{\text{wall}}^2\right)$, with the distance d towards the nearest obstacle, thus is a viable choice

$$v_i^{(\iota)} = 1 - \mathcal{N}\left(d \mid 0, \sigma_{\text{wall}}^2\right)$$
$$d = \min_{v_j} \text{dist}_{\text{xyz}}\left(v_i, v_j\right), \quad v_j \in \left\{v \in V \mid |\text{neighbors}(v)| < 8\right\}. \tag{3.30}$$

(3.30) is calculated only once, after the navigation grid was created. The left half of figure 3.18 visualizes the behavior of $v_i^{(\iota)}$ for every vertex when using $\sigma_{\text{wall}} = 0.5\,\text{m}$ and $\text{gs} = 20\,\text{cm}$. Darker colors denote the adjacency to an obstacle, brightening up with increasing distance. To affect the shortest path calculations, $v_i^{(\iota)}$ is used to adjust the weight metric (3.29)

$$\text{w}_{\text{dijkstra}}\left(e_{i,j}\right) = \text{w}_{\text{dijkstra}}\left(v_j \mid v_i\right) = \frac{1}{v_j^{(\iota)}} \text{dist}_{\text{xyz}}\left(v_i, v_j\right)_{(3.21)}. \tag{3.31}$$

In (3.31), the Euclidean distance between two vertices is multiplied by the reciprocal of the target vertex' $v_j^{(\iota)}$, causing an artificially increased distance when this importance is < 1. The impact depends on the value chosen for σ_{wall} in (3.30). An example for $\sigma_{\text{wall}} = 0.5\,\text{m}$ is shown in the right half of figure 3.18. As can be seen, when approaching walls and other obstacles, the distance towards the destination increases, denoted by the heat map's color getting colder. While still leaving room for further improvements, the two shortest paths from figure 3.17 are now more realistic, avoiding obstacles, and using the center of narrow passages.

Finally, the modified weighting is used to favor all movements that approach the destination

$$p_{\text{dest}}\left(e_{i,j} \mid \boldsymbol{q}_{t-1}\right) = \begin{cases} \kappa_{\text{dest}} & \text{dist}^*_{\text{xyz}}\left(v_j, v_{\text{dest}}\right) < \text{dist}^*_{\text{xyz}}\left(v_i, v_{\text{dest}}\right) \\ \left(1 - \kappa_{\text{dest}}\right) & \text{else}. \end{cases} \tag{3.32}$$

The heuristic κ_{dest} in (3.32) is used to adjust the importance of the destination. While lower values allow for detours, 1.0 enforces every taken edge to approach the requested destination. This represents a tradeoff between prior knowledge, and the behavior of the pedestrian [Ebn+16].

Besides adjusting the weight metric (3.31) to yield more realistic walking paths, the vertex likelihood (3.30) can also be used directly within the random process of algorithm 1, to proactively avoid approaching nearby obstacles. $v_j^{(\iota)}$ can either be included using another threshold heuristic, or directly as probability. Depending on the actually used metric, a normalization η might be required beforehand

$$p_{\text{wall}}\left(e_{i,j} \mid \boldsymbol{q}_{t-1}\right) = \eta v_j^{(\iota)}. \tag{3.33}$$

Using (3.33) within algorithm 1 alongside with other probabilities will yield more natural walks, by avoiding obstacles whenever possible. Yet, it will not fully prevent the walk from getting stuck near obstacles, when there is no other option.

3.5.4 Continuous Results

The main disadvantage of the navigation grid is its discrete nature. By randomly walking from vertex to vertex, as shown in figure 3.19a, directions are limited to multiples of $45°$, and distances to multiples of gs. While impacts are mitigated by the two introduced error metrics, gs must still be reasonably small for individual errors to remain reasonable.

With walks and results directly tied to vertex positions, the approximated density $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ is also rather discrete. This can be mitigated by introducing additional noise. Instead of using the last vertex directly as prediction result, this position can be slightly modified, creating a slightly more continuous output. One option is to e.g. use a location drawn uniformly from the rectangle denoted by this vertex instead [Ebn+17]. This is shown in figure 3.19b, where start and end of the random walk are located somewhere within the rectangle around each vertex. When performing thousands of random walks, this technique does neither affect the walking distance, nor the heading, on average. The distance and heading of individual walks, however, can deviate significantly, depending on the chosen walking distance d_{walk} and vertex spacing gs. This approach thus represents only a minor improvement towards continuous results, and the discussed KDE is a better, yet much more costly, alternative [Bul+18].
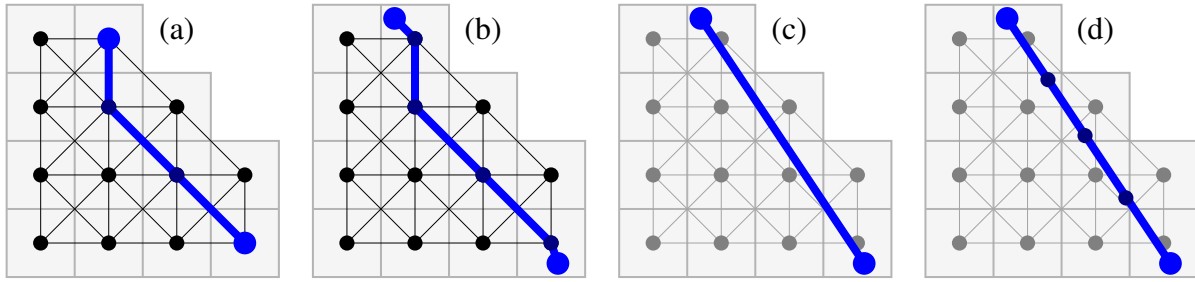
Figure 3.19: Different strategies for walking along a graph-based data structure. (a) matches the traditional approach, walking from vertex to vertex. (b) reduces the discreteness by randomly scattering around the destination vertex. (c) completely omits edge probabilities, just picking the destination based on given distance and heading, when such a location is reachable. (d) is a combination of all three, randomly moving between adjacent cells, without being bound to the vertices themselves.

A solution that satisfies each requested walking distance and heading exactly, but is still computationally efficient and suited for smartphone use, is given by using the continuous wall-intersection approach from section 3.3, but replacing costly intersection tests with cheaper queries on the graph data structure: Given a starting point q_{t-1}, heading $q_{t-1}^{(\Theta)}$, and walking distance d_{walk}, the destination is known directly. This case is shown in figure 3.19c. Whether it is actually reachable from the starting point, and not blocked by obstacles, can be determined by following edges that approximately point into the requested direction, similar to the A^* shortest path algorithm [HNR68]. Compared to the intersection tests from section 3.3, the graph directly allows for fast, locally constrained queries, considering only a few edges that reside within a certain direction and distance. Furthermore, as the query returns the vertex that the requested destination belongs to, if being reachable, it provides a 3D location with known z-component, which was impossible for the 2D intersection tests from section 3.3.

Compared to random walks, there is no uncertainty or variation when using the approach. If reachable, the destination is used directly as-is, based on the requested heading and distance. To add an uncertainty in walking speed and direction, $q_{t-1}^{(\Theta)}$ and d_{walk} must be modified beforehand, e.g. by adding zero mean random variables $\mathcal{N}(0, \sigma_{\text{turn}}^2)$ and $\mathcal{N}(0, \sigma_{\text{walk}}^2)$. The navigation grid is then solely used to replace the costly intersection test with a more efficient solution. Random walks are completely omitted, as well as any semantic information stored on vertices. When directly approaching an obstacle, where random walks got stuck in front, this approach will directly indicate that the requested walk is impossible. As earlier, one of several options, like omitting the current simulation, must be chosen, to address this case.

When modifying this setup, at least, the information whether a vertex belongs to a stair, can be included, to dynamically adjust the walking distance. For that, the algorithm sequentially departs from q_{t-1}, into the direction indicated by the heading $q_{t-1}^{(\Theta)}$. Each step is chosen to reach

the rectangle of the next vertex, if any, shown in figure 3.19d. In doing so, every segment of the walk knows its underlying surface, enabling dynamic walking distance adjustments. However, due to considering only vertices along the path defined by the requested heading, individual probabilities assigned to edges can not be considered. As discussed in section 3.3, while such a direct walking technique yields a continuous result, it is only suited for smaller simulation time-frames, as it relies on line-of-sight connections, where walking around corners is impossible. For longer simulations, the random walk is more suited, supporting fine-grained probabilities, yet producing more discrete results.

Some issues with discreteness can be mitigated by decreasing gs. However, this has a negative effect on computational requirements, as random walks have to follow more edges to reach a given distance, and memory requirements are increasing quadratically. For a $100\,\mathrm{m}$ by $100\,\mathrm{m}$ sized single floor, where $80\,\%$ of the surface is actually walkable, and $20\,\%$ belong to walls and other obstacles, $200\,000$ vertices are required when using a vertex spacing of $\mathrm{gs} = 20\,\mathrm{cm}$. As 3D placement only occurs for stairs, escalators and elevators, this number scales approximately linear with the number of floors in a building

$$|V| \approx 80\,\% \left(\frac{\mathrm{width}}{\mathrm{gs}} \frac{\mathrm{depth}}{\mathrm{gs}} \right) N \,, \quad N = |\mathrm{floors}| \,. \tag{3.34}$$

For a five story building, this results in $1\,\mathrm{M}$ required vertices. To uniquely identify this many vertices, a $24\,\mathrm{bit}$ integer is needed for the numbering index. As discussed earlier, vertices can store their location (x, y, z) using e.g. three $16\,\mathrm{bit}$ integers. Besides location, every vertex has to store a list of ten neighbor indices. All combined, each vertex requires at least $36\,\mathrm{byte}$ of data storage. While today's smartphones carry enough memory to easily handle this example, the situation is different for much larger public buildings, such as airports or hospitals. In case of multiple-compound buildings, connected by large outdoor areas, even more vertices are required, to model their interconnection. When additional semantic details are added to vertices and edges, memory consumption increases even further. At worst, a single map can require several hundred megabytes, becoming impractical for use on smartphones.

According to (3.34), memory concerns can be addressed by increasing gs, at the cost of accuracy. However, starting at $\mathrm{gs} >\approx 60\,\mathrm{cm}$, there is an increasing risk for overlooking parts of the building. When the regular placement pattern does not align with the walkable surface, doors or narrow passages can be missed, excluding everything behind from being added to the navigation grid. Real-world setups thus require more dense grids, ranging near $\mathrm{gs} \approx 25\,\mathrm{cm}$ to provide accurate results [Ebn+15; Ebn+16]. To predict movements within floorplans of huge buildings, or multiple compounds connected by large outdoor areas, simultaneously addressing the issue of discreteness, other data structures are required [Ebn+17].
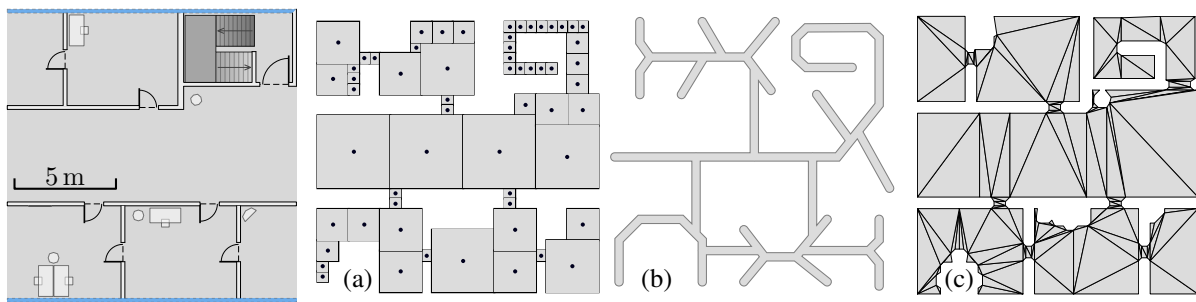
Figure 3.20: Different irregular spatial models for an example floorplan (left). The navigation grid can use irregular sizing and placements, to model the walkable surface using rectangles (a). Instead of modeling the whole walkable surface, only the walkable paths can be described (b). Polygons properly adapt to the shape of surrounding architecture (c).

3.6 Irregular Spatial Models for 3D Movement Prediction

As shown in the surveys from Afyouni et al. [ARC12] and Yanbing [Yan06], irregular placement patterns can address some of the aforementioned problems. They are more flexible, and adjust to local architectural details, allowing for a better representation with reduced memory footprint. Especially when compared against the navigation grid. Referring to the latter, its vertices could e.g. be placed irregularly. Using many small ones to describe stairs, doors, and narrow passages, while approximating open spaces with a few larger ones saves memory, and adds details only where required, shown in figure 3.20a. However, this causes the length of their connecting edges to vary significantly. Likewise, walkable directions become more discrete, not allowing for the random walks as discussed in section 3.5. Also, it is not directly clear how to correctly describe neighborhoods, when a single vertex describes a large region and connects to many vertices with smaller rectangles. Furthermore, rectangular shapes only work well with axis-aligned architecture, like shown in the figure. For buildings with a complex interior, the rectangular primitive suffers from several drawbacks, not representing an ideal data structure [Fet+18].

Besides modeling the walkable surface, some irregular approaches model only the walkable *paths* within a building, shown in figure 3.20b. An often used representative, are minimum *spanning trees* [Kru56] of a 2D *Voronoi diagram* [Lej50; Vor08]. First, the Voronoi diagram tessellates the walkable surface into smaller sections, adjusting to architectural details, such as doors, walls, and other obstacles. This results in an irregular placement of sub-surfaces, represented by polygons (cf. figure 3.21b and 3.21d). The edges in between adjacent polygons can hereafter be connected, forming a tree [Aur91]. A special variant of such a spanning tree is the *generalized Voronoi graph* (GVG) [CB95]. Applied to indoor scenarios, it describes potential routes through the building, similar to streets for car navigation, and can be used to describe potential walks. It is often slightly modified, to remove some extends of the tree, which
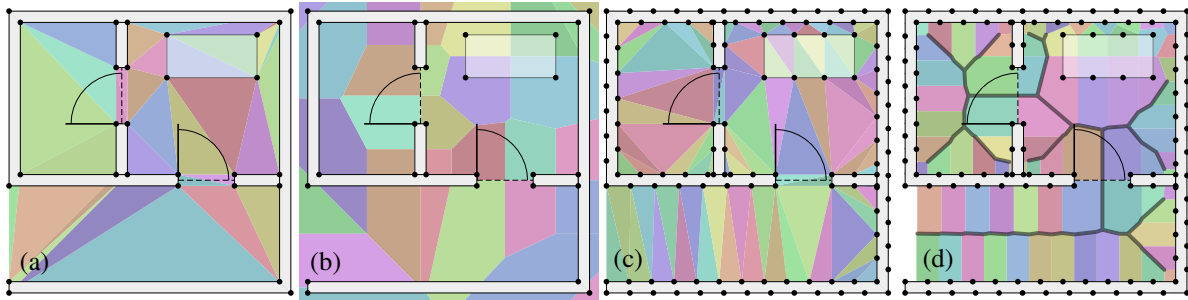
Figure 3.21: Delaunay triangulation and Voronoi diagram for an example floorplan. The corners of walls and other obstacles, shown as black dots, are connected by a Delaunay triangulation (a). This representation can be converted to its dual, the Voronoi diagram (b). Using additional interpolation points along all wall segments increases the quality of both diagrams shown in (c) and (d). The GVG in (d) connects edges of the underlying Voronoi, describing walkable paths within the floorplan.

are considered unlikely, or unnecessary for modeling pedestrian walking behavior. This variant is referred to as *reduced generalized Voronoi graph* (RGVG) [NC99].

Mentioned earlier, another spatial representation are 3D meshes, describing the walkable surface by triangles, or polygons in general, shown in figure 3.20c. A well known algorithm for creating such meshes, based on a given set of edge-points, is the *Delaunay triangulation* [Del34]. Aforementioned Voronoi graphs and Delaunay triangulation are closely related by duality. That is, one representation can be created from the other [DZM07]. Due to this duality, a 3D Voronoi graph can be derived from a triangulated 3D mesh, and vice versa. GVG and RGVG routes thus also work for 3D walking paths within buildings, along stairs, escalators and elevators, even though most of the aforementioned literature covered only 2D or 2.5D setups.

Depending on the way the floorplan is provided, the Voronoi diagram can e.g. be created by first performing a Delaunay triangulation, using the corner points of walls and other obstacles as to-be-triangulated vertices. This is shown in figure 3.21a, where all corner points, shown as black dots, are connected by the Delaunay triangulation, which is hereafter converted to its dual, the Voronoi diagram. This conversion can be performed by estimating a circle for each triangle, covering its three corners. For any two adjacent triangles, the centers of their circles are connected, forming the Voronoi diagram [Joe99]. However, the result in 3.21b does not reveal any walkable paths. This is due to using only a few corners to perform the triangulation, not clearly separating between walkable areas and obstacles. By interpolating additional vertices along each wall segment, the Delaunay triangulation in 3.21c gets more dense. Now, the walkable surfaces can clearly be separated from impassable areas. The latter is also described by triangles, which must be deleted, as they reside within obstacles. Calculating the corresponding Voronoi diagram 3.21d, the GVG is now visible as the spanning tree along the edges of the Voronoi cells [CB95]. The quality of the GVG thus directly depends on the number of

interpolations along obstacle outlines. When the building's floorplan is provided as 2D vector graphics, the Voronoi diagram can be created directly, using individual obstacle *segments*, given as 2D lines [Kar04]. If the floorplan is provided as a triangulated 3D mesh, the corresponding Voronoi and GVG can also be created directly, using aforementioned translation.

The GVG in figure 3.21d denotes walkable routes within the map, which are centered between walls and other obstacles [BJK05]. Resulting from an irregular tessellation, it can efficiently model complex architecture, and requires only a fraction of the memory demanded by a navigation grid. Similar to the latter, the GVG represents a walkable data structure, formed by connecting several nodes. Yet, it models completely different movement predictions, as it does not cover large open spaces. Containing only a single path for traversing every region of the building, it is mainly suited for narrow corridors, shown in figure 3.20b. To also cover scenarios with pedestrians walking through large open spaces, adjustments are required. Hilsenbeck et al. [Hil+14] suggest a mitigation by placing a dense mesh of walkable nodes throughout large open areas. While the result hereafter covers the whole walkable surface, it suffers from the same issues with discrete headings, previously discussed in section 3.5.

GVG und RGVG denote only a fraction of actually possible walks within the building. This has consequences for movement predictions $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ and $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$. In contrast to random walks along the graph, the risk of simulations getting stuck in front of obstacles is reduced, as there are less dead ends. On the other hand, only a fraction of all reachable locations can be predicted by this setup. In theory, algorithm 1, used for the random walk, still applies to GVG. However, cumulative heading errors within (3.25) can not be compensated, as often only two directions, back and forth, are possible. Similarly, the walking distance might not match with the distance between adjacent nodes of the GVG, which can be rather large. As a workaround, the random walk could stop somewhere along an edge, to match the requested d_{walk} exactly by interpolation. The strong limitations on allowed movements can be considered a benefit within narrow surroundings, such as hallways and stairwells. Getting stuck can be prevented, and the binary direction choice can suppress issues with sensor uncertainties. The GVG is also suited for navigation, supporting shortest path calculations, and adding constraints, like in (3.32). Yet, only affecting intersections, navigation constraints might be completely overruled by the heading. Furthermore, even after the adjustments from [Hil+14], the GVG suffers from drawbacks within large open environments, yielding similar issues with discreteness, as the navigation grid. For such regions, another spatial data structure is more suited.
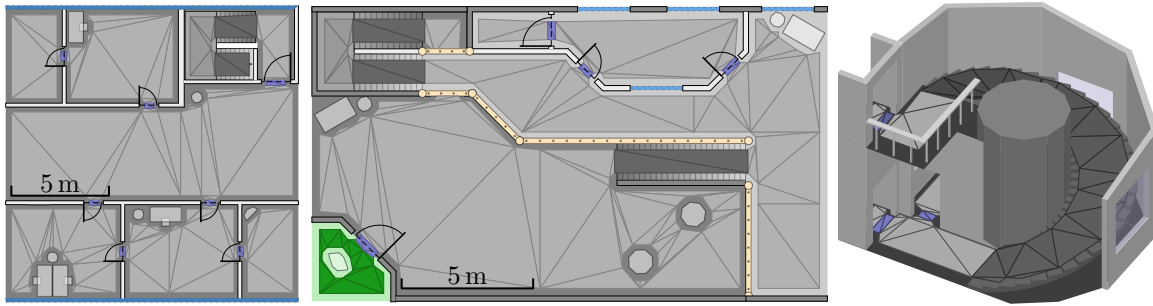
Figure 3.22: Three navigation mesh examples using triangles as primitives, including semantic information on indoor, outdoor, stairs and doors, shown in different colors. The segmentation algorithm ensures that primitives adjust to both, semantic boundaries and the surrounding architecture.

3.6.1   3D Navigation Meshes

Besides forming the dual of the Voronoi diagram, a Delaunay triangulation also represents a 2D or 3D mesh of the building's floorplan. It is created by triangulating the vertices interpolated along obstacle edges (cf. figure 3.21c). When omitting all triangles that would fully reside within obstacles, all remaining ones denote the walkable surface. While this irregular result is not viable for random walks, it can be used in the same way as discussed in section 3.5.4. Shown within the following, it can efficiently determine whether two locations within a building are disconnected by an obstacle, and it allows reconstructing the z-coordinate when changing floors. Compared to the navigation grid, this irregular representation perfectly models surrounding architecture, as it closely adheres to obstacle borders. Simultaneously, it conserves memory, as the triangles are significantly larger than aforementioned $20\,\text{cm}$ by $20\,\text{cm}$ grid cells [Fet+18].

When using a Delaunay triangulation, the average size of the triangles directly depends on the distance between the to-be-connected vertices (see figure 3.21). As the triangulation only considers vertices, it is unable to distinguish between walkable surface and obstacles. Using large distances thus increases the risk of triangles partially belonging to obstacles, shown in figure 3.21a. For the created mesh to be correct, the interpolated vertices are thus required to be rather close, like shown in figure 3.21c. This aspect unnecessarily increases the number of required triangles. Other algorithms besides Delaunay thus are more suited.

The general type of resulting data structure is referred to as *navigation mesh*, or *meadow map* [Ark87]. It is e.g. used in computer games, to model the walkable areas, and to provide navigation [CS11]. While these meshes can be created from various data sources, a 3D model of the building, containing floors, stairs, and objects combined with semantic information, is ideal [Fet+18]. Literature provides various algorithms, such as *watershed segmentation* [VS91; RM00]. They generate primitives which accurately resemble the floorplan, including its semantic information, while trying to combine large regions into a single primitive. In general,

the result describes the walkable surface using polygons. The algorithm ensures that adjacent polygons share one of their edges, which is a crucial requirement for modeling adjacency.

As the result is used to represent the walkable surface, the same requirements as discussed for the navigation grid (cf. section 3.5.1) apply. Walking below stairs or escalators is unlikely, and corresponding surfaces should be removed whenever the pedestrian has to crouch to fit beneath. Similarly, walking close to walls and other obstacles is unlikely, and should therefore be prevented. This can be addressed by eroding the resulting mesh after its creation, thus shrinking the walkable surface by a given constant. To correctly determine pedestrian walking speeds for a certain surface of the floorplan, semantic information must be retained. The polygons thus must be placed in a way that each one belongs to exactly one semantic group, like stairs, escalators or doors. Potential results are presented in figure 3.22. The depicted navigation meshes use triangles as primitives, and adjust to semantic and architectural borders. The walkable region was slightly eroded to ensure that regions directly adjacent to walls and other obstacles are not considered walkable. Compared to the navigation grid, only a few primitives are needed to accurately cover every nook and cranny of the floorplan.

The following focuses on movement predictions $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ and $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$, when the underlying spatial model is an irregular 3D navigation mesh of the walkable surface. While also applying to polygons in general, the generated 3D mesh is assumed to contain only triangles. This is due to the triangle primitive offering unique characteristics, yielding a speed-up for many of the required calculations, which is important for use on smartphones. Using triangles will slightly increase the required amount of memory, as more primitives and vertices are needed. However, the result still consumes only a fraction of the navigation grid [Fet+18].

3.6.2 Movement Prediction

Similar to the navigation grid, the navigation mesh allows for several different approaches, to predict potential movements. It can e.g. be used as a fast look-up, to determine whether two locations are connected, and reachable within a certain distance, providing a speed-up for the discussions from section 3.3. Like in (3.12) and (3.8), potential whereabouts $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ are determined by simulation, using a starting location, a walking direction, and distance, to predict new whereabouts. In contrast to earlier, equations are described from another point of view, directly describing new whereabouts as the result from a random process. Again, the starting point for each simulation is given by $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}\right)$, combined with the walking direction $q_{t-1}^{(\Theta)}$, and distance d_{walk}. The latter can e.g. be set to the average of $1.4\,\mathrm{m/s}$, or set to 0, to model the

case where the pedestrian is resting

$$
\begin{aligned}
q_t^{(x)} &= q_{t-1}^{(x)} + d\cos(q_{t-1}^{(\Theta)}) & d &= d_{\text{walk}} + \mathcal{X}_{\text{walk}} \\
q_t^{(y)} &= q_{t-1}^{(y)} + d\sin(q_{t-1}^{(\Theta)}) & \mathcal{X}_{\text{walk}} &\sim \mathcal{N}(0, \sigma_{\text{walk}}^2)
\end{aligned}
\qquad \langle \boldsymbol{q} \rangle_t = \langle (x, y, z, \Theta, \dots) \rangle_t \ . \quad (3.35)
$$

(3.35) predicts new potential whereabouts in x and y by moving a distance d from the starting position, into a given direction. For now, the z-coordinate $q_t^{(z)}$ is omitted, and will be discussed hereafter. Similar to (3.27), uncertainty in walking speed is given by a zero mean Gaussian $\mathcal{X}_{\text{walk}}$, added to the requested distance. The heading is incorporated based on available prior knowledge. Like earlier, the initial walking direction often is completely unknown

$$
q_0^{(\Theta)} \sim \mathcal{U}(0, 2\pi) \ . \tag{3.36}
$$

This heading is hereafter updated, depending on whether relative, absolute, or no heading observations are available

$$
q_t^{(\Theta)} \sim p_{\text{head}} \left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1} \right)_{(3.13)} . \tag{3.37}
$$

Again, due to the assumption of the pedestrian walking straight throughout a single simulation, this setup is only suited for small timeframes and short walking distances d_{walk}. As within (3.14), the result of (3.35) must only be accepted if $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_t\right)$ can be reached from $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_{t-1}\right)$. This information, and the omitted z-coordinate, are both provided by the navigation mesh.

   To determine whether two locations are reachable within a given distance, the triangles and their adjacency are used. First, the triangle the starting position $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_{t-1}\right)$ belongs to is determined, shown in figure 3.23a. This is achieved by using barycentric coordinates [Vin17], describing the location of a point based on a triangle's edges, hereby denoting whether it resides within, or not. To prevent numerical issues, the z-component is allowed to vary, e.g. by working with 2D triangles in the (x, y) plane, and ensuring the point's z is somewhere within the range of the original triangle's z-coordinates. This is viable, as the created mesh ensures that there is at least a pedestrian's height between triangles stacked along the z-axis. Finding this initial triangle is costly when creating many samples, due to $O(n)$, having to check every single triangle. This is addressed by caching, remembering the current triangle for every newly sampled $\boldsymbol{q}_t$, made clear within the following.

   Having found the initial triangle, its adjacent neighborhood is examined. A fraction from the original navigation mesh is extracted, describing the reachable surface, when starting from $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_{t-1}\right)$, and walking for a certain distance. This surface is created by recursively adding adjacent triangles using breadth-first search [GD18]. That is, first adding all directly adjacent triangles, hereafter adding their adjacencies, ignoring duplicates. The recursion is terminated as soon as a certain distance threshold has been reached, which depends on the to-be-simulated
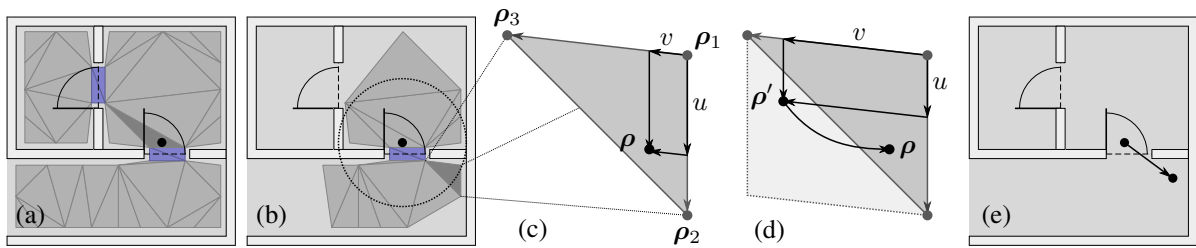
Figure 3.23: Process for uniformly sampling a new location within the vicinity of a starting point (black dot) when using navigation meshes as floorplan. First, the triangle the starting point belongs to is identified (a). Second, a reachable fraction of the navigation mesh is extracted, based on some user-defined threshold. One triangle within this fraction is chosen randomly, with respect to its size (b). Finally, a random point within this triangle is created (c,d), hereafter denoting the destination (e).

walking distance d_{walk}. For short simulation distances d_{walk}, the extracted fraction of the navigation mesh contains only a few triangles, visualized in figure 3.23b.

For $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_t\right)$ to be reachable from $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_{t-1}\right)$, it must belong to one of the triangles within the extracted fraction. As $\boldsymbol{q}_t$ was calculated by (3.35), only x and y are valid for now. Whether the new (x, y) results in a change in z, depends on the floorplan. If start and end of the walk both belong to triangles that are part of the ground floor, the z-component remains unchanged. If either of them e.g. belongs to a stair, the z-component is expected to be different. Thus, to determine whether $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_t\right)$ is part of the extracted fraction of the navigation mesh, only (x, y) is used, examining whether this 2D location lies within any of the extracted triangles, omitting their z-coordinate as well. If one triangle contains this 2D location, the omitted z-coordinate is re-calculated via barycentric interpolation [Vin17]. It reconstructs the z-coordinate based on (x, y), ensuring the resulting 3D coordinate resides on the triangle's surface. Again, omitting z is valid, if the simulated walking distance d_{walk} is small, and triangles are ensured to have a reasonable distance along the z-axis, which is a given for the described navigation mesh. To increase performance, the triangle that $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_t\right)$ belongs to is remembered, preventing costly inclusion checks for subsequent simulations.

If the location determined by (3.35) does not belong to any of the extracted triangles, it is unreachable from the starting position $\mathbf{pos}_{\text{xyz}}\left(\boldsymbol{q}_{t-1}\right)$. Similarly to previous discussions for other movement predictions, there are various options for dealing with such cases. Either by running the simulation again, yielding a slightly different location, due to the random noise for heading and distance, or by omitting this single simulation completely. Yet, a different strategy can be viable as well. Besides handling cases where the randomly determined location was unreachable, it can serve as a complete walking prediction on its own, able to include arbitrary probabilistic constraints, similar to the random walk from section 3.5.2.

According to section 3.2, a movement prediction model without prior knowledge just scatters potential new locations around the initial one. Not constraining the walking direction, and limiting the walking distance within a broad, uniform range. The models in section 3.2 performed this kind of prediction, but were unable to consider the floorplan, and were only suited for 2D setups. Using the navigation mesh, aforementioned model can be constrained by the floorplan, and supports 3D location predictions, by re-calculating the missing z-component.

A predicted $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_t\right)$ is somewhere near its origin $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}\right)$. It can thus be generated by determining the origins triangle, extracting a fraction of the mesh describing the walkable surface around it, and randomly picking a destination that belongs to this walkable area. This yields a new location within the vicinity of $\mathbf{pos}_{\mathrm{xyz}}\left(\boldsymbol{q}_{t-1}\right)$, constrained by the walkable surface, including 3D location information. Here, $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$ is simulated by randomly picking a point on the extracted fraction of the walkable surface. This is achieved by randomly choosing a triangle that is part of the extracted surface, shown in figure 3.23b, hereafter selecting a random point that resides within this triangle, visualized in figure 3.23c and 3.23d.

A random triangle is selected by drawing an uniformly distributed index, e.g. using a random number generator. When performing an infinite number of simulations, this produces an equal number of samples from every triangle. However, as the size of all triangles is irregular, this yields a non-uniform distribution of points with respect to the walkable surface, as samples will concentrate within smaller triangles. Thus, they must not be selected uniformly, but biased based on their surface area, where larger triangles are chosen more often than smaller ones. Each triangle's probability is given by

$$p(\mathrm{triangle}_i) = \eta \, \mathrm{area}(\mathrm{triangle}_i) \ , \tag{3.38}$$

including a normalization constant η. Details on implementing biased random processes are omitted for now, and presented later in chapter 4. As shown within figure 3.23c, every point $\boldsymbol{\rho}$ within a 2D or 3D triangle is then uniquely identified by a linear combination of two edges

$$\boldsymbol{\rho} = \boldsymbol{\rho}_1 + u(\boldsymbol{\rho}_2 - \boldsymbol{\rho}_1) + v(\boldsymbol{\rho}_3 - \boldsymbol{\rho}_1), \quad 0 \le (u + v) \le 1 \,. \tag{3.39}$$

A random point within the triangle can be sampled by creating two uniformly distributed random numbers $u, v \sim \mathcal{U}\left(0, 1\right)$. In case of $(u + v) > 1$, occurring for $50\,\%$ of the samples, the resulting point is outside of the triangle. Simply omitting those cases, and drawing a new one, can affect the output of poor (pseudo-)random number generators. The rejected result can be used when projecting the point back into the triangle, by mirroring it along the edge between $\boldsymbol{\rho}_2$ and $\boldsymbol{\rho}_3$, visualized in figure 3.23d. Analytically this is achieved by

$$\boldsymbol{\rho} = \boldsymbol{\rho}_1 + (1 - u)(\boldsymbol{\rho}_2 - \boldsymbol{\rho}_1) + (1 - v)(\boldsymbol{\rho}_3 - \boldsymbol{\rho}_1), \quad 1 < (u + v) \le 2 \,. \tag{3.40}$$

When using the described approach, potential new whereabouts are given as samples, uniformly distributed throughout the extracted fraction of the navigation mesh, shown in figure 3.24a. This yields a discrete drop to a probability of zero when leaving the fraction, not being realistic. It can be mitigated by applying a KDE to the samples, yielding a continuous probability density function, slowly fading near the boundaries, shown in figure 3.24b.

In section 2.7, the KDE used on top of discrete fingerprints also included a *weight* for each of them, based on their probability. That is, the weight *favored* some fingerprints over others, concentrating a larger part of the resulting density around fingerprints with higher weights. The same idea can be applied to the movement prediction. By calculating a weight $\mathrm{w}(\boldsymbol{q}_t)$ for each sample $\boldsymbol{q}_t$, and applying a KDE afterwards, the uniform behavior is modified. Including weights allows for creating arbitrary, non-uniform distributions, based on additional metrics, such as

$$\mathrm{w}(\boldsymbol{q}_t) = \begin{cases} p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}) & \text{transition probability without observations} \\ p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) & \text{transition probability with observation}\,, \end{cases} \tag{3.41}$$

hereafter being as powerful as the random walks. The behavior of (3.35) can e.g. be approximated by uniformly sampling multiple locations $\boldsymbol{q}_t$ that are reachable from $\boldsymbol{q}_{t-1}$, and weighting each one according to whether it matches the heading $q_{t-1}^{(\Theta)}$, and walking distance d_{walk}, including an uncertainty

$$\mathrm{w}(\boldsymbol{q}_t) = p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1}) = \mathcal{N}\left(\alpha \mid 0, \sigma_{\mathrm{turn}}^2\right)\mathcal{N}\left(d - d_{\mathrm{walk}} \mid 0, \sigma_{\mathrm{walk}}^2\right)$$
$$\alpha = \angle_\Delta\left(\angle_{\mathrm{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right), q_{t-1}^{(\Theta)}\right)\,, \quad d = \mathrm{dist}_{\mathrm{xy}}\left(\boldsymbol{q}_{t-1}, \boldsymbol{q}_t\right)\,, \quad \langle\boldsymbol{q}\rangle_t = \langle(x, y, \Theta, \ldots)\rangle_t\,. \tag{3.42}$$

An example of random sampling with and without weights is shown in figure 3.24. The first two figures depict the unweighted uniform sampling from a fraction of the navigation mesh, including a $3.5\,\mathrm{m}$ radius around the start $\boldsymbol{q}_{t-1}$. The corresponding KDE in figure 3.24b shows the uniform probability, fading to zero towards the borders. When adding weights (3.42), all samples with matching distance and heading are more important than others, visualized in figure 3.24c. The corresponding KDE in figure 3.24d denotes a density, shaped as fraction of a circle, where both, the requested heading and distance, are correct. This solution matches with (3.12) and (3.14), yet, including the building's floorplan, and supporting three dimensions.

Apart from the example (3.42), this approach yields room for an infinite number of metrics. Almost all of the densities discussed in chapter 2 can be applied as well, to combine a uniform floorplan-based movement prediction with sensor observations. By adjusting weights, an overall density is derived, that combines all individual viewpoints.
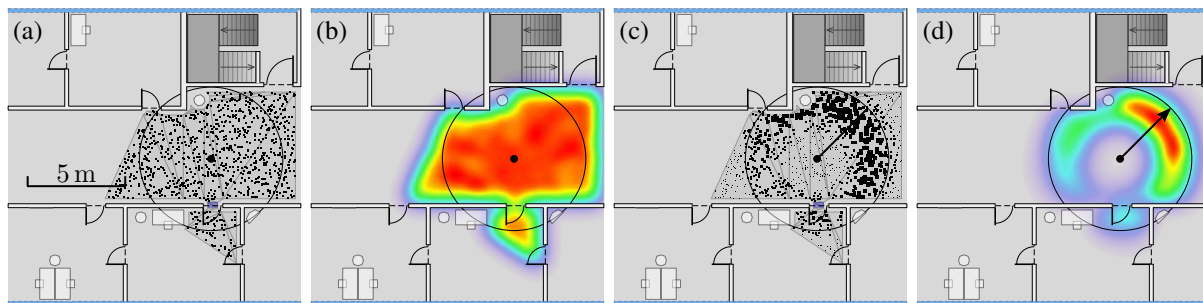
Figure 3.24: 1000 unweighted (a) and weighted (c) random samples from a fraction of the floorplan's navigation mesh, within a $3.5\,\mathrm{m}$ radius around the start $\boldsymbol{q}_{t-1}$ (black dot). The corresponding KDEs are shown as heat maps in (b) and (d). Weights used in (c) and (d) are based on (3.42), with $d_{\mathrm{walk}} = 2.8\,\mathrm{m}$ and $\sigma_{\mathrm{turn}} = 0.3$, assuming the pedestrian's heading to be $q_{t-1}^{(\Theta)} = 45°$.

Besides being versatile, this approach is often impractical for use on smartphones. As shown in figure 3.24, numerous random samples are required, to derive new whereabouts, when starting from a single origin. In contrast, (3.42) requires just a few calculations to provide a movement prediction. As discussed earlier, if the uncertainties for σ_{walk} and σ_{turn} are small, a single prediction should be sufficient to approximate $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1})$.

However, this approach can e.g. be used for situations where the direct prediction (3.35) is unable to provide a result due to obstacles. A result is then obtained by requesting one uniform sample with additional weighting, denoting how well the uniformly sampled location matches the initially requested heading and destination. Furthermore, the bottleneck solely stems from the required uniform sampling, to ensure some samples with a decent weight are present. The weighting process itself is efficient, and weights can e.g. be used to combine (3.35) with navigation information, favoring results that approach the pedestrian's destination.

3.6.3 Navigation

As briefly mentioned for the GVG, irregular spatial models are also suited for navigation purposes. The internal data structure of a car navigation system is similar to the way the GVG models walkable routes within the building. It describes streets and intersections by edges and vertices, placed at arbitrary, irregular positions. As streets are rather narrow, they are well-approximated by edges with a certain thickness. However, for large open spaces, often encountered indoors, a single edge does not provide a viable representation. It reduces the walkable surfaces to a narrow channel, preventing the pedestrian from reaching any arbitrary location (cf. figure 3.20). The previously presented navigation grid suffers from the same issue, but mitigated it, by placing numerous vertices and edges throughout the whole walkable area. The irregular navigation mesh models the same surface, but requires only a few triangles. However, shortest
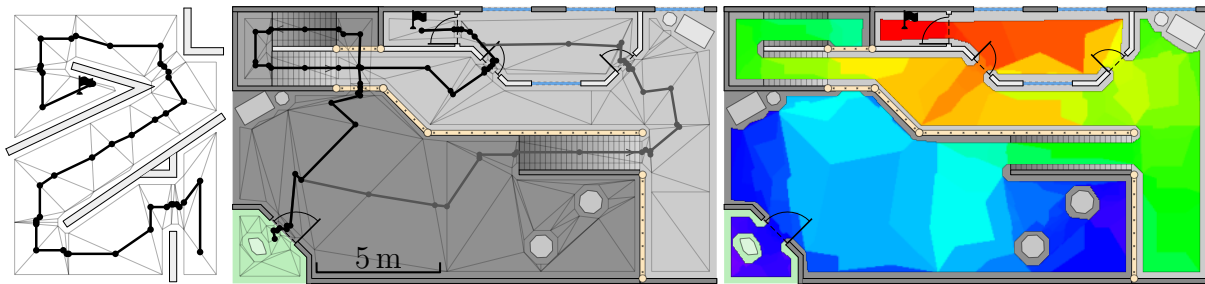
Figure 3.25: Synthetic (left) and real-world example for shortest paths on navigation meshes when using a graph created by connecting edge-midpoints for Dijkstra's algorithm. Resulting paths avoid obstacles but often suffer from abrupt heading changes or zig-zag patterns. The heat map visualizes the walking distance towards the destination for every location within the building, where warm colors (red) are near, and cold colors (blue) are farther away. The heat map is created by searching for the nearest edge-midpoint, and using its known distance towards the destination.

path algorithms, like Dijkstra or A^*, rely on a graph-based data structure. Here, the navigation mesh's representation is disadvantageous. While adjacent triangles always share two vertices, and thus also denote a graph-like structure, suited for Dijkstra or A^*, it is too sparse to provide viable results. Furthermore, this representation tends to favor exterior edges, as they provide the shortest possible connection [Kal10a], yielding the same unnatural walking patterns as the graph without an obstacle-avoiding metric (see figure 3.17 and section 3.5.3).

A slight improvement is given when deriving a graph by connecting the edge-midpoints of each triangle, instead of the their corner vertices. This avoids movements along outer edges [Kal05; Cha82]. The results for a synthetic and a real-world example are shown in figure 3.25. Except for some minor issues with abrupt heading changes, the overall result for the synthetic floorplan is viable. Within the real-world example, large open spaces clearly suffer from the discussed drawbacks, indicated by zig-zag connections. The quality of the routing graph strongly depends on the size and placement of the triangles, which in turn depend on the segmentation algorithm used for deriving the navigation mesh. A shortest path calculation based on this result thus represents an approximation of the actually required walking distance.

A corresponding distance-to-destination heat map is shown in the right of figure 3.25. For every location, it depicts the nearest edge-midpoint's known distance towards the destination. The lower left of this heat map indicates major changes in distance between adjacent triangles. Their size and placement is unfavorable for the shortest path calculation. While there are many variations of this strategy, like connecting triangle-centers instead of edge-midpoints, or a combination of both, they all suffer from similar drawbacks [Kal05].

Results can be improved by subdividing large triangles, before deriving the temporal graph. This is similar to the evenly-sized tessellation of the walkable surface, presented by [Hil+14]. Yet, this causes new issues, as it increases memory requirements, which were originally meant
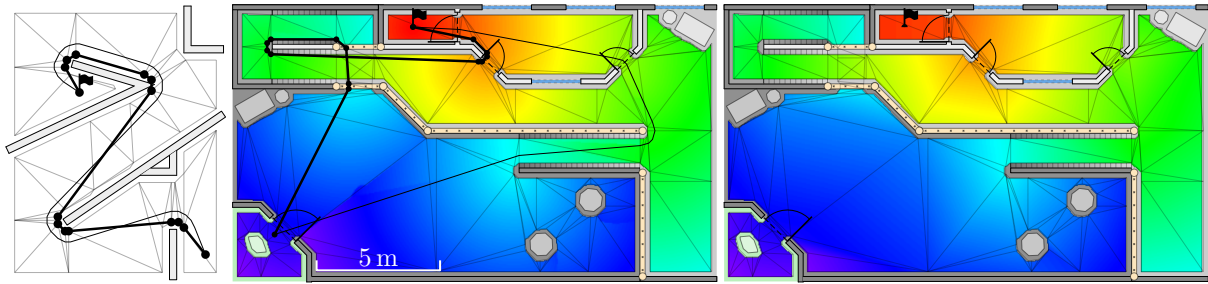
Figure 3.26: Synthetic (left) and real-world example for shortest paths on navigation meshes when using the funnel algorithm. Resulting paths are straight and realistic, but stick to obstacles (bold lines). The extended version of the algorithm adds additional clearance, avoiding obstacles, being more realistic (thin lines). The heat map (center) visualizes the walking distance towards the destination for every location within the building, is smooth, but computationally expensive. It can be approximated by barycentric interpolation (right), and differs only slightly from the original.

to be decreased by using the navigation mesh. Similarly, the complexity of all required computations increases as well, while still yielding inferior results.

A robust solution is given by the *(extended) funnel algorithm* [Kal10b; HS94]. Instead of strictly walking along edges, it also allows for straight movements through primitives. This is achieved by first calculating the shortest path like earlier, hereafter refining the result, by walking directly through primitives whenever possible. That is, the algorithm analyzes the walkable surface itself, instead of only the approximation. Its output is often superior to the one derived from the navigation mesh in section 3.5.3. Results for a synthetic and a real-world example are shown in figure 3.26. Compared to the one from figure 3.25, the path resulting from the funnel algorithm is smooth and straight. However, it sticks unnaturally close to obstacles. This is addressed by an extensions to the algorithm, adding margins, to walk around obstacles. Hereafter, resulting paths resemble actual pedestrian walking behavior. Both algorithms are also shown for the real-world floorplan. As earlier, when obstacles are not avoided, the upper-left stairs denote the shortest path towards the destination. When adding a safety margin to avoid obstacles, the shortest path uses the stairs on the right, and looks more natural.

While the algorithm provides realistic walking paths, it requires complex calculations for every single location in question. Compared to the graph-based solution from section 3.5.3, pre-calculating and storing the shortest path from every single location towards the destination, is impossible. However, as mentioned within the discussions on movement predictions, the actual path is irrelevant. The only information required is whether a new prediction $\mathbf{pos}_{xyz}\left(\boldsymbol{q}_t\right)$ is nearer to the destination than the previous one $\mathbf{pos}_{xyz}\left(\boldsymbol{q}_{t-1}\right)$. To perform this comparison, every location within the navigation mesh must know its distance towards the chosen destination. When using triangles as primitives, this information can be approximated by barycentric interpolation. After choosing a destination, the funnel algorithm is used to calculate the dis-

tance towards the destination for every *vertex* of the mesh's triangles, once. This hereafter allows estimating the approximate distance towards the destination for every location within the triangles, using the barycentric interpolation. While results will be slightly different than performing an actual calculation of the funnel algorithm, they are close enough. The interpolation is computationally inexpensive, and additional memory requirements for storing the distance information are negligible, thus well-suited for smartphone use. Figure 3.26 shows the corresponding heat maps, denoting the distance towards the destination within the small room on the second floor. The left uses exactly calculated distances from the funnel algorithm, the right is based on barycentric interpolation. As can be seen, there is only a minor difference between both. The result can be used in a similar way as earlier for the navigation grid (3.32), weighting movements on whether they approach the destination ρ_{dest}, or depart from it

$$
\begin{aligned}
\mathrm{w}(\boldsymbol{q}_t) &= p_{\text{dest}} \left(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1} \right), \quad \langle \boldsymbol{q} \rangle_t = \langle (x, y, z, \ldots) \rangle_t \\
&= \begin{cases} \kappa_{\text{dest}} & \text{dist}^*_{\text{xyz}} \left(\mathbf{pos}_{\text{xyz}} \left(\boldsymbol{q}_t \right), \rho_{\text{dest}} \right) < \text{dist}^*_{\text{xyz}} \left(\mathbf{pos}_{\text{xyz}} \left(\boldsymbol{q}_{t-1} \right), \rho_{\text{dest}} \right) \\ (1 - \kappa_{\text{dest}}) & \text{else}. \end{cases}
\end{aligned} \quad (3.43)
$$

3.7 Summary

Within this chapter, the likelihood for certain pedestrian movements and their probabilistic prediction, based on a floorplan, was discussed. Similar to car navigation systems, this information can be used to restrict certain movements, addressing sensor faults and uncertainties. Presented models ranged from simple analytical 2D setups, to highly discontinuous 3D variants, considering the floorplan and sensor observations. The latter require spatial data structures, where two representatives were introduced. Based on them, various new probabilistic movement predictions were derived. Finally, the aspect of navigation was discussed, developing new algorithms for realistic routing within buildings, and corresponding movement predictions.

For an initial impression, the first movement predictions were completely unconstrained, analytical 2D variants. They visualized the drawbacks of not including the floorplan, and other prior knowledge. Denoting large homogeneous shapes for new potential whereabouts, they are viable for a few use cases only. As pointed out, especially the current walking direction is crucial for the quality of the estimated predictions. However, including this information already reached the limitations of analytical approaches. While possible in general, the result represents only an approximation of actual likelihood.

Therefore, the concept of simulation was briefly introduced. Instead of analytical calculations, densities were represented by multiple samples. When applying a KDE on top of these

samples, a continuous and calculable density is derived. These simulations also enable considering the building's floorplan. Each sampled movement can be examined whether it is blocked by an obstacle, e.g. by using intersection tests. However, all discussed approaches referred to two dimensional setups only. To efficiently predict 3D pedestrian movements, spatial floorplan models are required, describing the walkable surface.

The first examined data structure was a graph with vertices and edges. Its vertices were placed regularly throughout the whole walkable area of a building, covering ground floor, stairs and similar. When two adjacent vertices are physically reachable from one another, they got connected by an edge. The result describes the building's walkable surface, referred to as navigation grid. Hereafter, an algorithm to estimate new potential whereabouts based on random walks along this graph was developed. A potential destination is created by starting from a given origin, and randomly following adjacent edges. For derived destinations to be meaningful, edges are chosen based on probability metrics, denoting their likelihood, for example based on a known heading. Thus, all resulting samples are distributed based on the likelihood of the edges they followed. The navigation grid allows for an efficient 3D movement prediction, based on available knowledge. With graphs well suited for shortest path estimations, this strategy also enabled navigation. By introducing some adjustments, the calculated paths became realistic, denoting potential routes through the building, and avoiding nearby obstacles. Yet, the navigation grid requires large amounts of memory, and shows a rather discrete behavior.

Therefore, the navigation mesh was discussed as an alternative, also providing a spatial representation, yet, requiring less memory. It describes a building's interior by triangles of varying size and location. As adjacent triangles share one of their edges, they closely resemble the walkable surface. This data structure does not allow for random walks, and new probabilistic predictions had to be developed. One was similar to an initial analytic variant, calculating potential destinations based on walking speed, direction and uncertainty, yet, accepting it only if physically reachable. In contrast to earlier, the navigation mesh determines the reachability efficiently, and allows for a 3D estimation of the destination, impossible for the analytic variant. However, including additional probabilistic constraints besides distance and heading is limited. Therefore, the concept of uniform sampling and weighting was briefly introduced, equipping each sample with a weight, which denotes the sample's likelihood. These weights also allowed for including navigational knowledge, indicating whether a potential movement approaches the destination. The required routing was calculated by the extended funnel algorithm. It derives realistic walking paths, by avoiding obstacles. To prevent costly calculations, an efficient interpolation strategy was developed that only needs to be calculated once.

The presented approaches allow an efficient prediction of potential movements, usable to constrain sensor uncertainties. Combining both aspects is the topic of the following chapter.

Chapter 4

Recursive Density Estimation

As discussed within the two previous chapters, just using sensor observations to perform location estimations will yield results that are unstable, due to noise and uncertainty present within the readings. If the unknown state is observed by a single sensor only, and estimated solely by its noisy measurement, the resulting estimation directly depends on this measurement's quality. By using additional information on surroundings, like a road map or the building's floorplan, impossible movements indicated by some sensor can be addressed and compensated. However, there remains a discrepancy, when sensor readings are erroneous, but the indicated movement is possible based on the map, yielding unstable and jumping location estimations. As discussed in chapter 2, averaging is able to compensate sensor noise, especially when it is of the zero mean Gaussian type, by using more than one reading to perform the estimation. However, the presented types of low-pass filters resulted in increased delays of the output. While delays are not a problem for static measurands, like a moored ship, or a resting pedestrian, they are cumbersome for dynamic systems with moving objects, which represent the main topic of this work.

The effect of using the moving average of sensor data to perform an estimation, is analyzed by an example. Assuming the one dimensional problem, to determine the position of a static object. Its current 1D position is observed within o_t, by a noisy location sensor. For now, its real position $\tilde{q}_t$ is constant, at $10\,\mathrm{m}$, and the location sensor's uncertainty is a zero mean Gaussian:

$$\tilde{q}_t = 10\,\mathrm{m}\,, \quad o_t = \tilde{q}_t + \mathcal{N}(0, \sigma^2)\,, \quad \sigma = 3\,. \tag{4.1}$$

To improve the location estimation, the incoming sensor data o_t is filtered by a moving average low-pass filter. It is implemented using a continuous equation, similar to simple IIR filters and the complementary filter discussed in section 2.4.1 and 2.4.2:

$$q_t = \kappa\, q_{t-1} + (1 - \kappa)o_t\,, \quad \kappa = 0.9\,, \quad q_0 = 0\,. \tag{4.2}$$

In (4.2) the incoming observations o slowly contribute to the new state variable q_t, linearly mixed with the previous one q_{t-1}, representing the low-pass. Due to the mixing, the estimated result will experience a delay, as new observations are slowly faded in, depending on the value of $(1 - \kappa)$. This effect is depicted in figure 4.1a. After the sensor provides its first readings, it takes some time for the estimation q_t to move from $q_0 = 0$ towards the real location. As the latter belongs to a non-moving object, the system is stable after the low-pass filter converges.

If the underlying measurand is dynamic, e.g. a to be localized moving object, aforementioned delays remain present at any instant in time. For a corresponding example of a dynamic case, the state of an object's position is assumed to start at $\tilde{q}_0 = 10\,\mathrm{m}$, and hereafter increase by $0.5\,\mathrm{m/s}$. The behavior of the observation remains as for the static variant in (4.1):

$$\tilde{q}_t = 10\,\mathrm{m} + t\,0.5\,\mathrm{m/s}\,, \quad o_t = \tilde{q}_t + \mathcal{N}(0, \sigma^2)\,, \quad \sigma = 3\,. \tag{4.3}$$

The results are shown in figure 4.1b. While the output of (4.2) starts to approach the unknown state of (4.3) based on the value chosen for κ, the state has already changed to a new value. Thus, the filter's output constantly lacks behind the real value. However, when the behavior of the dynamic entity, like a car, ship or pedestrian, is known, assumptions on potential changes can be made. This is where the dynamic transition models from chapter 3 come in to play. When e.g. the speed of a moving object is known, future locations can be predicted, depending on the elapsed amount of time. The delay in (4.2) results from the old estimation κq_{t-1} being favored over new sensor readings, and q_t is unable to catch up. This can be addressed when the dynamic behavior is known.   Instead of solely using κq_{t-1} as starting value for the next estimation q_t, the system's behavior is included as well, *predicting* expected changes since q_{t-1}. If this prediction $\Delta\overline{q}_t$ is added to the previous estimation q_{t-1}, the delay induced by the filter is compensated by the known behavior. The resulting filter is written as follows:

$$q_t = \kappa\left(q_{t-1} + \Delta\overline{q}_t\right) + (1 - \kappa)o_t\,, \quad 0.5 < \kappa < 1.0\,, \quad q_0 = 0\,, \tag{4.4}$$

where $\Delta\overline{q}_t$ defines the system's dynamic behavior from (4.3), in this case thus given by

$$\Delta\overline{q}_t \overset{!}{=} \tilde{q}_t - \tilde{q}_{t-1} = t\,0.5\,\mathrm{m/s}\,. \tag{4.5}$$

(4.4) equals the complementary filter (2.43), except that its gyroscope sensor is replaced by a prediction $\Delta\overline{q}_t$ of the system's behavior. Figure 4.1c shows the resulting estimation when using (4.4) and (4.5) with $\kappa = 0.9$, instead of (4.2). Due to the known system dynamics, the prediction $\Delta\overline{q}_t$ is able to compensate the introduced delay, and the filtered output converges similar to the static version shown in figure 4.1a.
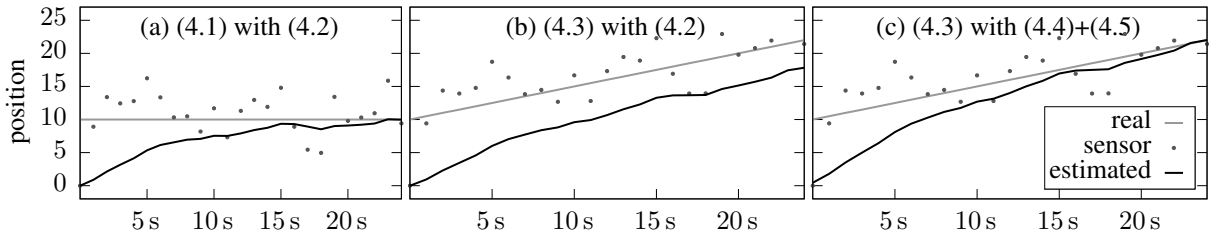
Figure 4.1: Behavior of a state estimation for (4.1) and (4.3), using the filters (4.2) or (4.4)+(4.5) on sensor observations. When the measurand is static (a), the estimation stabilizes after several seconds. For dynamic measurands, the estimation is a delayed version of the real value (b). This can be compensated when the system's behavior is known and can be predicted (c).

Figure 4.2 depicts the results for another dynamic system, where the unknown state performs a sinusoidal oscillation, observed by a sensor within an uncertainty of $\sigma = 1$

$$\tilde{q}_t = \sin(0.125\,t)\,4\,\mathrm{m/s}\,, \quad o_t = \tilde{q}_t + \mathcal{N}(0, \sigma^2)\,, \quad \sigma = 1\,. \tag{4.6}$$

The prediction $\Delta\overline{q}_t$ used within (4.4) is adjusted accordingly:

$$
\begin{aligned}
\Delta\overline{q}_t &= \tilde{q}_t - \tilde{q}_{t-1} = 4\,\mathrm{m/s}\Big(\sin(0.125\,t) - \sin\big(0.125\,(t-1)\big)\Big) \\
&\approx \Delta t \left(\Big(\sin\big(0.125\,(t-0.5)\big)\big)4\,\mathrm{m/s}\Big)\frac{\mathrm{d}}{\mathrm{d}t}\right) = \Delta t\,\frac{1}{2}\cos\big(0.125\,(t-0.5)\big)\,.
\end{aligned}
\tag{4.7}
$$

As earlier, $\Delta\overline{q}_t$ within (4.7) describes the system's dynamic change within a timeframe. Instead of using the difference between $\tilde{q}_t$ and $\tilde{q}_{t-1}$, this change can also be described by the derivative of the underlying process, times the timeframe. While the latter is an approximation, it is desirable for some situations, discussed later. As can be seen within figure 4.2a, when not using a prediction, the filtered output quickly starts to deviate from the real underlying value. The filter not only introduces a delay, the amplitude of the oscillation is changed as well. This is due to its design, using $(1 - \kappa)$ to slowly introduce new observations. As soon as q_t gets near the oscillation's maximum amplitude, actual sensor values already started to decrease, preventing the estimation from reaching it. For lower frequencies in (4.6), or smaller values of κ, the effect of the changed amplitude becomes less pronounced. A smaller κ, however, increases the impact of sensor noise, as it changes the low-pass filter's cut-off frequency. When including the known system behavior as prediction in figure 4.2b, filtered results become viable.

In the presented examples, κ was an empiric, constant choice, which will not suit real-world scenarios. While lower values of κ introduce less delay, they pass more sensor noise, and vice versa. κ thus depends on the amount of noise present within the sensor readings, which might be dynamic as well. If noise is minor, $(1 - \kappa)$ should be large, to favor the current observation o_t, and vice versa. Within previous examples, the system's behavior was well-
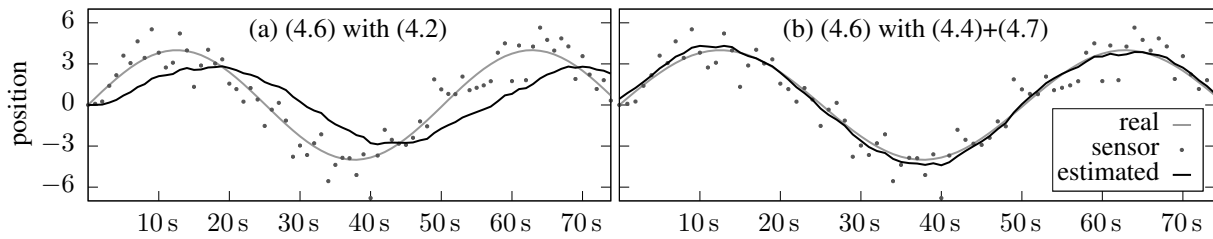
Figure 4.2: Behavior of a state estimation for (4.6) using the filters (4.2) or (4.4)+(4.7) on sensor observations. When no prediction is available (a) the estimation constantly lacks behind the real location. With prediction (b), estimation and real value are almost identical.

known, and the prediction was expected to be exact without errors. For real-world use cases, however, the prediction is uncertain as well. The dynamics often are not fully known, can not directly be inferred for absolute points in time, and are as uncertain as the observed sensor data [Jaz70; GSS93]. The movement behavior of a ship, for example, depends on the current speed and rudder position, which are subject to changes, and can only be estimated based on sensor readings. Furthermore, speed and current location are influenced by *unknown external factors*, such as wind, waves and ocean current. Just as the unknown state, a prediction can be dependent on sensor observations, and is subject to unknown external influences. This also leads to the question of how trustworthy the prediction currently is.

While (4.4) already contains a concept for either favoring the prediction or the sensor, given by κ, it supports only a single sensor. More could be added as $(1 - \kappa)o$ terms – one for every sensor, each using a different κ – but determining the correct value for each is cumbersome. Furthermore, the presented example described the unknown state as a single scalar. For indoor localization, however, the state contains the pedestrian's 3D position and heading. Also, not every sensor contributes to all four values (cf. chapter 2 and chapter 3). Additionally, the presented prediction is unable to include prior information, e.g. given by a floorplan. Finally, while (4.4)'s result was more realistic than the unfiltered observation, the quality of this result is unknown. Also, it is unable to include known uncertainties in a probabilistic way, representing a drawback, discussed in the two previous chapters.

This chapter thus focuses on methods and algorithms that can be used to combine multiple sensor inputs, to optimally estimate an unknown state. Algorithms must be able to include uncertainties present within both, sensor readings, and the dynamic prediction process. Referring to indoor localization and navigation, the sensor readings from the smartphone, discussed in chapter 2, are combined with potential movements of the pedestrian, discussed in chapter 3. The unknown state, that is, the pedestrian's whereabouts and heading, is determined based on a probabilistic fusion of the evaluations and transitions, described within both previous chapters.

$$p(q \mid o) = \eta \, p(o \mid q) \, p(q)$$

$p(q)$

$p(o \mid q)$
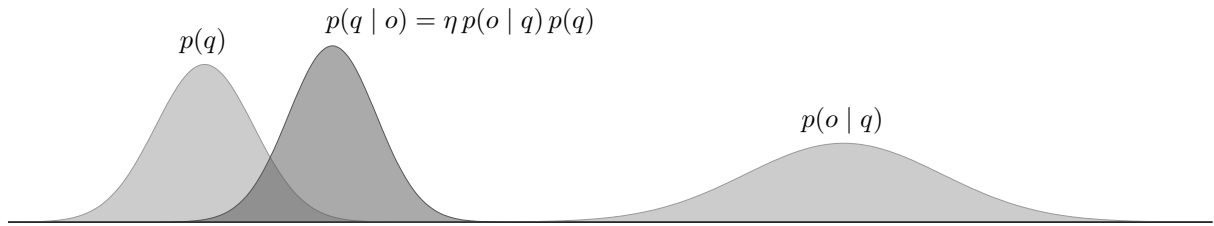
Figure 4.3: Combining a prior estimation (left) with current sensor observations (right) to determine the current posterior (center) via multiplication and normalization. Even though both probabilities (left/right) seem near zero, normalization yields another Gaussian distribution.

4.1 Probabilistic Information Fusion

As discussed, to determine the unknown state of a dynamic system, such as a moving vehicle, ship or pedestrian, information available by sensors and models should be combined in a probabilistic manner. Doing so includes known uncertainties, and provides a result that is *optimal* for the given values, including an accuracy indication for the output. Referring to the simple 1D localization example from earlier, a probabilistic combination based on uncertainties is briefly examined. Assuming the 1D location of the object to be known, including some uncertainty, given by $p(q)$ with $q = x$, referred to as *prior*. At the current location of the object, a sensor reading $o = x$ is observed, yielding a hint on the current location, as seen from the sensor, called *measurement* $p(o \mid q)$. According to Bayes' rule, the object's most likely whereabouts q given the sensor reading o and initial guess $p(q)$, is called *posterior* $p(q \mid o)$ and defined as

$$\underbrace{p(q \mid o)}_{\text{posterior}} = \frac{\overbrace{p(o \mid q)}^{\text{measurement}} \overbrace{p(q)}^{\text{prior}}}{\underbrace{p(o)}_{\text{normalization}}} = \eta \, p(o \mid q) \, p(q) = \begin{cases} \dfrac{p(o \mid q) \, p(q)}{\sum_{q'} p(o \mid q') \, p(q')} & \text{discrete} \\[3ex] \dfrac{p(o \mid q) \, p(q)}{\int_{-\infty}^{\infty} p(o \mid q') \, p(q') \, \mathrm{d}q'} & \text{continuous .} \end{cases} \tag{4.8}$$

Shown within (4.8), the posterior is given by the product of the initial expectations and current sensor observations, including a normalization, to ensure the integral of the result yields 1, and thus describes a PDF as well. For readability, the normalization is abbreviated as η. Figure 4.3 depicts a corresponding example of the posterior resulting from prior and observation. Even though both distributions barely overlap, and the result of their product is near zero, the normalization η yields a new PDF. When both, prior and observation, are distributed normally, such as within the figure, the normalized posterior denotes a normal distribution as well [Smi11]

$$\eta \, \mathcal{N}\left(x \mid \mu_1, \sigma_1^2\right) \mathcal{N}\left(x \mid \mu_2, \sigma_2^2\right) = \mathcal{N}\left(x \mid \mu_{1,2}, \sigma_{1,2}^2\right) . \tag{4.9}$$

The new $\mu_{1,2}$ and $\sigma_{1,2}^2$ of the resulting distribution are given by

$$\mu_{1,2} = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_1^2 + \sigma_2^2}, \quad \sigma_{1,2}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}. \tag{4.10}$$

For the following discussions, a slightly different notation is used for $\mu_{1,2}$ and $\sigma_{1,2}^2$, where a part of the equations is shared, and denotes how the two PDFs are combined. By adding the substitution term $+(\sigma_1^4 - \sigma_1^4)$, $\sigma_{1,2}$ can be rewritten as

$$\sigma_{1,2}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \frac{\sigma_1^2 \sigma_2^2 + \sigma_1^4 - \sigma_1^4}{\sigma_1^2 + \sigma_2^2} = \frac{\sigma_1^2 \sigma_2^2 + \sigma_1^4}{\sigma_1^2 + \sigma_2^2} - \frac{\sigma_1^4}{\sigma_1^2 + \sigma_2^2} = \sigma_1^2 - \frac{\sigma_1^4}{\sigma_1^2 + \sigma_2^2}. \tag{4.11}$$

$\mu_{1,2}$ is adjusted similarly, by adding $+(\mu_1 \sigma_1^2) - (\mu_1 \sigma_1^2)$

$$\mu_{1,2} = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_1^2 + \sigma_2^2} = \frac{(\mu_1 \sigma_1^2) + \mu_1 \sigma_2^2 + \mu_2 \sigma_1^2 - (\mu_1 \sigma_1^2)}{\sigma_1^2 + \sigma_2^2} = \frac{\mu_1 \sigma_1^2 + \mu_1 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} + \frac{\mu_2 \sigma_1^2 - \mu_1 \sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$
$$= \frac{\mu_1 (\sigma_1^2 + \sigma_2^2)}{\sigma_1^2 + \sigma_2^2} + \frac{\sigma_1^2 (\mu_2 - \mu_1)}{\sigma_1^2 + \sigma_2^2} = \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} (\mu_2 - \mu_1). \tag{4.12}$$

Finally, (4.10) can be condensed to

$$\begin{aligned} \mu_{1,2} &= \mu_1 + k(\mu_2 - \mu_1) \\ \sigma_{1,2}^2 &= \sigma_1^2 - k\,\sigma_1^2 \end{aligned} \quad \text{with} \quad k = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} = \sigma_1^2 \left(\sigma_1^2 + \sigma_2^2\right)^{-1}. \tag{4.13}$$

According to (4.13), the new mean and variance are combined by a scalar value k, which solely depends on the uncertainty of the to-be-combined distributions: the prior and the measurement. Due to the fraction, $0 \le k \le 1$ with $k = 0$ for $\sigma_1 = 0$, and $k = 1$ for $\sigma_2 = 0$. The new mean $\mu_{1,2}$ depends on the prior's mean plus k times the difference between the prior and measurement. In other words, for $k = 0$, the new mean equals the prior's mean, for $k = 1$ the new mean equals the measurement's mean. For $0 < k < 1$ it is somewhere in between. k thus works exactly as κ within (4.2)/(4.4), deciding how to mix the prediction and the measurement. Yet, k is not empirically chosen, but depends on uncertainties. If the σ_1 of the prior is smaller than the σ_2 of the measurement, $k < 0.5$, and the posterior is nearer to the prior than to the measurement. If σ_2 is the smaller one, $k > 0.5$, and the posterior shifts towards the measurement. In other words, k decides whether the prediction or the measurement is to be trusted more.

As (4.13) points out, the new uncertainty $\sigma_{1,2}$ is also affected by k. However, no mixing between both uncertainties σ_1 and σ_2 is applied here. The new uncertainty equals the one of the prior, minus a fraction of the same value. Thus, the uncertainty of the posterior is unable to grow, but can only decrease. Put another way, the posterior will always be more accurate than the prior, independent of the uncertainty of the measurement.

The discussed concept also allows for adding observations from more than one sensor, that occurred at the same instant in time. Assuming statistical independence of all sensors, the combined probability of all currently available sensor readings $\boldsymbol{o}$, given the current unknown state q, is determined by the product of the individual probabilities

$$p(\boldsymbol{o} \mid q) = \prod_{i=1}^{N} p(o_i \mid q), \quad \boldsymbol{o} = (o_1, \ldots, o_N) . \tag{4.14}$$

Again, if all observations are distributed normally

$$p(o_i \mid q) = \mathcal{N}\left(o_i \mid \mu_i, \sigma_i^2\right) , \tag{4.15}$$

the result of (4.14) is given by applying (4.13) $N - 1$ times, to combine all N sensor readings into a single normal distribution. If some sensor was faulty and did not provide an observation, it can simply be omitted. Due to (4.13), every additional sensor yields a decrease in uncertainty, rendering the result more accurate. However, (4.15) only holds true for independent sensors, providing their observations at the same instant in time. For subsequent observations, a different approach is required, as it is still unclear how the time component, and e.g. older observations, contribute to the overall result. Within the previously presented example of the IIR filter, new sensor readings yielded a contribution as they occurred, stabilizing the location estimation of an object over time, refining the previous estimation. Furthermore, a prediction was added, to describe the system's dynamic behavior and thus compensate delays introduced by the filtering process. Both aforementioned aspects must be included within the probabilistic variant as well.

4.2 Bayes Filter

This leads to the question of how a probabilistic estimation can be improved when there are subsequent observations from sensors at different points in time, that is, a history of observations, and when a system and its unknown state are dynamic. The answer to both is provided by making the posterior $p(q \mid o)$ from (4.8) dependent on the current time t, and conditioning it not only on the current observation o, but on the complete history of observations $o_{1:t}$ since the beginning up until t, as defined in (2.3). The dynamic behavior is added by conditioning the posterior on all internal and external influences $u_{1:t}$, that affect the state

$$\langle \boldsymbol{u} \rangle_t = \boldsymbol{u}_{1:t} = \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1}, \boldsymbol{u}_t \quad \text{with} \quad \langle \boldsymbol{u} \rangle_t = \langle (\ldots) \rangle_t . \tag{4.16}$$

For localization problems, u_t might e.g. be given by a ship's rudder angle, but is also affected by environmental influences, such as wind and ocean current. In other words, influences that somehow affect the whereabouts over time. The discussed adjustments result in

$$p(q \mid o)_{(4.8)} \quad \to \quad p(q_t \mid o_{1:t}, u_{1:t}) = \mathrm{bel}(q_t)\,, \tag{4.17}$$

which, in literature [TBF05; Jaz70; Aru+01], is also referred to as *belief*, how the unknown state q_t might currently look like, based on the complete history of observations $o_{1:t}$, and influences $u_{1:t}$. The notation for the history of previous values is equivalent to

$$p(q_t \mid o_{1:t}, u_{1:t}) = p(q_t \mid o_t, \dots, o_1 \ , \ u_t, \dots, u_1)\,. \tag{4.18}$$

To rewrite the conditional probability (4.18) just like earlier in (4.8), a modified version of Bayes' rule, conditioned on more than one variable, can be used

$$
\begin{aligned}
p(a \mid b, c, \dots) &= \frac{p(a, b, c, \dots)}{p(b, c, \dots)} & &\mid \text{reorder, } p(a, b) = p(b, a) \\
&= \frac{p(b, a, c, \dots)}{p(b, c, \dots)} & &\mid \text{apply } p(a, b) = p(a \mid b)\, p(b) \\
&= \frac{p(b \mid a, c, \dots)\, p(a, c, \dots)}{p(b \mid c, \dots)\, p(c, \dots)} & &\mid \text{apply } p(a, b) = p(a \mid b)\, p(b) \\
&= \frac{p(b \mid a, c, \dots)\, p(a \mid c, \dots)}{p(b \mid c, \dots)} \frac{p(c \mid \dots)\, p(\dots)}{p(c \mid \dots)\, p(\dots)} \\
&= \frac{p(b \mid a, c, \dots)\, p(a \mid c, \dots)}{p(b \mid c, \dots)} & &\mid \text{move denominator to } \eta \\
&= \eta\, p(b \mid a, c, \dots)\, p(a \mid c, \dots)\,.
\end{aligned} \tag{4.19}
$$

Applying (4.19) to (4.18) then yields:

$$
\begin{aligned}
\mathrm{bel}(q_t) &= p(q_t \mid o_t, \dots, o_1 \ , \ u_t, \dots, u_1) \\
&= \eta\, p(o_t \mid q_t, o_{t-1}, \dots, o_1 \ , \ u_t, \dots, u_1)\, p(q_t \mid o_{t-1}, \dots, o_1 \ , \ u_t, \dots, u_1) \\
&= \eta\, \underbrace{p(o_t \mid q_t, o_{1:t-1}, u_{1:t})}_{\text{measurement probability}}\, \underbrace{p(q_t \mid o_{1:t-1}, u_{1:t})}_{\text{prior}}\,,
\end{aligned} \tag{4.20}
$$

which is still similar to (4.8), yet uses the discussed conditioning variables. However, (4.20) can be simplified when introducing a new assumption: The overall problem formulation of estimating the current location of an object, based on sensor observations, influences, and prior knowledge, can be described as a continuous *(hidden) Markov model* (HMM). Hidden refers to observations that only provide implicit hints on the current state, as they denote only relative information, like speed and heading change, and/or are erroneous (cf. chapter 2) [BP66; TBF05].

Such models satisfy the so called *Markov property* [Mar51]. Here, this property states that the next state q_t can be predicted solely by the previously estimated state q_{t-1}, and the *current* observations o_t, and influences u_t, respectively. In other words, when the Markov property holds true, past observations and influences provide no additional information. They are implicitly contained within every state q. Thus, neither the history of estimated states, nor the observations and influences that lead to those estimations, are required. When the Markov property holds true, the *measurement probability* from (4.20) can be simplified to

$$p(o_t \mid q_t, o_{1:t-1}, u_{1:t}) = p(o_t \mid q_t). \tag{4.21}$$

This is possible, because (4.21) is conditioned on q_t, which fully contains all observations and influences up to time t, when the Markov property holds true.

This simplification does not work for the prior given in (4.20), as it is not conditioned on the state q. To also utilize the Markov property for the prior, the previous state has to be added to the equation. This is achieved using the *law of total probability* [BC12]

$$p(a) = \int p(a \mid b) \, p(b) \, db$$
$$p(a \mid \ldots) = \int p(a \mid b, \ldots) \, p(b \mid \ldots) \, db. \tag{4.22}$$

By applying (4.22), the prior from (4.20) can be conditioned on some new arbitrary variable. This allows for conditioning the prior on the state q, required for utilizing the Markov property. In this case, q_{t-1} is used as conditioning variable, for reasons yet to discuss

$$p(q_t \mid o_{1:t-1}, u_{1:t}) = \int p(q_t \mid q_{t-1}, o_{1:t-1}, u_{1:t}) \, p(q_{t-1} \mid o_{1:t-1}, u_{1:t}) \, dq_{t-1}. \tag{4.23}$$

As (4.23) is conditioned on q_{t-1}, all information that lead to this state can be omitted. That is, the history of observations $o_{1:t-1}$, and the influences $u_{1:t-1}$, except the most recent one u_t. Within the second parentheses, u_t is removed, as this is future information regarding q_{t-1}, and can thus safely be omitted. The discussed changes result in

$$p(q_t \mid o_{1:t-1}, u_{1:t}) = \int p(q_t \mid q_{t-1}, u_t) \, p(q_{t-1} \mid o_{1:t-1}, u_{1:t-1}) \, dq_{t-1}. \tag{4.24}$$

Examining (4.17), the second parentheses within (4.24) clearly denotes the previous believe, at time $t-1$. The resulting equation uses this previous believe, and combines u_t with a transition probability $q_{t-1} \rightarrow q_t$, to convert it into a new believe. For the localization problem this reads: *"Assuming the last believe was correct, given the current influences and some transition probability, where might the dynamic object have moved to?"*. In literature, this equation is therefore

often referred to as *predicted believe* or *prediction* [GSS93; KB61; TBF05], written as

$$\overline{\text{bel}}(q_t) = p(q_t \mid o_{1:t-1}, u_{1:t}) = \int \underbrace{p(q_t \mid q_{t-1}, u_t)}_{\text{prediction}} \underbrace{\text{bel}(q_{t-1})}_{\text{previous believe}} \, dq_{t-1} \, . \qquad (4.25)$$

The predicted believe $\overline{\text{bel}}(q_t)$ is given by the previous believe $\text{bel}(q_{t-1})$ and a transition probability, modeling the object's dynamics. As can be seen in (4.25), the prediction describes how a *single* state q_{t-1} is converted into some new state, which is due to utilizing the law of total probability. As the old state represents a distribution as well, every potential value has to undergo the prediction of what it might be hereafter, denoted by the integral. This can be interpreted as a convolution between the prior and the prediction model. Applying the presented simplifications from (4.21) and (4.24) to the initial equation (4.20), the posterior can be written as

$$
\begin{aligned}
\text{bel}(q_t) = & \quad p(q_t \mid o_{1:t}, u_{1:t}) \\
= & \eta \quad p(o_t \mid q_t, o_{1:t-1}, u_{1:t}) \quad p(q_t \mid o_{1:t-1}, u_{1:t}) \\
= & \eta \quad p(o_t \mid \overline{q}_t) \qquad\qquad\quad \overline{\text{bel}}(q_t) \\
= & \eta \quad \underbrace{p(o_t \mid \overline{q}_t)}_{\text{measurement}} \qquad\qquad \int \underbrace{p(q_t \mid q_{t-1}, u_t)}_{\text{prediction}} \underbrace{\text{bel}(q_{t-1})}_{\text{recursion}} \, dq_{t-1} \, ,
\end{aligned}
\qquad (4.26)
$$

representing the final equation. It infers the posterior, that is, the believe, or probability distribution, of the unknown state q_t at some point in time t, based on the current sensor observations o_t, the $\overline{q}_t$ predicted from the influences u_t, and the previous state q_{t-1}. For clarification, it distinguishes between the predicted state $\overline{q}_t$, and the posterior q_t. Due to the dependency on q_{t-1}, the equation represents a *recursive* process, which is often referred to as *recursive state estimation*, *recursive Bayesian estimation*, *Bayesian filtering*, or, in short, *Bayes filter* [Sär13; TBF05]. For it to work, the distribution of the first state q_0 at time $t = 0$ must be provided. However, just distributing it uniformly over the whole state space is a valid decision.

The behavior of the presented filter is depicted using a simple one-dimensional example, similar to (4.3), yet utilizing probability distributions instead of scalar values. The unknown state q_t solely contains the object's 1D location. Again, the object's actual dynamic movement is assumed with $0.5\,\text{m/s}$, however, now also including a zero mean Gaussian uncertainty

$$\tilde{q}_t = 10\,\text{m} + t\,(0.5\,\text{m/s} + \mathcal{X}), \quad \mathcal{X} \sim \mathcal{N}(0, \dots)\,. \qquad (4.27)$$

The initial believe $\text{bel}(q_0)$ at time $t = 0$ is the object's initial location, if known, with some degree of uncertainty. Here, it is expressed by the following distribution

$$\text{bel}(q_0) = \mathcal{N}(10, 0.2^2)\,. \qquad (4.28)$$

The prediction also addresses the movement uncertainty $\mathcal{X}$ from (4.27), by using another normal distribution to introduce a similar uncertainty when predicting the object's movement

$$\overline{\text{bel}}(q_t) = q_{t-1} + \mathcal{N}\left(u_t, \sigma^2\right), \quad \sigma = 0.1, \quad u_t = 0.5. \tag{4.29}$$

Shown in (4.29), the new state q_t depends on the previous state q_{t-1}, plus the expected movement of $0.5\,\text{m/s}$, including an uncertainty σ. As (4.29) resulted from the law of total probabilities, it denotes the transition from *one explicit* starting state q_{t-1}. It thus has to be applied to every potential prior state q_{t-1} (cf. integral (4.26)), which is somewhat similar to a convolution between the prior believe $\text{bel}(q_{t-1})$, and each transition probability (4.29). The predicted believe $\overline{\text{bel}}(q_t)$ will thus become more uncertain. This effect is depicted in the left half of figure 4.4, where the prediction is shifted by 0.5 to the right, and is slightly wider than the prior.

The increased uncertainty after the prediction is addressed by sensor observations. Within this example, a sensor measures the object's absolute position with a small uncertainty $\sigma = 0.5$. This uncertainty is included by assigning a probability to the difference between the observations and the prediction $\overline{\text{bel}}(q_t)$, that exactly resembles the expected sensor noise $\sigma = 0.5$

$$p(o_t \mid \overline{q}_t) = \mathcal{N}\left(o_t \mid \overline{q}_t, \sigma^2\right) = \mathcal{N}\left(\Delta \mid 0, \sigma^2\right), \quad \sigma = 0.5, \quad \underbrace{\Delta = o_t - \overline{q}_t}_{\Delta\text{sensor}\leftrightarrow\text{prediction}}. \tag{4.30}$$

Results after two prediction and measurement update iterations are depicted in figure 4.4. The predictions, shown in the left half, clearly denote an increase in uncertainty whenever the object's dynamics are predicted. The resulting distribution is moved by 0.5 to the right, and gets slightly wider. This is intuitive behavior, as estimating a moving object's location, e.g. a car passing a tunnel, will become increasingly inaccurate over time. Hereafter, the following measurement update compares the current sensor observations o_t against the prediction $\overline{q}_t$ including the known sensor noise. This step refines the prediction, and results in the posterior, which is more accurate (figure 4.4, right half). If sensor observations are currently unavailable, the posterior is equal to the predicted believe. The prediction ensures that the unknown state is adjusted, even without observations, yielding an increasing uncertainty over time. Again, this is similar to a car, passing a tunnel, and being tracked solely based on driving speed. This estimation becomes increasingly uncertain. As soon as it exits the tunnel, observations are available, and the cumulated uncertainty is decreased by the observations. Following previous discussions on (4.13), observations decrease uncertainty, while the newly introduced prediction increases it.

The presented equations for prediction and measurement update clearly resemble the probabilistic sensor and transition models discussed in chapter 2 and 3. Yet, it is unclear how to actually calculate the densities for the prediction $\overline{\text{bel}}(q_t)$ and posterior $\text{bel}(q_t)$. While some of
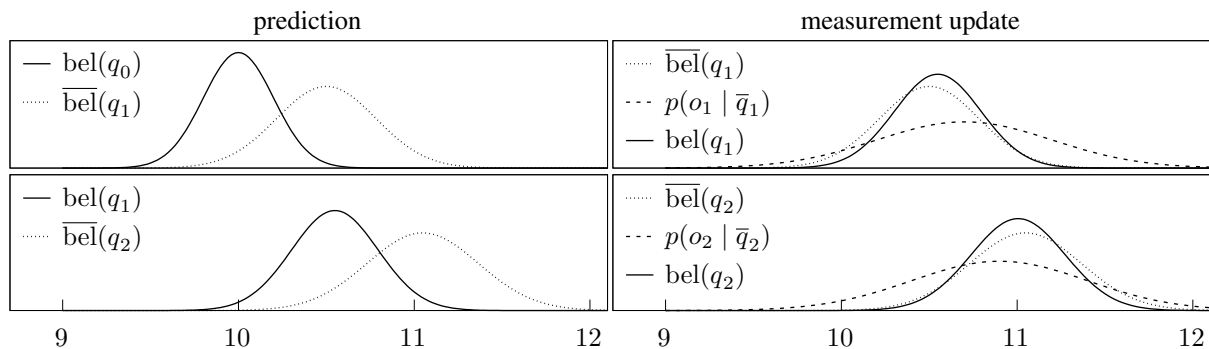
Figure 4.4: Two prediction/update steps using the Bayes filter (4.26) on the presented example (4.27). The left half denotes the prediction of the object's dynamics (4.29). Due to uncertainties, the predicted believe becomes wider than the prior. The right half shows the measurement update (4.30), where the prediction is refined by the current sensor observations, yielding a slightly more certain posterior.

the models from the previous chapters are given as simple normal distributions, just like the presented 1D example, others required multivariate normal distributions, or represent mixture distributions. Dependent on the to-be-solved problem formulation, analytical versions for calculating those densities exist. However, they often come at certain limitations. The same holds true for more complex algorithms, with support for arbitrary distributions. While often being more flexible, they suffer from other to-be-discussed drawbacks. The following sections will focus on real-world implementations for applying the presented Bayes filter to actual problems, and address potential advantages and disadvantages of specific implementations.

4.3   Kalman Filter

The Bayes filter is not limited to specific density functions, and supports arbitrary distributions for believe, prediction and sensor noise. The way actual results are calculated within (4.26) depend on the chosen models, and thus the used density functions. For both, simplicity and correctness, analytical solutions to (4.26) are preferred. However, the analytical product and summation of two densities is only available for a few representatives. The same holds true for the prediction step, affecting the believe density, where analytical solutions are often only available for simple, linear models [TBF05], also shown in chapter 3.

This section focuses on analytical solutions for the Bayes filter. It covers problems that can solely be modeled using normal distributions for the believe, prediction and the measurement update step, and where both, prediction and update, denote a linear system, $Ax = b$. Whenever those constraints can be satisfied, the *Kalman filter* [Kal60; KB61] can be used as analytical implementation of the Bayes filter. Even though normal distributions limit the applicability to unimodal problems, and the requirement for linear systems restricts potential predictions and

measurement updates, it is still applicable to many real-world scenarios. Applications range from general object tracking [CHP79], over inertial navigation [Meh70], and satellite localization [Fit71], towards medical applications, such as tracking retinal blood vessels [CZK98].

The Kalman filter is based on the principles discussed in section 4.1 and 4.2. First, the prior density is adjusted by a prediction step, hereafter being refined by a measurement update, resulting in the posterior. Here, both steps solely refer to the adjustment of normal distributions. Section 4.1 has already shown that the second step, of deriving the posterior, can be solved analytically, as the product of two normal distributions denotes another normal distribution. (4.13) calculated the new μ and σ for the one dimensional case. However, it is yet unclear how to handle the convolution required by the prediction within section 4.2, when using normal distributions. Furthermore, for arbitrary problems, a single dimension is insufficient, and multiple dimensions are required. 3D localization, for example, requires at least 3 dimension's for the whereabouts (x, y, z). The Kalman filter thus uses multivariate normal distributions, to describe believe, prediction and sensor noise:

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right) . \qquad (4.31)$$

The distribution's mean is given by the vector $\boldsymbol{\mu}$, and its uncertainty by the covariance matrix $\boldsymbol{\Sigma}$. The latter not only models the variance for every component, e.g. a position in x, y and z, but also influences between the components, if any. For example, while the three position coordinates might be statistically independent, yielding zeros beside the diagonal of the matrix, an object's *speed* or *direction* influence its position, creating a dependency and thus a *covariance*. Likewise, the distribution's center is given by the vector $\boldsymbol{\mu}$, holding the mean value for every single component. Within the Kalman filter, the distribution's $\boldsymbol{\mu}$ represents the mean value of the believe. The most likely estimation of the unknown state is thus directly given by this vector. While the typical notation in literature uses $\boldsymbol{x}$ for the state, and either $\boldsymbol{y}$ or $\boldsymbol{z}$ for the observations [TBF05; GA01; Kal60], to consent with the previous sections, within the following the state's mean is referred to as $\boldsymbol{q}$, the observation as $\boldsymbol{o}$, and the influences as $\boldsymbol{u}$.

Following (4.26), the dynamics of the underlying model are defined within the prediction step, where the prediction $\overline{\boldsymbol{q}}_t$ incorporates a set of known update rules based on the prior $\boldsymbol{q}_{t-1}$, known internal influences $\boldsymbol{u}_t$, and unknown external influences, which represents a convolution of two distributions. Regarding normal distributions, a convolution of two yields another one, which is due to the following facts: A convolution in sample space equals a multiplication in frequency space [Smi99, p. 180], the Fourier transform of a Gaussian equals another Gaussian [Smi99, p. 216], and the product of two Gaussians yields another Gaussian, as previously shown in section 4.1. As a consequence, the convolution of the prediction step yields a new

normal distribution with different mean and covariance. For the Kalman filter, this adjustment is described by a *linear transformation*. As will be shown shortly, a linear system applied to a Gaussian will yield another Gaussian as well. The transformation is split into two steps, one for the distribution's mean and one for the covariance. The first is given by

$$\overline{\boldsymbol{q}}_t = \boldsymbol{A}\boldsymbol{q}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t \; (+\boldsymbol{\mathcal{X}}), \quad \boldsymbol{\mathcal{X}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \,. \tag{4.32}$$

(4.32) derives the new predicted state $\overline{\boldsymbol{q}}_t$ based on the previous state $\boldsymbol{q}_{t-1}$, adjusted by a matrix $\boldsymbol{A}$. Additionally, known influences $\boldsymbol{u}_t$, adjusted by a matrix $\boldsymbol{B}$, and unknown external influences $\boldsymbol{\mathcal{X}}$, are included as well. The matrix $\boldsymbol{A}$ describes the change in state, when no additional influences are given. The matrix $\boldsymbol{B}$ denotes the mapping from the influences $\boldsymbol{u}$ onto the state $\boldsymbol{q}$. This mapping is necessary, as the influences might be expressed using different units, do not affect every component of the state, or affect more than one component. The zero mean Gaussian noise $\boldsymbol{\mathcal{X}}$ has no impact on the calculation of $\overline{\boldsymbol{q}}_t$, but is assumed to be present. It describes expected uncertainties and systematic errors, increasing the uncertainty of the predicted state $\overline{\boldsymbol{q}}_t$, which is subject to an additional equation.

The essence of (4.32) can be explained using an example, often found in literature [Jaz70], where an object's location x and velocity $\dot{x}$ represent the unknown state, which is affected by a known acceleration $\ddot{x}$ and unknown influences, such as wind, ocean current or similar

$$
\begin{aligned}
x_t &= \overbrace{x_{t-1}}^{\text{previous position}} + \overbrace{\dot{x}_{t-1}\Delta t}^{\text{previous speed}} + \overbrace{0.5\ddot{x}_t(\Delta t)^2}^{\text{known acceleration}} \\
\dot{x}_t &= \underbrace{\dot{x}_{t-1}}_{\text{previous speed}} + \underbrace{\ddot{x}_t\Delta t}_{\text{known acceleration}}
\end{aligned}
\tag{4.33}
$$

To use (4.33) with the Kalman filter, it is rewritten according to (4.32)

$$\overline{\boldsymbol{q}}_t = \underbrace{\begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{pmatrix}}_{\boldsymbol{q}_{t-1}} + \underbrace{\begin{pmatrix} 0.5(\Delta t)^2 \\ \Delta t \end{pmatrix}}_{\boldsymbol{B}} \underbrace{\begin{pmatrix} \ddot{x}_t \end{pmatrix}}_{\boldsymbol{u}_t} \; (+\boldsymbol{\mathcal{X}})$$

$$\langle \boldsymbol{q} \rangle_t = \langle (x, \dot{x})^T \rangle_t \,, \quad \langle \boldsymbol{u} \rangle_t = \langle (\ddot{x}) \rangle_t \,. \tag{4.34}$$

Within (4.34), the matrix $\boldsymbol{A}$ derives the new predicted location and speed $\overline{\boldsymbol{q}}_t$, based on the prior $\boldsymbol{q}_{t-1}$. Likewise, $\boldsymbol{B}$ incorporates the current influences $\boldsymbol{u}_t$, containing the known acceleration, mapping the influences onto the state. As the state represents a normal distribution, this step adjusts its mean, which can be thought of the average of a random variable $\boldsymbol{\mathcal{X}}$

$$\mathbb{E}(\boldsymbol{\mathcal{X}}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \,. \tag{4.35}$$

As can be seen, this value is linearly affected by multiplication and addition. The result of (4.32) thus shifts the distribution's mean, yielding the new center of the predicted state. Furthermore, adding a zero mean random variable $\mathcal{X}$ does not affect the mean itself, but only its covariance, referred to as $\overline{\boldsymbol{P}}_t$. This value is also affected by the multiplication of $\boldsymbol{A}$, yet, in a different way. In general, the covariance of a multidimensional random variable $\mathcal{X}$ is defined as

$$\mathbf{Cov}(\mathcal{X}) = \mathbb{E}\left((\mathcal{X} - \mathbb{E}(\mathcal{X}))\left(\mathcal{X} - \mathbb{E}(\mathcal{X})\right)^T\right). \tag{4.36}$$

According to (4.32), the covariance $\boldsymbol{P}_{t-1}$ belonging to $\boldsymbol{q}_{t-1}$ is affected by $\boldsymbol{A}\boldsymbol{q}_{t-1}$, and $+\mathcal{X}$. The term $+\boldsymbol{B}\boldsymbol{u}_t$ has no impact, as adding a constant value does not affect the covariance, but only the mean. According to (4.36), the zero mean Gaussian noise $\mathcal{X}$ from (4.32) is incorporated by simply adding it to $\boldsymbol{P}_{t-1}$. The impact of the product $\boldsymbol{A}\boldsymbol{q}_{t-1}$ on $\boldsymbol{P}_{t-1}$ is given by

$$\mathbf{Cov}(\boldsymbol{A}\mathcal{X}) = \mathbb{E}\left((\boldsymbol{A}\mathcal{X} - \mathbb{E}(\boldsymbol{A}\mathcal{X}))\left(\boldsymbol{A}\mathcal{X} - \mathbb{E}(\boldsymbol{A}\mathcal{X})\right)^T\right)$$
$$= \mathbb{E}\left((\boldsymbol{A}\mathcal{X} - \boldsymbol{A}\mathbb{E}(\mathcal{X}))\left(\boldsymbol{A}\mathcal{X} - \boldsymbol{A}\mathbb{E}(\mathcal{X})\right)^T\right) = \mathbb{E}\left(\boldsymbol{A}(\mathcal{X} - \mathbb{E}(\mathcal{X}))\left(\boldsymbol{A}(\mathcal{X} - \mathbb{E}(\mathcal{X}))\right)^T\right)$$
$$= \mathbb{E}\left(\boldsymbol{A}(\mathcal{X} - \mathbb{E}(\mathcal{X}))(\mathcal{X} - \mathbb{E}(\mathcal{X}))^T\boldsymbol{A}^T\right) = \boldsymbol{A}\,\mathbb{E}\left((\mathcal{X} - \mathbb{E}(\mathcal{X}))(\mathcal{X} - \mathbb{E}(\mathcal{X}))^T\right)\boldsymbol{A}^T$$
$$= \boldsymbol{A}\,\mathbf{Cov}(\mathcal{X})\,\boldsymbol{A}^T. \tag{4.37}$$

Combining all aspects yields the Kalman filter's equation for the predicted covariance $\overline{\boldsymbol{P}}_t$

$$\overline{\boldsymbol{P}}_t = \boldsymbol{A}\boldsymbol{P}_{t-1}\boldsymbol{A}^T + \boldsymbol{Q}, \tag{4.38}$$

that belongs to $\overline{\boldsymbol{q}}_t$ from (4.32). It is based on $\boldsymbol{A}\boldsymbol{P}_{t-1}\boldsymbol{A}^T$, to match the product $\boldsymbol{A}\boldsymbol{q}_{t-1}$, and the matrix $\boldsymbol{Q}$ includes the impact of the previously excluded random variable $\mathcal{X}$. Similar to figure 4.3, (4.32) and (4.38) adjust the center, size, and orientation of a normal distribution. Their combination thus resembles the Bayes filter's convolution step.

After predicting the next state based on known behavior, sensor observations are used to refine the believe, denoting the posterior. The combination of the prediction with the current measurements is performed in the same way as discussed in (4.13). If the prediction's covariance $\overline{\boldsymbol{P}}_t$ is more certain than the measurement's covariance $\boldsymbol{R}_t$, it is preferred, and vice versa. The linear mixing of both covariances is given by the Kalman gain $\boldsymbol{K}$, describing the ratio

$$\boldsymbol{K} = \overline{\boldsymbol{P}}_t\left(\overline{\boldsymbol{P}}_t + \boldsymbol{R}_t\right)^{-1}. \tag{4.39}$$

$\boldsymbol{K}$ is hereafter used to infer the new mean, by interpolating between prediction and measurement, and its covariance, by reducing the uncertainty based on the amount of mixing

$$\boldsymbol{q}_t = \overline{\boldsymbol{q}}_t + \boldsymbol{K}(\boldsymbol{o}_t - \overline{\boldsymbol{q}}_t), \quad \boldsymbol{P}_t = \overline{\boldsymbol{P}}_t - \boldsymbol{K}\overline{\boldsymbol{P}}_t. \tag{4.40}$$

For now, (4.39) and (4.40) strictly used the same calculations as the one dimensional case (4.13). However, just like the influences $\boldsymbol{u}_t$, the observations $\boldsymbol{o}_t$ might use different units than the state, refer only to some parts of it, or more than one part of it. Therefore, a translation between both is required. This adjustment is provided by another matrix, in literature often named $\boldsymbol{C}$ or $\boldsymbol{H}$, translating the predicted state $\overline{\boldsymbol{q}}_t$ into potential observations:

$$\boldsymbol{o}_t = \boldsymbol{C}\overline{\boldsymbol{q}}_t \; (+\boldsymbol{\mathcal{X}}), \quad \boldsymbol{\mathcal{X}} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_{\text{obs}}). \tag{4.41}$$

Again, the relationship denoted by (4.41) reads as *"assuming the object's current state is $\overline{\boldsymbol{q}}_t$, what should the sensors observe?"*. It is given by the mapping $\boldsymbol{C}$, converting from the state to the observations, including a known zero mean Gaussian sensor noise $\boldsymbol{\mathcal{X}}$. Again, the latter is only given to denote the presence of noise, but is not included within the calculation. Actual uncertainties of the current observation are denoted by the matrix $\boldsymbol{R}_t$ from (4.39). (4.39) and (4.40) thus are only correct for the special case of $\boldsymbol{C} = \boldsymbol{I}$, where the observations can be directly mapped onto the state, without adjustments. For a general solution, $\overline{\boldsymbol{q}}_t$ must be converted to measurement space using $\boldsymbol{C}\overline{\boldsymbol{q}}_t$, before it can be compared against the observations. The same holds true for their covariances. According to (4.37), the covariance $\overline{\boldsymbol{P}}_t$ from the state space is converted to the measurement space using $\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T$. The Kalman gain (4.39) hereafter reads as

$$\boldsymbol{K}' = \boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T \left(\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1}. \tag{4.42}$$

Due to the impact of $\boldsymbol{C}$, the result of (4.42) uses the coordinate system of the measurements, and thus the measurement space. As the Kalman gain is used to adjust the state, and its covariance, the result must be mapped from the measurement space back to the state space using $\boldsymbol{C}^{-1}$:

$$\boldsymbol{q}_t = \overline{\boldsymbol{q}}_t + \overbrace{\boldsymbol{C}^{-1}\underbrace{\boldsymbol{K}'(\boldsymbol{o}_t - \boldsymbol{C}\overline{\boldsymbol{q}}_t)}_{\text{measurement space}}}^{\text{state space}}, \quad \boldsymbol{P}_t = \overline{\boldsymbol{P}}_t - \overbrace{\boldsymbol{C}^{-1}\left(\underbrace{\boldsymbol{K}'(\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T)}_{\text{measurement space}}\right)\boldsymbol{C}^{-1^T}}^{\text{state space}} \tag{4.43}$$

$$= \overline{\boldsymbol{P}}_t - \boldsymbol{C}^{-1}\boldsymbol{K}'\boldsymbol{C}\overline{\boldsymbol{P}}_t.$$

However, when $\boldsymbol{C}$ is not a square matrix, e.g. when only some of the attributes are observed by sensors, $\boldsymbol{C}^{-1}$ is undefined. While this can be addressed e.g. by using the *Moore-Penrose inverse* [Pen55] instead, this introduces unnecessary roundoff errors. Therefore, the problem is addressed by moving $\boldsymbol{C}^{-1}$ into the calculation of the Kalman gain $\boldsymbol{K}'$ from (4.42), which cancels with the first $\boldsymbol{C}$. This step finally yields the well-known Kalman filter equations for the measurement update step, where the Kalman gain is given by

$$\boldsymbol{K} = \boldsymbol{C}^{-1}\boldsymbol{K}' = \boldsymbol{C}^{-1}\left(\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T\left(\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1}\right) = \overline{\boldsymbol{P}}_t\boldsymbol{C}^T\left(\boldsymbol{C}\overline{\boldsymbol{P}}_t\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1}, \tag{4.44}$$

and is used to derive the posterior's mean and covariance

$$\boldsymbol{q}_t = \overline{\boldsymbol{q}}_t + \boldsymbol{K}(\boldsymbol{o}_t - \boldsymbol{C}\overline{\boldsymbol{q}}_t), \quad \boldsymbol{P}_t = \overline{\boldsymbol{P}}_t - \boldsymbol{K}(\boldsymbol{C}\overline{\boldsymbol{P}}_t) = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C})\,\overline{\boldsymbol{P}}_t. \tag{4.45}$$

Just like for the 1D case, the posterior's mean ranges somewhere in between the prediction and the measurement, depending on the ratio $\boldsymbol{K}$ of the uncertainties. Similarly, the overall uncertainty is reduced based on this ratio, refining the posterior over time.

Reverting to the example (4.34) of estimating an object's 1D location and velocity, the Kalman filter requires the initial believe $\mathrm{bel}(\boldsymbol{q}_0)$ as a starting point. In case it is unknown, it is assumed to be distributed uniformly throughout the whole state space, which is approximated by $\mathcal{N}(\boldsymbol{0}, \infty)$, or, in practice, some large numbers. Assuming that the initial location x of the example object is approximately known, but its velocity $\dot{x}$ is not

$$\boldsymbol{q}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}, \tag{4.46}$$

and there are no known or unknown influences. That is, the prediction only depends on the estimated state itself, and is assumed to be exact:

$$\boldsymbol{u}_t = \begin{pmatrix} 0 \end{pmatrix}, \quad \boldsymbol{Q} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{4.47}$$

The first prediction after $\Delta t = 1\,\mathrm{s}$ will not affect the mean, as the currently estimated velocity $\dot{x}$ is still 0, causing no change in position. The state's covariance, however, is updated:

$$\overline{\boldsymbol{P}}_1 = \boldsymbol{A}\boldsymbol{P}_0\boldsymbol{A}^T + \boldsymbol{Q} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \Delta t & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 101 & 100 \\ 100 & 100 \end{pmatrix}. \tag{4.48}$$

As can be seen in (4.48), the uncertainty in velocity directly creates an uncertainty in location, as the location depends on the current velocity, and thus its variance. Now, the object is actually moving at $5\,\mathrm{m/s}$, and a single sensor observes the current location x, at a variance of $2\,\mathrm{m}$. The first observation takes place after $\Delta t = 1\,\mathrm{s}$, where the object resides at $x = 5\,\mathrm{m}$, and is related to the predicted state $\overline{\boldsymbol{q}}_1$ via $\boldsymbol{C}$

$$\boldsymbol{o}_1 = \begin{pmatrix} 5 \end{pmatrix}, \quad \boldsymbol{R} = \begin{pmatrix} 2 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 1 & 0 \end{pmatrix}. \tag{4.49}$$

The Kalman gain $\boldsymbol{K}$ denotes the ratio between predicted covariance and measurement noise

$$\boldsymbol{K}_1 = \overline{\boldsymbol{P}}_1\boldsymbol{C}^T \left(\boldsymbol{C}\overline{\boldsymbol{P}}_1\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1} \approx \begin{pmatrix} 0.98 \\ 0.97 \end{pmatrix}, \tag{4.50}$$

that is, the ratio between 101 and $101 + 2$ for the location x, but in state space, including the covariance between location and velocity. Using $\boldsymbol{K}$ to estimate the posterior then yields

$$\boldsymbol{q}_1 = \overline{\boldsymbol{q}}_1 + \boldsymbol{K}\left(\boldsymbol{o}_1 - \boldsymbol{C}\overline{\boldsymbol{q}}_1\right) \approx \begin{pmatrix} 4.90 \\ 4.85 \end{pmatrix}, \quad \boldsymbol{P}_1 = \overline{\boldsymbol{P}}_1 - \boldsymbol{K}\boldsymbol{C}\overline{\boldsymbol{P}}_1 \approx \begin{pmatrix} 1.96 & 1.94 \\ 1.94 & 2.91 \end{pmatrix}. \quad (4.51)$$

(4.51) depicts two important aspects of the Kalman filter. Even though the initial velocity $\dot{x}$ was unknown, assumed 0, and not observed by any sensor, the filter was able to estimate the current velocity based on the covariance between location and velocity. Furthermore, even though the initial variance estimation for the velocity was 100, it dropped to 2.91 after a single measurement update, which did not measure the velocity itself.

While the Kalman filter provides a robust estimation of the unknown state after only a few measurement updates, it only supports for linear predictions and measurement updates. When angles and trigonometric functions come into play, the Kalman filter can not be used as is, due to their nonlinear behavior. Regarding the localization context, this relates to predictions where an angular heading is involved, or measurement updates that contain angular information, such as radar and LIDAR [SCI13; TBF05]. For those cases, adjustments are required.

4.4   Extended Kalman Filter

As discussed within the previous section, linear transformations of a normal distribution generate another normal distribution, which is scaled and shifted by constants. However, not all predictions and measurement updates can be described using linear systems. As soon as functions like sine and cosine are involved, the system becomes nonlinear. For most navigation purposes, however, those functions are mandatory, e.g. to adjust an object's current whereabouts based on the estimated heading and speed, as mentioned within chapter 3

$$\begin{aligned} x_t &= x_{t-1} &+& \quad v_{t-1}\cos(\Theta_{t-1}) \\ y_t &= y_{t-1} &+& \quad v_{t-1}\sin(\Theta_{t-1}). \end{aligned} \quad (4.52)$$

When such nonlinear predictions are applied to a Gaussian prior, the resulting predicted believe will rarely be distributed normally around the mean value [TBF05]. This behavior is depicted in figure 4.5, where the Gaussian prior from the upper left is adjusted by three different transformation functions for comparison

$$f_a(x) = 2x + 2 \qquad f_b(x) = \begin{cases} x & |x| < 1 \\ 4x - 3\,\mathrm{sgn}(x) & \text{else} \end{cases} \qquad f_c(x) = x + \frac{3}{4}\cos(x). \quad (4.53)$$
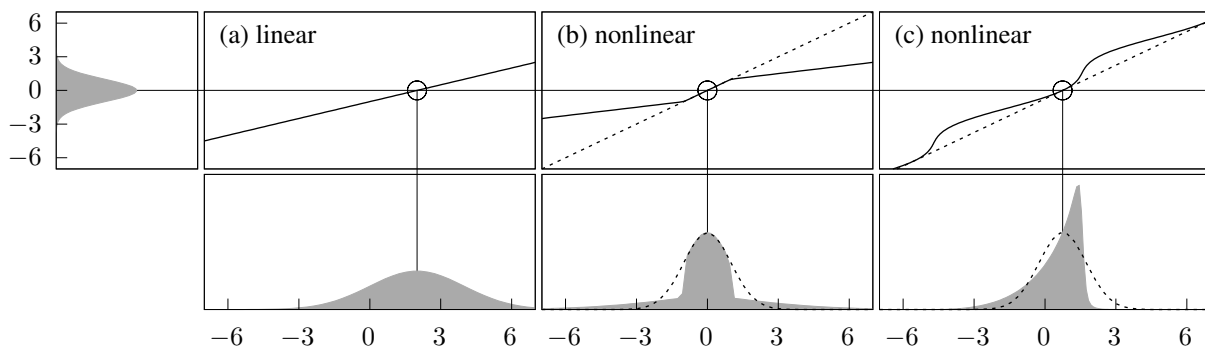
Figure 4.5: Linear and nonlinear modification of a Gaussian prior (upper left). If the density is modified by a linear transformation (a), the resulting distribution is another Gaussian. The nonlinear transformations of (b) and (c) yield a non Gaussian result. Here, a Gaussian can be approximated (dashed curves) by linearizing the transformation (dashed lines). Adapted from [TBF05, p. 57] figure 3.4.

When the linear transformation $f_a(x)$, shown in figure 4.5a, is applied to the Gaussian prior, it is shifted and stretched by a factor of 2, and the result represents another Gaussian. The transformation $f_b(x)$ from figure 4.5b uses a conditional combination of two linear transformations, thus being discontinuous. While each of the two just stretches the prior, the overall result is non Gaussian. The same holds true for $f_c(x)$, shown in figure 4.5c. The impact of $\cos(x)$ moves the density's mass towards one direction, producing a non Gaussian result. Transformations such as $f_b(x)$ and $f_c(x)$ thus can not be used within the Kalman filter.

This drawback is addressed by the extended Kalman filter (EKF) [SSM62; Jaz70]. Compared to the regular Kalman filter, it allows for arbitrary prediction and measurement update functions, but still relies on Gaussian distributions. Switching from linear to arbitrary functions within the prediction and measurement update step is achieved by replacing the matrix multiplications with two functions, often referred to as $\boldsymbol{g}()$ and $\boldsymbol{h}()$:

$$
\begin{aligned}
\overline{\boldsymbol{q}}_t = \overbrace{\boldsymbol{A}\boldsymbol{q}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t}^{\text{linear}} \quad (+\boldsymbol{\mathcal{X}}) &\Rightarrow \quad \overline{\boldsymbol{q}}_t = \overbrace{\boldsymbol{g}(\boldsymbol{q}_{t-1}, \boldsymbol{u}_t)}^{\text{nonlinear}} \quad (+\boldsymbol{\mathcal{X}}) \\
\boldsymbol{o}_t = \boldsymbol{C}\overline{\boldsymbol{q}}_t \quad (+\boldsymbol{\mathcal{X}}) &\Rightarrow \quad \boldsymbol{o}_t = \boldsymbol{h}(\overline{\boldsymbol{q}}_t) \quad (+\boldsymbol{\mathcal{X}}).
\end{aligned}
\tag{4.54}
$$

The new mean $\overline{\boldsymbol{q}}_t$ is the result of a prediction function $\boldsymbol{g}()$, and is directly mapped onto the observations using a translation $\boldsymbol{h}()$. By replacing the linear matrix equations with functions, a general solution for arbitrary models becomes available. However, with the underlying distribution still being a multivariate normal distribution, the previous calculations of $\overline{\boldsymbol{P}}_t$ (4.38) and $\boldsymbol{K}$ (4.44) are no longer valid. With the linear matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ replaced by nonlinear adjustments, the uncertainty becomes distorted, and is no longer a Gaussian (cf. figure 4.5).

The main idea behind the EKF is to approximate the two matrices $\boldsymbol{A}$ and $\boldsymbol{C}$, based on the functions $\boldsymbol{g}()$ and $\boldsymbol{h}()$, at the drawback of accuracy [Do+09; SSM62]. Both matrices denote

how a covariance is changed, when an underlying random variable is transformed by them. So do the two newly introduced functions $g()$ and $h()$. Except that their change is not necessarily linear (cf. figure 4.5). However, by assuming them to behave linearly throughout the whole range, that is, enforcing linearization, A and C can be approximated. As the covariance is unaffected by addition of constants, the approximation is given by the partial derivative of $g()$ and $h()$, with respect to each output, that is, their Jacobian matrix [BC12]

$$J = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} + \ldots + \frac{\partial f_1(x)}{\partial x_j} \\ \vdots \\ \frac{\partial f_i(x)}{\partial x_1} + \ldots + \frac{\partial f_i(x)}{\partial x_j} \end{pmatrix}, \quad \frac{\partial f(x)}{\partial x_j} = \lim_{\Delta \to 0} \frac{f(x_1, \ldots, x_j + \Delta, \ldots) - f(x)}{\Delta} \quad (4.55)$$

Like A and C, their Jacobian describes each dimension's change in output depending on a change in input. Similar to a linear tangent around a center point, known from the Taylor series [Tay15]. Regarding the prediction and measurement update of the EKF, the smaller the adjustments made by $g(q_{t-1}, u_t)$ and $h(\overline{q}_t)$, the more accurate the assumption of the linear behavior is. The same holds true for the size of the covariances, $\det(P_t)$ and $\det(R)$, yielding more accurate approximations for smaller values. As can be seen in (4.55), even if the partial derivative can not be determined analytically, it can be approximated numerically [TBF05].

For the EKF, the two matrices A and C from the Kalman filter are thus replaced by

$$A = \begin{pmatrix} \frac{\partial g_1(q_{t-1}, u_t)}{\partial q_1} & \cdots & \frac{\partial g_1(q_{t-1}, u_t)}{\partial q_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_i(q_{t-1}, u_t)}{\partial q_1} & \cdots & \frac{\partial g_i(q_{t-1}, u_t)}{\partial q_j} \end{pmatrix}, \quad C = \begin{pmatrix} \frac{\partial h_1(\overline{q}_t)}{\partial \overline{q}_1} & \cdots & \frac{\partial h_1(\overline{q}_t)}{\partial \overline{q}_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_i(\overline{q}_t)}{\partial \overline{q}_1} & \cdots & \frac{\partial h_i(\overline{q}_t)}{\partial \overline{q}_j} \end{pmatrix}, \quad (4.56)$$

which are then often referred to as F and H, respectively [GA01; Sär13]. For linear models $g()$ and $h()$, such as the presented example from (4.33) and (4.34), the content of the two Jacobians is identical to the Kalman filter's original matrices:

$$\frac{\partial x_t}{\partial x_{t-1}} = 1, \quad \frac{\partial x_t}{\partial \dot{x}_{t-1}} = \Delta t, \quad \frac{\partial \dot{x}_t}{\partial x_{t-1}} = 0, \quad \frac{\partial \dot{x}_t}{\partial \dot{x}_{t-1}} = 1 \quad \Rightarrow \quad A = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}. \quad (4.57)$$

The extended Kalman filter allows for combining multiple sensors and prediction models with nonlinear behavior, while still maintaining the benefits of fast, analytical calculations. This puts the filter to a broad use, from electronic battery impedance observing [Do+09], to multicopter location estimation [Gar+16]. Especially for the latter, the EKF presents an ideal candidate, as potential movements outdoors are unlimited by surroundings. Here, the assumption of unimodal

normal distributions provides an efficient analytical solution, to combine multiple sensors with predictable behavior. Indoors, however, potential movements are constrained by architecture, often yielding multimodalities and discontinuities, discussed in chapter 3. Furthermore, compared to controlled vehicles, the movement of pedestrians is much more uncertain, requiring the prediction to predict all potential variants. For such use cases, Kalman filters are too restricted in terms of predictions and density modeling, requiring for different approaches.

4.5 Particle Filter

Working based on a single, multivariate normal distribution is the (extended) Kalman filter's key for fast, analytical calculations. Yet, this also represents its strongest drawback. For many real-world scenarios, the state space is constrained, requiring multimodal, arbitrarily shaped distributions. Referring to the models discussed in chapter 2 and 3, it becomes clear that the combination of a building's architecture, and human walking behavior, often can not be represented by such simple distributions. The predicted density must not propagate through impassable objects, such as walls, and must be able to divide itself, when e.g. more than one walking direction is likely. The Kalman filter and its variants thus can not be used for most of the probabilistic prediction and measurement update models discussed in chapter 2 and chapter 3, as they rely on more complex density functions than a single normal distribution.

Besides the Kalman filter as analytical implementation of the Bayes filter, there are other, sometimes non-analytical, representatives, allowing for arbitrary density functions. One pool of approaches is given by the so called *Monte Carlo* algorithms, named and described by Stanislaw Ulam, John von Neumann and Nicholas Metropolis [Met87] in the mid 1940s. The method was developed during World War II, to solve problems related to thermonuclear reactions, where many calculations either could not be formulated in an analytical manner, or were too cumbersome to calculate. Due to the advent of the first electronic computers, such as the *ENIAC*, simulations of physical circumstances, such as the behavior of *particles*, like neutrons, became possible. The results of the simulations were used to gather details on processes that could not be described in an analytical way. Here, a particle was a single entity with some state and behavior, stored on a punched card [MU49]. The simulation used multiple particles, stored on several punched cards, representing a sort of initial state. The cards were fed into the computation unit, applying known physical models based on the parameters stored for each particle, to derive a new set of particles. To estimate the result after a certain amount of time, this step was repeated. Seeming slow at first glance, as described by Metropolis and Ulam, this type of simulation can be highly parallelized. Every particle can be simulated on its own, whenever the problem formulation assumes independence. This, however, is not always a given [Del98].
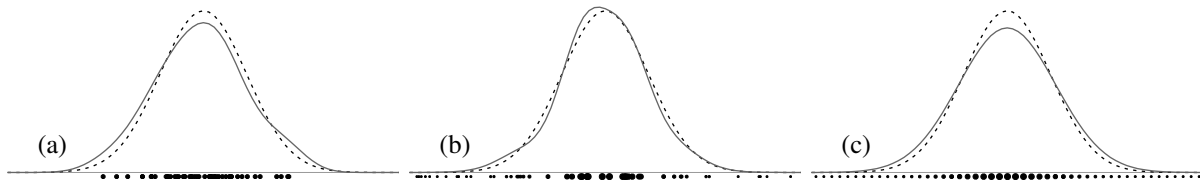
Figure 4.6: Approximation of a Gaussian (dashed line) using 50 discrete samples (dots). Samples can be placed according to the underlying probabilities, yielding a concentration based on the distribution (a). Alternatively, samples can be distributed uniformly throughout the whole state space, and are hereafter weighted (dot size) based on the underlying probability, shown in (b) and (c). A continuous representation (solid line) is provided by applying a kernel density estimation.

Before simulating complex physical behavior, Ulam used the Monte Carlo method for a statistical analysis on the probability of winning the *Canfiled solitaire* card game. While thinking about a combinatorial solution to the problem, he envisioned a more practical approach. It was given by simulating multiple games based on random moves, and counting the number of positive outcomes, using the power of the first computers [Eck87]. As postulated by the central limit theorem, for an infinite number of repetitions, such random experiments will yield an almost exact result. For a limited number of runs, the quality of the approximated result depends on both, the number of runs, and the complexity of the simulated problem. The Monte Carlo method thus allows for an approximation of various types of problems, and the name relates to the eponymous casino, gambling, effects of randomness, simulation and approximation.

Besides simulating experimental results, this approach can also be used to approximate arbitrary density functions. Similarly to estimating the chance of winning the Canfield Solitaire, a density can be approximated by a limited number of random samples from it [Neu51]. This is depicted in figure 4.6. Explained in detail later, a set of *equally important* random samples is distributed based on the probability of the density. That is, more likely regions receive more samples, and vice versa (cf. figure 4.6a). Assuming a sufficiently large number of samples, their histogram resembles the underlying distribution. For a continuous solution, the discrete samples can be applied to a KDE (cf. section 2.7) which can be thought of as a smoothed, interpolated histogram. As discussed, this approximation requires the random samples to follow the behavior of the to-be-approximated density. While describing a uniform distribution by generating uniformly distributed samples within a certain range is trivial, generating non uniform samples based on an arbitrary PDF is not [Neu51].

A solution is given by drawing samples uniformly throughout the whole *domain* of the PDF, but hereafter *weighting* them based on the PDF's probability, at each drawn location. When using the weighted samples within the histogram or the KDE, the same result is achieved, shown in figure 4.6b and 4.6c. However, this approach usually requires more samples, as the whole
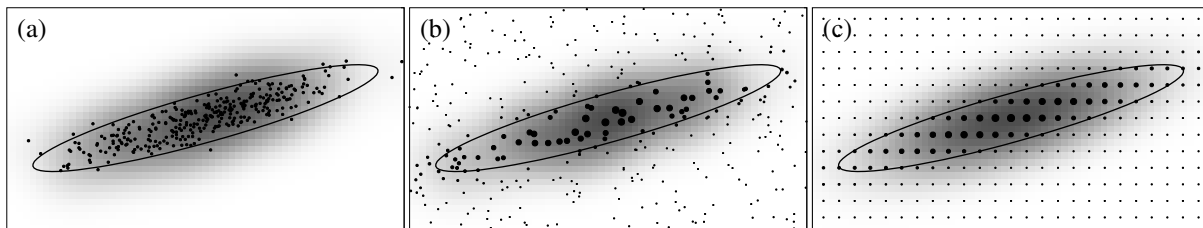
Figure 4.7: Approximation of a multivariate Gaussian (ellipse $= 2\sigma$) using 325 discrete samples (dots). Samples can be placed according to the underlying probabilities, yielding a concentration based on the distribution (a). Alternatively, samples can be distributed uniformly throughout the whole state space, hereafter weighted (dot size) based on the underlying probability, shown in (b) and (c). A continuous representation is provided by applying a kernel density estimation, shown as gray background.

state space of the distribution, even unlikely regions, must be covered. This is especially cumbersome for unbounded PDFs, such as the normal distribution, ranging between $\pm\infty$. Uniform samples can be created either in a random pattern (see figure 4.6b), or using an ordered, equidistant spacing, shown in 4.6c. For both, the size of the samples depicted as dots represents their weight. It is given by the to-be-approximated PDF, shown as dashed line. As can be seen within figure 4.6b and 4.6c, many of the samples are located within unlikely areas, receiving infinitesimally small weights. For 4.6a, where samples are equally important, and drawn based on the distribution, this effect does not occur. While 4.6c yields a smooth symmetric result for the underlying density, equidistant spacing is rather inflexible, especially for large, high-dimensional state spaces. Yet, the same holds true for 4.6b. Uniform sampling of high-dimensional problems with large domain ranges will yield many samples belonging to insignificant regions, as the probability mass often concentrates within a small area. Regarding indoor localization, the state space is e.g. given by all possible 3D locations within a building. Depending on its size, numerous samples are required, even though pedestrian whereabouts are expected to be focused on a small region. This drawback can already be observed for two dimensions, depicted in figure 4.7. For the uniformly distributed and weighted samples in 4.7b and 4.7c, only $\approx 16\,\%$ belong to the 2σ region of the underlying multivariate normal distribution, indicated by an ellipse. Whereas, by definition, $95\,\%$ of the samples from 4.7a reside within the 2σ contour.

Within the following discussions on a Monte Carlo-based implementation of the Bayes filter, the mentioned approaches are used to derive the results of prediction, measurement update, and thus the posterior. One state of the art representative for nonlinear and non Gaussian estimation based on Monte Carlo methods is the *particle filter* [Del96], also referred to as *sequential Monte Carlo* method [LC98]. It uses a set of weighted samples to approximate the Bayes filter's PDF of the unknown state, based on the history of all observations and influences

$$p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1}, \boldsymbol{u}_{1:t}) \quad \rightarrow \quad \left\{ \left(\boldsymbol{q}_{t,[i]}, w_{t,[i]} \right) \right\}_{i=1}^{N} . \tag{4.58}$$

In (4.58), $\boldsymbol{q}_{t,[i]}$ represents one potential state at time t including its weight $w_{t,[i]}$. The overall density is approximated by N of such weighted samples, referred to as *particles*. For this representation to denote a proper PDF, the constraint

$$\int_{-\infty}^{+\infty} p(\boldsymbol{q}_t \mid \boldsymbol{o}_{1:t-1}, \boldsymbol{u}_{1:t}) \, \mathrm{d}\boldsymbol{q}_t \overset{!}{=} 1 \quad \rightarrow \quad \sum_{i=1}^{N} w_{t,[i]} \overset{!}{=} 1 \, . \tag{4.59}$$

must be satisfied. Like earlier with fingerprinting (cf. section 2.7), given this discrete approximation of several weighted samples, a continuous representation can be derived, e.g. by applying a multivariate KDE, adding a small uncertainty around every individual sample

$$\begin{aligned} p(\boldsymbol{q}_t{}' \mid \boldsymbol{o}_{1:t-1}, \boldsymbol{u}_{1:t}) \; &\approx \; \mathrm{kde}\left(\boldsymbol{q}_t{}', \left\{\left(\boldsymbol{q}_{t,[i]}, w_{t,[i]}\right)\right\}_{i=1}^{N}\right) \\ &= \frac{1}{N} \sum_{i=1}^{N} w_{t,[i]} \frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{q}_{t,[i]} - \boldsymbol{q}_t{}'\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{q}_{t,[i]} - \boldsymbol{q}_t{}'\right)\right)}{\sqrt{(2\pi)^{\dim} \det(\boldsymbol{\Sigma})}} \, . \end{aligned} \tag{4.60}$$

$\boldsymbol{\Sigma}$ adds a certain probability around each individual sample. The multivariate distribution also fits the individual ranges of the state parameters, like location (x, y, z) and heading Θ. An example result is shown in figure 4.7, with the KDE as dark background. As can be seen, the KDE of all three presented sampling approaches is almost identical.

To implement the Bayes filter, the particle filter must approximate the density presented in (4.26). Ideally, the algorithm uses an existing set of weighted samples, the prior, and, for every step, derives a new set, with its samples distributed according to (4.26), including the prediction and measurement update. However, analytical solutions to draw from such a mixture of distributions rarely exist [RC01]. While they can be approximated, e.g. by using a KDE on the prior set of samples, and randomly creating new samples based on the prediction and measurement probability, this involves sampling from the whole state space and thus is inefficient.

Literature suggests a solution to this problem, which is similar to figure 4.7b. If sampling directly from the Bayes filter's posterior is impossible, samples are taken from another distribution, referred to as *proposal*, and hereafter weighted based on the posterior. While using a uniform distribution as proposal is possible, just like shown in previous figures, this yields many samples within areas of low probabilities, thus requiring more samples. Therefore, the proposal should be close to the posterior, to yield many samples within areas of high probabilities.

For uniformly distributed proposals, the samples are weighted solely by the probability of the posterior, just as shown within the figures. For all other cases, the proposal is biased, and the weights can not be used directly, as the bias has to be removed beforehand. In general, when samples are drawn according to $p_{\mathrm{b}}(\boldsymbol{q}_{t,[i]})$ but should be distributed based on $p_{\mathrm{a}}(\boldsymbol{q}_{t,[i]})$, the bias

of $p_{\mathrm{b}}(\boldsymbol{q}_{t,[i]})$ can be removed by division [RC01; GSS93; Aru+01; TBF05]

$$w_{t,[i]} = \frac{p_{\mathrm{a}}(\boldsymbol{q}_{t,[i]})}{p_{\mathrm{b}}(\boldsymbol{q}_{t,[i]})} = \frac{\text{posterior}}{\text{proposal}} \, . \tag{4.61}$$

While Doucet et al. [DGK01] have shown that an optimal proposal distribution exists, and is given by the one that minimizes the variance among all $w_{t,[i]}$ from (4.61), this approach is not efficient for most real-world scenarios [Aru+01]. Therefore, many different proposal distributions are discussed in literature, ranging from simple approaches towards combinations with the Kalman filter, and hybrid techniques, often dependent on explicit use cases [RC01; Wan+11a]. An important aspect regarding this choice is given by the way the proposal and posterior are used during the calculation: It must be possible to draw random samples from the proposal. Also, both, proposal and posterior, must be evaluable, to calculate (4.61).

One common choice for the proposal distribution thus uses the prediction model [RC01; Wan+11a; Aru+01]. Regarding (4.61), this yields the following weight update function:

$$\begin{aligned} w &= \frac{\text{posterior}}{\text{proposal}} = \frac{\text{bel}(\boldsymbol{q}_t)}{\overline{\text{bel}}(\boldsymbol{q}_t)} \\ &= \frac{\eta p(\boldsymbol{o}_t \mid \overline{\boldsymbol{q}}_t) \int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{u}_t) \text{bel}(\boldsymbol{q}_{t-1}) \, \mathrm{d}\boldsymbol{q}_{t-1}}{\int p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1}, \boldsymbol{u}_t) \text{bel}(\boldsymbol{q}_{t-1}) \, \mathrm{d}\boldsymbol{q}_{t-1}} = \eta p(\boldsymbol{o}_t \mid \overline{\boldsymbol{q}}_t) \, . \end{aligned} \tag{4.62}$$

As the filter works with discrete samples from those distributions, it is rewritten as

$$w_{t,[i]} = \eta p \left(\boldsymbol{o}_t \mid \overline{\boldsymbol{q}}_{t,[i]} \right) \, . \tag{4.63}$$

By using the prediction as proposal, only the measurement probability is required to be evaluable. The prediction is not evaluated, but must support for drawing random samples from it. As can be seen, the resulting setup works similar to the Kalman filter, but with discrete samples. Based on a set of weighted samples, denoting the prior, a new set of samples is created, representing potential states after the prediction step, increasing uncertainty. This set is reweighted, based on the current sensor observations, decreasing uncertainty. The combination of drawing and weighing creates the posterior, as shown within previous figures.

This setup is often referred to as *bootstrap filter* [GSS93] or *CONDENSATION algorithm* [IB98]. The movement models discussed in chapter 3 were designed with this kind of setup in mind: All of them support drawing random samples, but most of them can not be evaluated, as their results are also created by some Monte Carlo-like random process, such as the random walks along a grid. While using the prediction as proposal is a convenient solution, it represents one important drawback: Random samples are drawn solely based on the prediction, without

considering the current measurements, yielding samples only in regions where the prediction is likely. As long as the prediction is certain, samples are placed ideally. If, however, the prediction is uncertain, and does not match with the sensor observations, samples are placed inefficiently within unlikely regions, resulting in a poor sampling of the posterior [RC01]. Due to its analytical nature, the Kalman filter does not suffer from such problems. All following discussions are focused on a particle filter using the prediction as proposal distribution.

In particle filter literature, the prediction and measurement update steps are also referred to as *sampling/transition* and *weight-update/evaluation* [Del96; LC98; GSS93; TBF05]. Their implementation is explained best by using an example: Assuming a dynamic object, moving in a direction Θ in 2D space (x, y), with the unknown state given by (x, y, Θ). When the initial state $\boldsymbol{q}_0$ is known, the Kalman filter uses the known attributes as mean, and assigns corresponding variances, depending on the expected accuracy. If the initial state is unknown, a zero mean Gaussian with infinite variances approximates a uniform likelihood of all potential whereabouts. Within the particle filter, unknown initial states are modeled by uniformly distributing random samples throughout the whole state space, all with the same weight

$$p(\boldsymbol{q}_0) \rightarrow \left\{ \left(\boldsymbol{q}_{0,[i]}, w_{0,[i]} \right) \right\}_{i=1}^{N} , \quad \langle \boldsymbol{q} \rangle_{t,[]} = \langle (x, y, \Theta) \rangle_{t,[]}$$

$$q_{0,[i]}^{(x)} \sim \mathcal{U}\left(\ldots, \ldots\right) , \quad q_{0,[i]}^{(y)} \sim \mathcal{U}\left(\ldots, \ldots\right) , \quad q_{0,[i]}^{(\Theta)} \sim \mathcal{U}\left(0, 2\pi\right) , \quad w_{0,[i]} = \frac{1}{N} . \tag{4.64}$$

As can be seen in (4.64), compared to the Kalman filter, the particle filter requires discrete bounds for the two uniform distributions for x and y. Furthermore, depending on this range, more samples might be needed, to accurately cover the whole state space (cf. figure 4.7). If the initial state is well known, all samples can be initialized with the known value instead.

This set of weighted samples serves as an approximation of the unknown state's initial distribution, which is then subject to the prediction step. The Bayes filter's prediction (4.25) denoted something similar to a convolution, where every potential prior is adjusted by a prediction model. For the particle filter, (4.25) is reinterpreted as

$$\overline{\mathrm{bel}}(\boldsymbol{q}_t) = \frac{1}{N} \sum_{i=1}^{N} p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1,[i]}, \boldsymbol{u}_t) \, w_{t-1,[i]} , \tag{4.65}$$

where every weighted sample $(\boldsymbol{q}_{t-1,[i]}, w_{t-1,[i]})$ from the prior contributes to the prediction, dependent on its weight $w_{t-1,[i]}$ and transition probability $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1,[i]}, \boldsymbol{u}_t)$. As discussed, this density is approximated using a new set of random samples, which are drawn based on this likelihood, therefore referred to as *importance sampling* [LC98; TBF05]. Regarding discrete convolutions, this is similar to creating N new samples per input $\boldsymbol{q}_{t-1,[i]}$, based on a convolution kernel given by $p(\boldsymbol{q}_t \mid \boldsymbol{q}_{t-1,[i]}, \boldsymbol{u}_t)$. This, however, increases the number of samples after every

prediction, which is inconvenient for practical use. Due to required memory and computation time, the number of new samples must be reduced. As described previously, a density is approximated by N samples, with quality increasing for larger N. Thus, 1 new sample for every of the N input samples is an approximation, but a valid decision. That is, every sample $\boldsymbol{q}_{t-1,[i]}$ yields exactly one predicted sample $\boldsymbol{q}_{t,[i]}$, distributed according to $p(\boldsymbol{q}_{t,[i]} \mid \boldsymbol{q}_{t-1,[i]}, \boldsymbol{u}_t)$. All resulting samples combined denote the proposal distribution.

4.5.1 Random Sampling

In general, there are various ways to sample from a distribution. Most of which start with a uniformly distributed value, as this is well-researched and can be provided by dedicated hardware components [HD62; SMV15]. This uniform value is hereafter adjusted in some nonlinear way that the distribution of several repetitions approximates the underlying PDF. In general, this effect can be achieved by using a function's inverse [Met87]. Regarding distributions, this only applies to the exponential distribution [Kne18], where the inverse function can be used to convert a uniformly distributed random variable $\mathcal{Y} \sim \mathcal{U}(0, \max_x f(x, \lambda))$, into a random variable $\mathcal{X}$, that represents the exponential PDF. This is due to the exponential distribution's similarity between its PDF and cumulative distribution function (CDF)

$$P(x) = \int_{-\infty}^{x} p(x') \, \mathrm{d}x' \quad \text{with} \quad P(\infty) \overset{!}{=} 1 \,. \tag{4.66}$$

The CDF represents a general solution for sampling analytical densities, as its codomain stays within $[0, 1]$, and the function is unique. Using a uniform random variable $\mathcal{Y} \sim \mathcal{U}(0, 1)$, the inverse $P^{-1}()$ can be used to create $\mathcal{X}$, which resembles the behavior of the corresponding PDF [Neu51]. This technique is referred to as *inverse transform sampling* [Kne18]. As the corresponding output of the CDF is unique, this approach also works for symmetric PDFs. However, for many distributions, like the normal distribution and discontinuous representatives, the CDF can not be described analytically, and/or the inverse is undefined due to non-invertible functions, such as the factorial $n!$. To achieve invertibility, additional approximations, such as *Stirling's equation* for the factorial [Moi56], would be required.

While this drawback can be addressed by using a discrete, precomputed lookup table (LUT) to draw from, doing so imposes new drawbacks: Within the LUT, the keys are usually placed equidistant, and the input range [0, 1] is divided linearly into equally sized chunks. The output range, following the sampled distribution, almost always follows a nonlinear behavior. Regarding the inverse CDF of a normal distribution, a change around $x \approx 0.5$ yields almost no change in the output of $y = P^{-1}(x)$. Near $x \approx 0$ and $x \approx 1$, however, $y = P^{-1}(x)$ changes dramat-
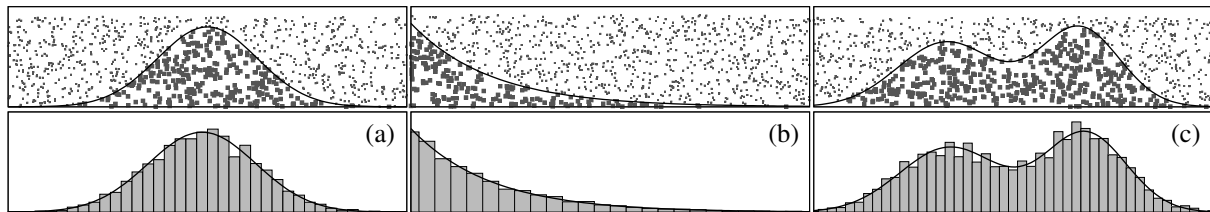
Figure 4.8: Rejection sampling for three probability density functions. The histogram of all accepted samples (large dots) resembles the underlying PDF. While this works for arbitrary distributions, the computation time depends on the number of outliers (small dots) during the sampling.

ically, converging towards $\pm\infty$. Within those regions, the LUT suffers from two aspects: For small changes in x, which should yield fairly large changes in y, the LUT will often provide the same output. Furthermore, the contents of the LUT are limited to a discrete range. The values provided for $P^{-1}(0)$ and $P^{-1}(1)$ are thus discretely limited, not resembling the infinite behavior of a real normal distribution. While these drawbacks can somewhat be mitigated using interpolation and adding explicit entries for $x = 0 \to -\infty$ and $x = 1 \to \infty$, such adjustments strongly depend on the underlying distribution, and are not generally valid.

A practical solution, inspired by Monte Carlo methods, is *rejection sampling* [Neu51]. It works for every density where the PDF can be evaluated, even if the function is multivariate, discontinuous, or not given analytically. For the one dimensional case, two uniformly distributed random variables are drawn, and compared against each other using the PDF

$$
\begin{aligned}
\mathcal{X} &\sim \mathcal{U}\left(-\infty, +\infty\right) \\
\mathcal{Y} &\sim \mathcal{U}\left(0, 1\right)
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
\mathcal{Z} &= \{x \in \mathcal{X} \mid y \in \mathcal{Y} \leq \eta\, p(x)\} \\
\mathcal{Z} &\sim p(x),
\end{aligned}
\tag{4.67}
$$

where η is used as a scaling factor, to ensure that PDF is mapped to the $[0, 1]$ interval of the random variable $\mathcal{Y}$, enforcing $\eta\, p(x) \leq 1\, \forall x$. Usually, only a fraction $[\min x, \max x]$ of the PDF holds meaningful results. When this range is known, above equation is approximated by

$$
\begin{aligned}
\mathcal{X} &\sim \mathcal{U}\left(\min x, \max x\right) \\
\mathcal{Y} &\sim \mathcal{U}\left(0, \max_x p(x)\right)
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
\mathcal{Z} &= \{x \in \mathcal{X} \mid y \in \mathcal{Y} \leq p(x)\} \\
\mathcal{Z} &\sim p(x).
\end{aligned}
\tag{4.68}
$$

(4.68) leads to a visual representation of rejection sampling, shown in figure 4.8. x is drawn from the PDF's *domain*, and y from its *codomain*. Accepted are all samples that reside below the curve described by the PDF. Just requiring the latter to be calculable, this allows for sampling from arbitrary, discontinuous, multivariate PDFs. The drawback is required computation time. Most of the drawn pairs will reside outside of the curve, and thus are rejected. Furthermore, unbounded intervals between $\pm\infty$ can not be used practically, requiring for discrete limits.

All of above approaches exhibit considerable drawbacks. Actual solutions are thus often tailored directly to the distribution to sample from, such as the Box-Muller transform [BM58], to draw from a standard normal distribution. However, this is not possible for all distributions. The random walks along the navigation grid (cf. section 3.5.2), for example, were already based on samples, and can be used as they are. Other variants, such as some from section 3.3, and the navigation mesh-based movement models from section 3.6, however, can not. On a first glance, predictions such as (3.12), can only be sampled by rejection sampling, which is unsuitable for use on smartphones. However, the prediction model can be reformulated, significantly reducing the sampling complexity

$$
\begin{aligned}
\overline{q}_{t,[i]}^{(\Theta)} &= \alpha & \alpha &= q_{t-1,[i]}^{(\Theta)} &+ \mathcal{N}(0, \sigma_{\text{turn}}^2) \\
\overline{q}_{t,[i]}^{(x)} &= q_{t-1,[i]}^{(x)} + v\cos(\Theta) & & & \\
\overline{q}_{t,[i]}^{(y)} &= q_{t-1,[i]}^{(y)} + v\sin(\Theta) & v &= u_t^{(v)} &+ \mathcal{N}(0, \sigma_{\text{step}}^2) \\
\overline{w}_{t,[i]} &= w_{t-1,[i]} & \langle \boldsymbol{q}\rangle_{t,[]} &= \langle(x,y,\Theta)\rangle_{t,[]}, & \langle \boldsymbol{u}\rangle_t &= \langle(v)\rangle_t.
\end{aligned}
\tag{4.69}
$$

Despite seeming completely different, (4.69) describes the same behavior as (3.12), yet, from a sample-based perspective. For each transition $\boldsymbol{q}_{t-1,[i]} \to \boldsymbol{q}_{t,[i]}$, the new heading is given by the heading of the input sample $\boldsymbol{q}_{t-1,[i]}$, plus an uncertainty, that is drawn from a zero mean normal distribution. The location of $\boldsymbol{q}_{t,[i]}$ is then given by the location from $\boldsymbol{q}_{t-1,[i]}$, adjusted by this heading and the known velocity, which is also modified by a zero mean Gaussian, to incorporate its uncertainty. As can be seen, this prediction denotes a nonlinear movement, including uncertainties for velocity and direction changes, clearly representing a non Gaussian. However, only two Gaussians are required to perform this prediction. The complex problem (3.12) is reduced to updating discrete states, by drawing random samples from two Gaussian distributions. Mentioned earlier, this is well-captured by the Box-Muller transform. As the resulting prediction is based on sample locations, all weights remain as they are.

(4.69) yields a new sample set with more variance than before, as discussed for the Kalman filter. When this prediction is used as proposal, the increased uncertainty is reduced by the sensor observations, similar to the Bayes filter's measurement update (4.26)

$$
\text{bel}(\boldsymbol{q}_t) = p(\boldsymbol{o}_t \mid \overline{\boldsymbol{q}}_t)\,\overline{\text{bel}}(\boldsymbol{q}_t) \quad \text{with} \quad \overline{\text{bel}}(\boldsymbol{q}_t) \to \left\{ \left(\overline{\boldsymbol{q}}_{t,[i]}, \overline{w}_{t,[i]}\right) \right\}_{i=1}^N.
\tag{4.70}
$$

For 1D and multivariate normal distributions, (4.70) was solved analytically in section 4.1 and 4.3. For sample-based density representations, the set of weighted samples denoting $\overline{\text{bel}}(\boldsymbol{q}_t)$ is simply reweighted based on (4.63), also including the previous weight

$$
w_{t,[i]} = p(\boldsymbol{o}_t \mid \overline{\boldsymbol{q}}_{t,[i]})\,\overline{w}_{t,[i]}.
\tag{4.71}
$$

(a) initial   (b) transition   (c) evaluation   (d) transition   (e) evaluation
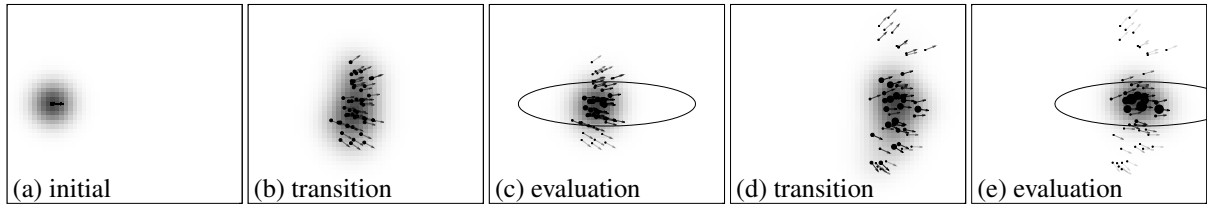
Figure 4.9: Two transition and evaluation steps for example (4.69) with prediction parameters (4.72), and observation (4.73), starting from a well known location using a particle filter with 50 samples. Ellipses denote the observation's 95 % confidence. The density resulting from a KDE is shown as black shading.

That is, every sample $\left(\overline{q}_{t,[i]}, \overline{w}_{t,[i]}\right)$ from the *prediction* is reweighted based on its likelihood $p(o_t \mid \overline{q}_{t,[i]})$ of matching the current sensor observations o_t. If a sample does not resemble the measurements, its weight is decreased, thus affecting the histogram or KDE of all samples. Afterwards, all new weights are normalized, to satisfy (4.59).

To depict (4.69), the initial state is assumed to be well known, with the velocity and heading update as follows

$$q_{0,[i]} = (0, 0, 0°) \, \forall i\,, \quad u_t^{(v)} = 0.7\,, \quad \sigma_{\text{walk}} = 0.1\,, \quad \sigma_{\text{turn}} = \frac{\pi}{8}\,. \tag{4.72}$$

Current whereabouts are observed by a sensor, measuring the location (x, y), with its uncertainty given by a multivariate zero mean Gaussian

$$\Sigma = \begin{pmatrix} 0.16 & 0 \\ 0 & 0.01 \end{pmatrix}\,. \tag{4.73}$$

Results when approximating this setup by 50 samples are depicted in figure 4.9. The initial state is shown in 4.9a, where all samples share the same location and heading. After the first transition, the samples in 4.9b are spread according to the uncertain heading and velocity (4.72). Hereafter, their individual weights $\overline{w}_{t,[i]}$ are adjusted based on the sensor observation. The depicted ellipse denotes the 95 % contour of the sensor's uncertainty. The dot size and arrow intensity in 4.9c depict the resulting weights. As can be seen, the KDE, shown as dark background, concentrates where prediction and observation match. After the next transition in 4.9d, the samples are spread again, hereafter refined by an observation in 4.9e. This setup is referred to as *sequential importance sampling* [DGK01].

The example depicts a previously discussed issue. While 33 of the 50 samples reside within the 95 % contour of the observation 4.9c, only 22 reside within the observation of 4.9e. Over time, most of the samples receive a low observation probability, and the important parts of the posterior are described by only a few samples. This aspect requires adequate countermeasures.

4.5.2 Resampling

The particle filter uses a limited number of samples to approximate the density of the unknown state. Therefore, as many samples as possible should reside within probable areas, to ensure a viable approximation. Figure 4.7a represents the ideal situation, where the unweighted samples are distributed according to a PDF, and most of them reside within likely regions. When using the transition as proposal, it creates such a set of samples. Afterwards, they are weighted based on the observation. When not matching with the observation, a sample's weight is reduced. Sequentially repeating those two steps, an increasing number of samples will obtain infinitesimally small weights, and the approximation of the density suffers, known as *degeneracy problem* or *sample impoverishment* [RC01; Wan+11a; DGK01; Fet+17].

By *resampling*, the weighted representation is converted back to an unweighted one, where samples are placed based on probabilities. For this, unlikely samples are removed, and replaced by likely ones. The purpose is to redistribute samples according to the posterior. Thereby, the actual density must not be altered, within the limits of an approximation. By omitting unlikely samples and adding more important ones, the problem of degeneracy is addressed. The combination of using the transition as proposal, weighting this result based on the observation, and (occasionally) applying resampling, addresses aforementioned issue. Due to the additional resampling step, this setup is referred to as *sequential importance resampling* [RC01; Sär13].

The most common resampling technique is based on previously discussed inverse transform sampling [GSS93]. Here, the set of samples is converted into a CDF, by cumulating weights. Then, a uniformly drawn random variable $\mathcal{X} \sim \mathcal{U}(0,1)$ is used to randomly pick elements

$$\left\{ \left(\boldsymbol{q}_{t,[i]}, W_{t,[i]} \right) \right\}_{i=1}^{N} \quad \text{with} \quad W_{t,[i]} = \sum_{j=1}^{i} w_{t,[j]} \,. \tag{4.74}$$

For discrete samples, drawing random elements can easily be accomplished. Instead of calculating $P^{-1}()$ for some random variable $\mathcal{X}$, to determine the probability that lead to it, the sample that lead to the corresponding $W_{t,[i]}$ is chosen. As the cumulative weights are strictly ordered $W_{t,[i]} \geq W_{t,[i-1]}$, this sample is determined by a binary search that yields the first sample satisfying $W_{t,[i]} \geq \mathcal{X}$ [IB98], resulting in the overall complexity of $O(n \log n)$. All N newly generated samples are assigned the same normalized importance weight $w_{t,[i]} = 1/N$. As samples with high initial weights occupy a larger segment of the cumulative [0, 1] interval, they are drawn more often than samples with lower weights. Unlikely samples thus often are removed, slightly affecting the overall density approximation. This implementation is referred to as *multinomial resampling* [DC05].

With minor adjustments, the complexity can be reduced to $O(n)$. Instead of drawing one uniformly distributed random number per output sample, the $[0, 1]$ range of the CDF can be sampled linearly, as shown in 4.6c and figure 4.7c. Pseudo random numbers are generated by starting at e.g. $x_0 = 0$, and using a fixed increment $x_j = x_{j-1} + {}^1/_N$. This also produces uniformly distributed samples $\in [0, 1]$, but in a linear way. As x_j is strictly increasing, exactly as the cumulative weights $W_{t,[i]}$, the binary search can be omitted. Dependent on its weight interval $[W_{t,[i-1]}, W_{t,[i]}[$, and ${}^1/_N$, a sample is skipped, or added more than once to the output. Due to sampling evenly from the CDF, this is referred to as *systematic resampling* [BDH04].

Bolić et al. [BDH04] provide an overview on resampling techniques, comparing their computational and memory performances, important for smartphone use. They also provide an adjustment named *residual-systematic resampling*, where the cumulative weights (4.74) are not required, further reducing the complexity. Another overview is given by Douc and Cappé [DC05], yet focusing on probabilistic differences, such as the variance of the result.

Aforementioned approaches only use existing samples, and each input sample might be contained more than once within the resulting sample set. While this still increases the number of meaningful samples approximating the posterior, the variance of the state space is slightly reduced. A more correct resampling is given by a continuous approach. First, the discrete samples are converted to a continuous density, using the KDE (4.60). This creates a calculable PDF, which closely approximates the posterior. Hereafter, the presented rejection sampling can be used to randomly draw samples from this PDF. For each dimension of the state, a random sample is drawn within the range of its domain. For the current example, the coordinates (x, y) might e.g. be constrained by the building, and $\Theta \in [0, 2\pi[$. These randomly sampled parameters denote a potential state $\boldsymbol{q}$. It is accepted, when the corresponding KDE $\mathrm{kde}\,(\boldsymbol{q}, \dots)$ yields a probability that is higher or equal than an uniformly drawn random variable $\mathcal{X} \sim \mathcal{U}\,(0, \max_{\boldsymbol{q}} \mathrm{kde}\,(\boldsymbol{q}, \dots))$. This approach produces a completely new set of distinct samples from the posterior, at the expense of significantly increased computational costs, due to both, the KDE and the rejection sampling. While the KDE estimation can be accelerated by approximations, the overhead is still notable. In practice, determining the upper bound $\max_{\boldsymbol{q}} \mathrm{kde}\,(\boldsymbol{q}, \dots)$, is non trivial as well [Ebn+17; Bul+18]. Furthermore, the KDE from (4.60) is unconstrained. That is, regarding localization problems, if several samples are located near physical obstacles, the KDE also assigns a non-zero probability to nearby locations behind the obstacle. This potentially yields resampled samples moving through walls, and other obstacles.

The difference between discrete and continuous resampling is shown in figure 4.10. Simply copying samples based on their probability reduces variation, as some appear more than once. The KDE-based resampling, however, increases the variation. Within the depicted example, a Gaussian with $\sigma_x = \sigma_y = 0.05$, $\sigma_\Theta = 0.03$, and no covariance was used as KDE-kernel.
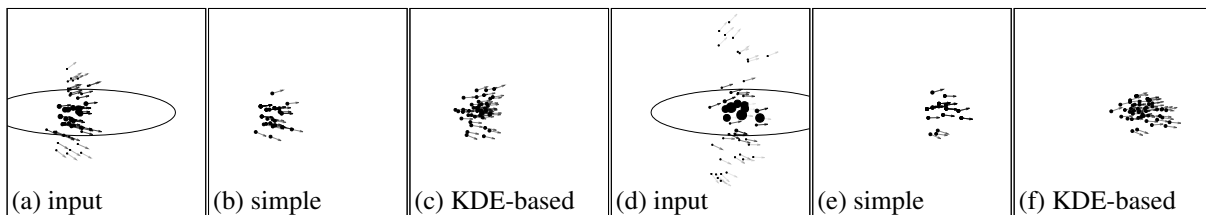
Figure 4.10: Comparison of two resampling strategies, for two weighted posteriors (a) and (d). The simple approach in (b) and (e) uses direct copies from (a) and (d), based on their likelihood. Here, some samples appear more than once. The KDE-based resampling in (c) and (f) draws new samples by rejection sampling. The latter clearly shows more variation than the simple variant.

Independent of the strategy, resampling slightly affects the approximation of the posterior. Therefore, it is often only applied when needed. That is, when an insufficient number of samples has a significant weight assigned to them. This number is referred to as the *effective sample size* [KLW94], depends on the variance among sample weights, and can be approximated [Aru+01]

$$N_{\text{eff}} = \left(\sum_{i=1}^{N} \left(w_{t,[i]} \right)^2 \right)^{-1} \quad \text{with} \quad \sum_{i=1}^{N} w_{t,[i]} \stackrel{!}{=} 1 \,. \tag{4.75}$$

Whenever this number falls below a threshold heuristic, resampling is applied. However, with resampling, one initially mentioned drawback of using the transition as proposal remains, and gets even more pronounced. In case of an erroneous measurement, the approximated density can end up with only a few samples with high weights, rendering the majority of samples unimportant. After resampling, unimportant samples are likely to be removed from the set. The next transition is solely based on the samples that matched the erroneous measurement. Analytical solutions, like the Kalman filter, are able to compensate such single faulty measurements. The particle filter, however, entails the risk of not recovering from the faulty measurement, due to insufficient samples belonging to actually more correct state representations [Fet+17].

4.5.3 Estimation

Yet unanswered is the question of how to infer the unknown state. For the Kalman filter, the mean of the posterior denotes the most likely estimate of the unknown state, usable as-is. For the particle filter, a similar estimation is given by the weighted average of all samples [CGM07]

$$\boldsymbol{q}_t^* = \sum_{i=1}^{N} \boldsymbol{q}_{t,[i]} w_{t,[i]} \quad \text{when} \quad \sum_{i=1}^{N} w_{t,[i]} \stackrel{!}{=} 1 \,. \tag{4.76}$$

Due to aforementioned effects, this estimation should be conducted before applying the resampling step [LC98; CGM07]. When the sample set denotes a Gaussian, or at least an unimodal
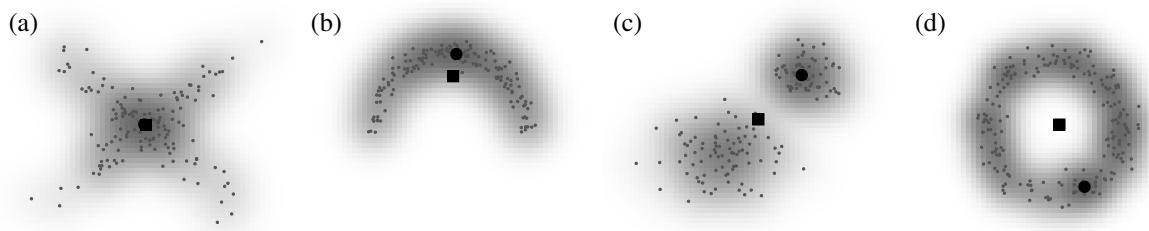
Figure 4.11: Estimation of the unknown state using either the (weighted) average of all samples (square) or the location of the highest probability within the KDE (circle). For densities such as (a), both approaches provide similar results. For densities like (b) and (c), often encountered in indoor localization, the KDE is able to provide more plausible results by focusing on the actual maximum. Especially for (c), where the weighted average is located directly between two modes. In case of many regions with equal probability (d), the KDE can not provide a stable result, as there is no clear maximum value.

distribution, the weighted average yields a viable, single estimation of the unknown state. For multimodal distributions, however, this approach will often yield incorrect results. Assuming a sample set with two modes, similar in probability. Here, the weighted average resides between both modes, where the probability will, most likely, be rather small, as shown in figure 4.11c.

Literature suggests various other estimation techniques, often dependent on the use case and resulting posterior. A broad overview on available discrete and continuous techniques is given by Deng and Wickham, discussing limitations and applicability of several algorithms [DW11].

Due to its simplicity, a common approach is to just use the sample that corresponds to the highest individual weight $w_{t,[i]}$, therefore named *best particle* [Nis+09]. However, this is analytically incorrect for the presented filter setup. When the transition is used as proposal, the weights only denote the likelihood of the observation. The probability of the predicted prior is modeled solely by the *placement* of samples, and thus completely discarded when only considering sample weights [CGM07]. A compromise is given by a spatially limited weighted average, only considering the samples within some vicinity around the one with the highest weight. Whether a sample belongs to this vicinity, is determined by a heuristic. For correctness, the weights of all samples belonging to this region must be normalized to sum up to 1.

Adaptive clustering techniques are similar to this approach, approximating the density using a reduced number of new samples based on their probabilities, referred to as *vector quantization* [UN94]. They are especially useful when regional estimations are preferred over single scalar results. Also, they can be considered within the resampling process, or to strategically decrease the number of samples, to reduce computational requirements without affecting the quality.

Besides discrete approaches, other techniques derive a continuous representation of the sample set. Regression-based methods can e.g. be used to infer a nonlinear function that approximates the weighted samples. Type and degree of the function strongly depend on the underlying

problem, as well as resulting samples [Loa99]. The derived function then serves as basis for further evaluations. Similar to the KDE, this derives a continuous representation of the density, approximated by the sample set. However, neither of both directly indicates the most likely state, as given by the Kalman filter's mean. Such a single value is given by finding the maximum within the continuous representation

$$\boldsymbol{q}_t^* = \arg\max_{\boldsymbol{q}} \text{kde}\left(\boldsymbol{q}, \left\{\left(\boldsymbol{q}_{t,[i]}, w_{t,[i]}\right)\right\}_{i=1}^N\right) , \quad \boldsymbol{q}_t^* = \arg\max_{\boldsymbol{q}} f_{\text{reg}}\left(\boldsymbol{q}\right) . \tag{4.77}$$

When the fitted $f_{\text{reg}}\left(\boldsymbol{q}\right)$ allows for calculating extrema, $\boldsymbol{q}_t^*$ can be determined analytically. For the KDE-based variant, however, optimization algorithms are required for estimating $\boldsymbol{q}_t^*$. These algorithms suffer from the risk of not converging, or getting stuck within a local maximum (see section 2.7). Furthermore, the complexity of KDE calculations depends on the number of samples, and the chosen kernel. Discrete approximations can address this problem, simultaneously providing the global maximum that was encountered during the discrete calculations [Bul+18]. Figure 4.11 depicts a comparison between the weighted average, and KDE-maximum estimation techniques. In case of multimodalities, or oddly shaped densities, the KDE can circumvent issues encountered with the weighted average (cf. 4.11b and figure 4.11c). For densities like figure 4.11d, where many regions share a similar probability, the KDE can provide unstable results. Additionally, small changes within the underlying sample set can shift the global maximum to some completely different state, yielding noisy estimations over time. This effect is not confined solely the KDE, but affects others as well.

All aforementioned estimations derive the most likely state solely based on the *current* posterior sample set. For many real-world use cases, like localization and navigation problems, there is a strong temporal dependency between the sets. It thus makes sense to relate the current, and previous estimations. This aspect is addressed by *smoothing* techniques, considering not only the current but also previous sample sets, similar to a trajectory [GDW01].

By including the history of sample sets, temporal ambiguities can be addressed. A density that is split in half by an obstacle yields potential issues within the estimation. If one of the two modes gets unlikely over time, e.g. due to architectural constraints or sensor observations, smoothing can be used to refine *previous* estimations, preventing splitting from the start. While this is mainly suitable for offline use, where pre-recorded datasets are analyzed and refined, online use is also possible. By slightly delaying the result presented to the user, future observations and predictions become available, and can be considered when deriving the delayed result. This is referred to as *fixed lag smoothing* [PW09; Fet+16].

As shown, in contrast to the Kalman filter, the particle filter is more versatile, but comes at the cost of being more discrete, and having increased computational requirements.

4.6   Summary

Within this chapter, the theoretical background of sensor fusion was discussed. It is used to combine the information from sensors (chapter 2) with the pedestrian walking models, and the floorplan (chapter 3). The presented fusion uses the Bayes filter. It describes the most likely state of a problem, based on the history of all sensor observations, and predictions. There possible implementations for the Bayes filter were introduced: The Kalman filter, the extended Kalman filter and the particle filter. During discussions, the benefits and drawbacks of each implementation were pointed out.

After briefly describing the limitations of simple filtering, such as the moving average, the concept of the probabilistic Bayes filter was introduced. It derives the likelihood of an unknown state, based on sensor observations and predictions. That is, instead of using scalar values only, it considers uncertainties, to derive both, the globally best result, and its quality. It also allows combining multiple sensors, all contributing to the same result. A prediction step is used to describe the behavior of the unknown state, and to include external influences.

The Kalman filter was introduced as an analytical implementation of the Bayes filter. It uses matrices and vectors, to describe the unknown state, and its behavior, from the viewpoint of multivariate normal distributions. While being analytical and optimal, it is limited in terms of supported problems, which are required to be linear and Gaussian. The extended Kalman filter mitigates some of these limitations, supporting nonlinear problems, but is still bound to multivariate normal distributions. Thus, both filters are unsuitable for floorplan-based indoor localization and navigation, as discrete influences, such as walls, can not be considered.

To handle the complex probability densities from chapter 2 and 3, the particle filter was introduced. It also implements the Bayes filter, but uses discrete samples to approximate all required probability densities. While this is more versatile, it introduces discreteness, and increases computational overhead, dependent on the number of samples used. Furthermore, the process requires drawing random samples from the densities described in 3. Therefore, the concept of random sampling was introduced theoretically, including brief modifications, to reduce computational complexity. Hereafter, sensor observations are included by weighting all resulting samples. To ensure that they remain representative, the concept of resampling was discussed. Finally, the most likely state is inferred by one of several estimation techniques.

Focusing solely on the theoretical backgrounds, several problem-dependent aspects are still unresolved. The following chapter will thus focus on the practical aspects concerning sensor fusion in smartphone-based pedestrian indoor localization and navigation.

Chapter 5

Indoor Navigation

Previous sections examined the individual components, which are required for a smartphone-based indoor localization and navigation system, as shown in figure 1.3. Observations from sensors were used to evaluate potential movements or whereabouts, and models predicted pedestrian walking behavior, utilizing the building's floorplan to restrict movements by known architecture. Hereafter, both parts were combined on a probabilistic basis, using recursive density estimation to derive potential whereabouts after a certain amount of time, based on all sensor observations, movement predictions and the building's floorplan. However, many minor, yet important, aspects are still vague or unclear. Such as the way observations and predictions are fused within the recursive density estimation. While the general approach was discussed in detail, it e.g. remains unclear whether sensors should be considered within the prediction or evaluation step. Furthermore, while the necessity for an update interval was briefly discussed, its actual timing, and whether it depends on the pedestrian currently walking or resting, is not yet defined. Similarly, the semantic floorplan, used to derive the navigation grid or mesh from chapter 3, was assumed to be a given. Actual strategies for deriving it, e.g. based on existing floorplans, with support for later modifications, are yet to be discussed. Depending on the architectural complexity of the building, and the resulting floorplan, certain issues might arise, affecting the quality of the localization, eventually causing errors, yielding the recursive density estimation to get stuck. Potential issues, and how to address them, also require further examination.

This section provides missing details, briefly describing requirements for editing and deriving indoor floorplans. How to actually fuse sensors on a smartphone, providing visual feedback in realtime, without requiring large amounts of computational power and memory. Preventing potential issues, and how to set up the system from scratch, with later experiments in mind.
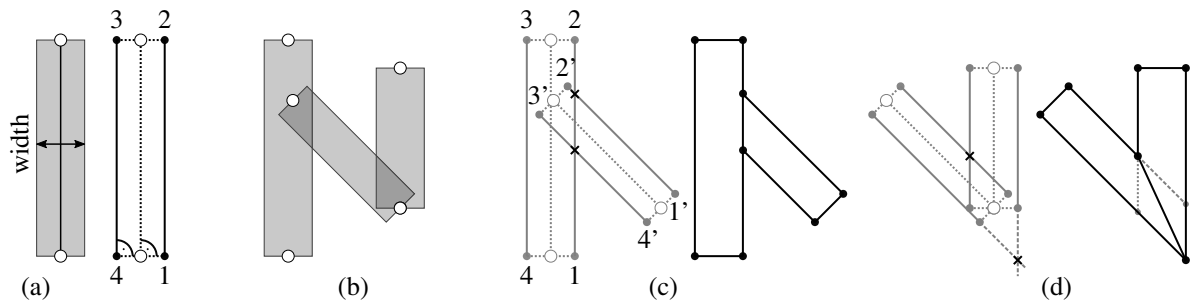
Figure 5.1: Representing walls using either a line between two points and a certain width, or a quad of four points (a), and potential intersections between them (b). If one wall ends within another, two of its four points are replaced by intersections (c). Two connected walls (same from/to) are either joined by calculating an inner and outer intersection (d), or by cutting one and enlarging the other (dashed/dotted).

5.1   Complex Indoor Maps

Besides providing a visualization to the pedestrian, previous discussions on sensor and movement models clearly indicated that a floorplan with semantic information yields substantial improvements for the localization process. The following thus gives a brief overview on the way floorplans are modeled, to be usable within the overall localization and navigation system. Especially, not requiring large amounts of data, allowing for adding semantic information, and maintenance [ARC12]. Omitting minor exceptions [Fet+18], most buildings are composed of stacked floors with a constant height, connected by stairs, elevators or escalators, and their walls starting at the bottom, up to the ceiling, using the complete height. Floors and walls can thus be modeled in the 2D plane, e.g. by using 2D polygons, or other suitable primitives. For a 3D representation, walls are extruded, e.g. based on the current floor's height, reaching the ceiling.

For the presented system, all walls are modeled as 2D line-segments, defined by two points, and a certain thickness, denoting a quadrilateral. The latter is extrudable up to the floor's height, to form a 3D obstacle. When using this representation, adjacent or intersecting walls do not share a smooth interconnection. As depicted in figure 5.1, when the angle between both is not a multiple of 90°, they overlap in one of the shown ways. While issues could be mitigated by placing walls only at multiples of 90°, many buildings will not satisfy this requirement [Fet+18]. Incorrect intersections are only a minor problem for calculating the navigation grid/mesh, but they impose issues when using the resulting model for 3D ray tracing, or within a 3D visualization presented to the user. Depending on the type of overlapping, *z-Fighting* can e.g. cause visual artifacts [VF13]. Also, 3D ray tracing approaches require clear entry and exit points for the rays when traversing obstacles, such as walls. This requirement is not satisfied when one wall ends within another. When wall thickness is considered, as depicted in figure 5.1, 2D ray tracing is affected as well.
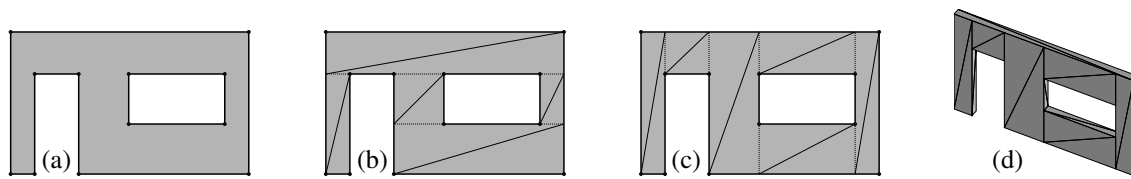
Figure 5.2: Typical wall with cutouts for doors and windows (a). An optimal triangulation of the wall's front face requires 10 triangles (b). The discussed simple triangulation algorithm requires 12 triangles (c). The final 3D result, including lateral surface and window frame, requires 48 triangles (d).

There are multiple options to address intersecting 3D elements. One of the most common are Boolean operations on 3D meshes. However, depending on input geometry and Boolean operation, related approaches often produce suboptimal results due to numerical issues [JHS98; Wan11]. As all walls will be modeled in 2D, their intersection problem can also be solved 2D, using quadrilateral intersections, followed by extruding the modified quad by a predefined height, to create the 3D obstacle. The two most common wall intersections within floorplans are walls ending within another, and walls directly connected to each other, shown in figure 5.1b.

To detect walls ending within other walls, each of the four corner points from figure 5.1a is tested, whether it is contained within the quad of another wall. This *point-in-polygon* problem can e.g. be solved using the *even-odd rule* or by calculating the *winding number* [HA01]. If a point is identified to end within another wall, its line segment is intersected with the ones from the other wall, using simple line-line intersections.

Similarly, walls are connected to each other if they share one of their defining points. To smoothly connect them by creating a corner, two intersection points are identified, and segments are adjusted accordingly (cf. figure 5.1d). While this does not create realistic wall endings, as both are joined via a shared edge, it works reasonably well for both, 3D visualization and ray tracing. For a realistic joining, one of the two walls must be marked explicitly as the longer outer one, and the other as the shorter inner one, depicted as dashed and dotted lines in figure 5.1d. Here, it is not directly clear which of both variants is correct.

Additionally, cutouts for doors, and optionally windows, are required. Again, Boolean mesh operations could create them by subtraction. However, assuming that all walls, doors and windows are defined as extruded 2D quads, and doors/windows can not be placed above but only beneath each other, the problem is reduced to a simple triangulation, shown in figure 5.2. Dividing the wall into vertical slices, a window cutout is created by placing a 3D quad (two triangles) above and below it. Likewise, doors are created by placing a 3D quad above. The remaining empty parts right and left of the doors and windows are also filled by quads. This triangulation scheme is depicted in figure 5.2c. While the result uses more than the optimal number of triangles, shown in figure 5.2b calculated by 2D polygon Boolean operation [JHS98; MK89],
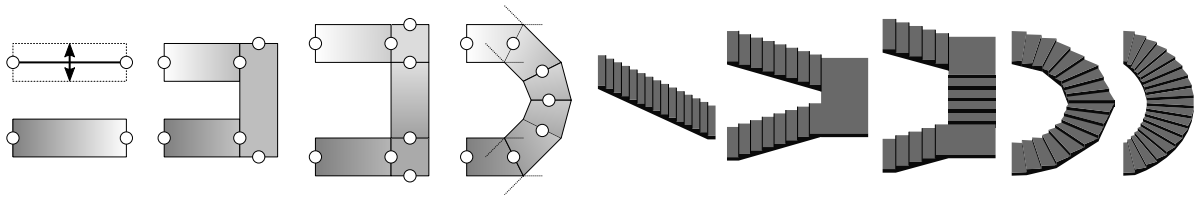
Figure 5.3: Stairs are defined by multiple adjacent quads (left), each given by two 3D points and a width. When configuring adjacent quads to be joined by intersection (similar to figure 5.1), this method also allows for approximating circular stairs. Quads can easily be converted into a 3D representation using triangulation (right). The approximated circular stair differs only slightly from the correct representation.

the approach does not require costly intersection operations [HNU99], and is numerically stable. After triangulating the front and back of the wall, its lateral surface and window frame are created by zig-zag connecting the front and the back face vertices, e.g. using a *triangle strip* [KSS17]. Figure 5.2d shows the final 3D result for a wall with one door and one window.

Due to numerous architectural options, stairs require increased modeling effort. Yet, most of them can be divided into several adjacent 3D quadrilaterals, defining both, the stair's shape and slope. To allow for 2D modeling, each quad is defined similarly to walls, using two points, and a thickness. The required 3D component is included by assigning both points an altitude above ground. A stair directly connecting two floors is thus expressed as a single quad, starting at the current floor, and ending at some altitude above it. For typical stairs, three quadrilaterals are used. One for the first slope, up to half the floor's height, one for the plateau in between, and one for the second slope, up to the next floor. This requires quads to be directly adjacent, gapless, sharing one edge, and can be addressed similarly to walls, by intersecting adjacent quads, adjusting their edge-points for a smooth intersection. All cases are shown in the left of figure 5.3. As before, the quad-based definition is used to derive a 3D representation, by linearly interpolating along each quad's edges, dividing it into several treads, based on average tread sizes. The interpolated points are then duplicated, and adjusted in altitude to denote the edge-points of each tread's upper and lower side. Pairs of eight vertices are then connected by triangles, similarly to a cube. Resulting 3D representations are shown in the right of figure 5.3.

Similarly to stairs and walls, floors are modeled as 2D polygons, containing cutouts for stair-wells, escalators and elevators. Each polygon is equipped with semantic information, whether it belongs to an indoor or outdoor area, e.g. to disable or enable the GPS component. Including cutouts directly when drawing a new polygon is often cumbersome. Therefore, the floor can be modeled without cutouts first, hereafter defining additional cutout-polygons, and subtracting them from the base using Boolean operations [Vat92]. To determine the polygon's 3D representation for visualizing or ray tracing, it is converted to triangles, using algorithms supporting polygons with holes [Hel98].
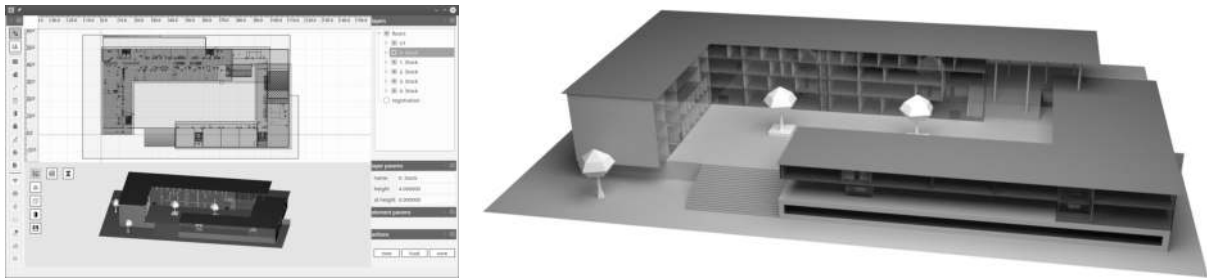
Figure 5.4: Screenshot of the developed Map-Editor, and a realistic rendering of the edited 3D map.

Maps discussed and presented within this work were created using a developed map-editor (see figure 5.4), allowing for fast editing, and including semantic information. Its 2D definitions are used to derive a 2D visualization for the pedestrian, and to generate the navigation grid from section 3.4. Similarly, the derived 3D representation serves as 3D visualization, and is required to build the navigation mesh from section 3.6. Even though it describes architecture in 2D and per-floor, instead of using real 3D modeling, all encountered buildings were editable, representing a viable tool for the indoor localization and navigation system.

5.2 Fusing All Components

The two main approaches for fusing available sensors with potential pedestrian movements, discussed in chapter 4, were the Kalman filter, and the particle filter. Both combined a movement prediction with an evaluation based on current sensor observations, that is, a recursive, periodic update routine, visualized in figure 5.5. While previous discussions focused mainly on the mathematical background, they did not consider all aspects required for a smartphone-based pedestrian indoor localization and navigation, examined within the following.

As pointed out, the Kalman filter is an analytical implementation of the Bayes filter. While results can be calculated quickly without requiring large amounts of memory, it is limited to linear Gaussian problems. This is mitigated by the extended Kalman filter with support for non-linearity, yet, still limited to Gaussian formulations, neither supporting multimodalities, often encountered with lateration (see section 2.7), nor offering capabilities for including floorplans. The Kalman filter's use concerning smartphone-based indoor localization and navigation is thus rather limited. While there are constellations where analytical filters are sufficient, or can be used additionally, the particle filter is a better suited implementation of the Bayes filter for an indoor localization system. Its sample-based density representation allows for multimodalities, encountered within observations from Wi-Fi, walks splitting into multiple directions, and non-linear problems, like the heading-based movement required for most prediction models from
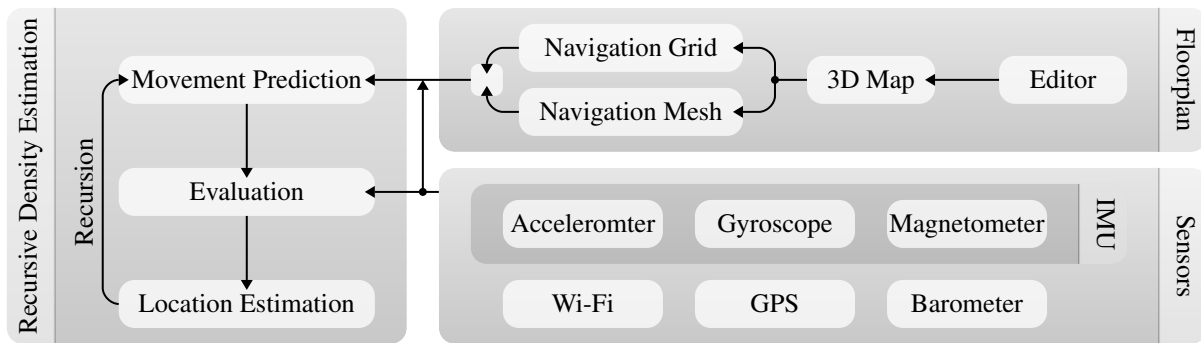
Figure 5.5: Overview of the overall system. Used as visual reference for the following discussions.

chapter 3. By using individual samples, it is also possible to evaluate intersections with the floorplan, for each potential movement. Furthermore, the sample-based representation directly resembles the concept of simulation, introduced and required for the random walks in section 3.5. Likewise, this filter matches both concepts for representing a sample-based density, discussed for predictions on the navigation mesh in section 3.6: Either by the *distribution* of samples with identical importance, or by assuming a uniform distribution, and applying *weights* to them. As the particle filter is well suited for solving the overall problem formulation, this section focuses solely on this type of filter, mentioning the Kalman filter only for reference, or when explicitly applicable. Yet, the particle filter comes at the cost of significantly increased computational requirements, already discussed for the simulations of statistical processes, as the approximation quality strongly depends on the number of samples, that is, used particles. Repercussions and potential workarounds will also be addressed by the following remarks.

For a first impression on the system's behavior, figure 5.6 depicts a 2D example, using step-detection, turn-detection, and an absolute evaluation, similar to the one provided by Wi-Fi or GPS. The unknown state is defined as $\langle \boldsymbol{q} \rangle_t = \langle (x, y, \Theta) \rangle_t$. At $t = 0$, location and heading are *unknown*. As discussed, this is modeled by uniformly placing particles (dots) throughout the walkable area, each with a random heading (thin lines). When this initial density is updated, all particles walk $\approx 0.7\,\mathrm{m}\ (1.4\,\mathrm{m/s})$ into the direction of their currently assigned heading, causing some of them to encounter impassable walls, indicated by gray dots for $t = 1$. With every subsequent update, the heading $q_t^{(\Theta)}$ is adjusted based on turn-detection, before moving another $\approx 0.7\,\mathrm{m}$. The absolute evaluation at $t = 3$ favors all particles within the circle, denoting the 1σ contour. Most particles outside of this range are unlikely hereafter, indicated in gray. After several updates, only a fraction of all initial particles remained likely, and the system converges with the pedestrian's actual walking path, indicated as thick line. Within the following, all omitted details, on when to perform a filter update, which observations to include, how to handle particles encountering an obstacle, which resampling and estimation to use, will be discussed.
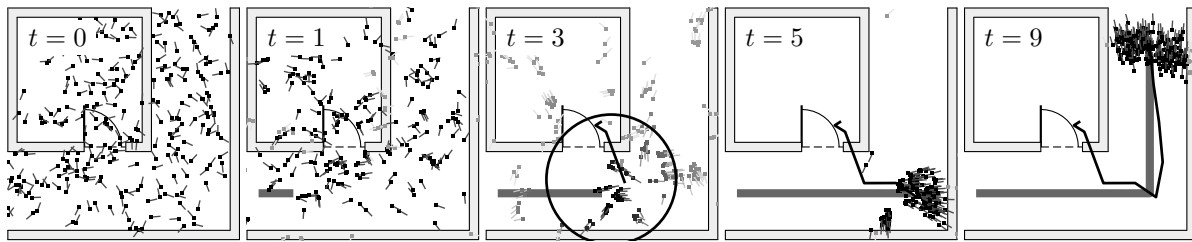
Figure 5.6: Example behavior, ignoring implementation details. At $t = 0$ all particles (dots) are uniformly distributed. After one filter update ($t = 1$), many of them described impossible walks (gray dots). The evaluation at $t = 3$ favors all particles within a small area (circle). Hereafter, the density converges with the actual path (thick gray line). Estimations are based on the weighted average (black line).

5.2.1 Update Frequency

One major aspect of Bayes filters, also concerning system performance, is the time period between consecutive updates $t-1 \rightarrow t$. That is, how often a prediction is performed and evaluated against current sensor observations. It is indicated by the loop within the left of figure 5.5. This time period influences how often new location estimations are available, and thus how often they can be presented to the pedestrian holding the smartphone. In terms of user experience, faster updates are preferred, as the pedestrian receives an immediate feedback on current whereabouts. As discussed, the same holds true for most movement prediction models, being more suited for shorter walks, that is, faster update rates (see chapter 3). Especially the navigation mesh, not using shortest path calculations but simple reachability tests, is more accurate for smaller walking distances. On the other hand, faster updates require more computations, affecting battery life. Many indoor localization systems refer to a constant update rate around $500\,\mathrm{ms}$ [Jim+12; Hel+13; Ndz+17; Ebn+17]. This value is potentially inspired by aforementioned pedestrian walking speeds, of approximately two steps per second. However, when assuming a constant update rate, predictions and evaluations are also performed when the pedestrian is currently resting, draining the smartphone's battery unnecessarily. To provide instant visual feedback, still conserving energy, it therefore makes sense to couple updates with the step detection process, only updating when a step was detected [Fet+18]. This strategy represents the basis for all following discussions, and is shown in figure 5.7, giving an overview on time and data behavior, that is, individual sensor contributions, intended as visual reference.

When using a step-detection-based update strategy, the current location of every particle is first used to determine whether it belongs to a leveled ground floor, or a sloped part of a stair. This information is then used to adjust the step length μ_{step}, that is, the walking distance for the prediction, discussed in chapter 3. When the pedestrian is currently resting, the necessity for updates depends on the visualized information. If solely the current whereabouts are presented,
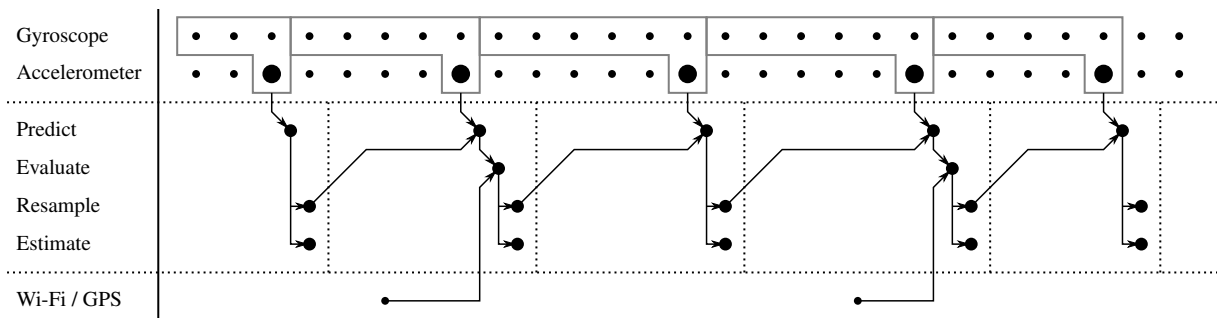
Figure 5.7: Overview on the system's time and data behavior. A prediction occurs when a step is indicated by the accelerometer (big dot), and includes the cumulated change in heading from the gyroscope. When Wi-Fi or GPS observations are available, the prediction is evaluated before conducting the estimation and resampling. The barometer and magnetometer, not shown within this setup, are applicable to both, prediction and evaluation. Theoretical backgrounds are discussed within the text.

no visual updates are required, as they remain constant. When the currently estimated heading is visualized, updates might be required, e.g. by triggering them not only after every detected step, but also after some timeout, when no steps were detected. In case of no steps, the walking length for the prediction model can then be set to $\mu_{\text{step}} = 0$, with an uncertainty $\sigma_{\text{step}} > 0$, allowing for slight deviations [Ebn+15].

The step-based update also covers probabilistic step-detection variants, like (2.29). Here, a filter update is performed for every peak detected within the accelerometer's magnitude. Depending on its likelihood to denote a step, a fraction of the particles uses the estimated step length. The remaining part uses a walking distance of zero with a small uncertainty. By configuring the number of particles for each of the two fractions based on the step-probability, the resulting prediction directly follows previous discussions on (2.29) and (2.31).

Shown in figure 5.5, independent of the chosen movement model, and the spatial model used for representing the floorplan, the first action of the filter update performs a movement prediction, by adjusting every single particle, e.g. based on a step length and heading. Hereafter, the evaluation fuses this intermediate result with current sensor observations to derive the posterior. For this second step, it is not directly clear which values may be considered, and which must not, as (4.21) includes only the most recent observations. Due to the sensors not sharing a common update rate, there might be recent observations for sensor A, but none for sensor B. When performing filter updates based on detected steps, that is, approximately every $500\,\text{ms}$, all IMU Sensors and the barometer can be expected to have provided new observations since the last filter update. Their readings can always be included within the evaluation to refine the prediction. GPS and Wi-Fi, however, might not have provided new observations, as their update rates are much slower, or they even might have failed completely. It is thus unclear how to proceed during the evaluation, when no recent sensor observations are available.

When referring to the GPS, the last received reading could be re-used, when its error indication is adjusted depending on its age. That is, σ_{gps} increasing with the age of the sensor observation. However, when using a Gaussian within (2.17), this yields an evaluation where the last sensor indication still denotes the most likely whereabouts, but nearby locations now being more similar in terms of likelihood, due to the increased uncertainty. When the regional distribution (2.22) is used instead, the behavior is different, and more as intended, as the region of potential whereabouts is increased, without favoring a single location. The correctness of re-using old observations by increasing their uncertainty thus strongly depends on the chosen probabilistic comparison. Similar rules hold true for Wi-Fi evaluations. However, as there are multiple observations, one per access point, multiple uncertainties must be adjusted, depending on the age of each measurement. After a certain amount of time, uncertainties become too large for observations to be valuable. In such cases, simply omitting missing observations is a better choice [TBF05]. The same holds true for normal distribution-based comparisons of old observations, as they can affect the result negatively. Thus, Wi-Fi and GPS are only included when recent sensor readings are available. All other sensors provide a decent refresh rate, for observations to be available at every single filter update. This allows for including the current compass heading, the latest absolute or change in atmospheric pressure, or the currently detected activity within every single evaluation.

5.2.2   Including Observations

Mentioned within chapter 2 and figure 5.7, sensor observations can be applied to either the prediction or the evaluation step. An example, denoting the difference between both, is shown in figure 5.8. Here, a particle filter is combined with two movement prediction models, step-detection and turn-detection, with the initial location and heading well known. The prediction in the top and center row uses a walking distance of $\mu_{\mathrm{step}} = 70\,\mathrm{cm}$ with $\sigma_{\mathrm{step}} = 10\,\mathrm{cm}$, but does not constrain the heading, proceeding into an unknown direction. This is similar to figure 3.6 and (3.11), but omitting the case of the pedestrian currently resting. The evaluation then includes heading and turn-detection from the gyroscope, with $\sigma_{\mathrm{turn}} = 3°$. Hereafter, only the particles moving to the right are likely, matching with the estimated pedestrian movement. During the resampling step (center row), all unlikely particles are replaced by more likely ones. Due to evaluation and resampling, the predicted density is refined from a circular shape into a single spot, denoting the starting point for the next iteration. Compared to figure 3.6, this prevents consecutive predictions from creating an unusual mixture of circles. If the resampling step is omitted (top row), the system fails to converge, as no particle remained likely after three update
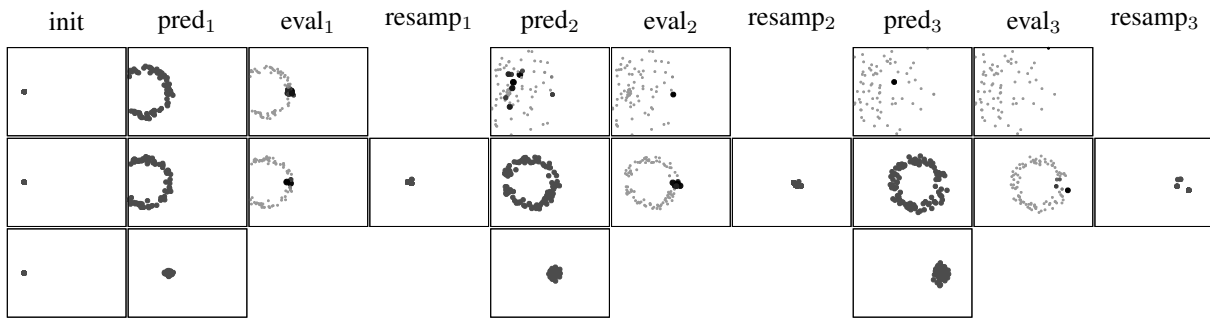
Figure 5.8: Three consecutive filter executions using a setup with: a poor prediction and evaluation (top row), a poor prediction, evaluation, and resampling (center row), a good prediction (bottom row). Each density is approximated by 100 particles, where more important ones are darker and slightly bigger. When raising the number of particles, all three variants become increasingly similar.

steps, clarifying the importance of this step. Compared to top and center row, the results shown in the bottom of figure 5.8 are significantly different, examined within the following.

Previous discussions of the Bayes filter were solely focused on the mentioned procedure: Using a model to perform the prediction, hereafter including all sensor observations to calculate the evaluation. Shown for the particle filter in figure 5.8, the prediction relocates the particles, and the evaluation assigns some weight to them, with both steps together forming the resulting posterior. As discussed, this two-step process is needed, as drawing directly from the posterior is often impossible [RC01]. Ideally, the prediction and evaluation agree. That is, the prediction places particles into regions, where the evaluation is likely as well. For an unconstrained prediction of pedestrian movements, this requirement often is unfulfilled, clearly visualized in the first two rows of figure 5.8. If the movement prediction is performed without any knowledge on potential heading changes, all walking directions must be assumed possible, as the pedestrian could have turned by 180° between two consecutive updates. Thus, particles are moved into a random direction, with solely the distance being constrained. This prediction widely scatters the previous set of particles. The evaluation then weights all resulting particles based on whether their movement matches the gyroscope observations, which are strictly constrained by σ_{turn}. Typically, this results in a large fraction of particles with small weights, as most predicted movements did not match the evaluation. When proceeding from this result, only a few particles remain meaningful, degrading even further over time, depicted in the top row of figure 5.8. This is addressed by resampling, replacing unimportant samples with important ones. However, when using the typical duplication-based resampling (cf. section 4.5.2), this still causes the density approximation to degrade, as only a few samples remain distinct, visualized in the center row. This problem can either be mitigated by utilizing advanced resampling techniques, like the KDE (see figure 4.10), or, preferably, by enhancing the prediction (bottom row).

As also stated by Thrun et al., it therefore makes sense to include sensor observations already within the prediction, as if they were known external influences $\boldsymbol{u}$, also referred to as control data [TBF05]. The validity of using observations during the prediction step is confirmed by the derivations from section 4.2, based on the Markov property assumption [KGD14]. The impact is shown in the bottom row of figure 5.8. When the prediction directly includes the observation on heading changes $o_t^{(\theta)}$ since the last filter update, particles can be moved into a single direction with an uncertainty σ_{turn}. To not add this information twice, it must be removed from the evaluation. For the currently presented example, the evaluation can thus be completely omitted, as every sensor contributes to the prediction. The posterior is thus modeled solely by particle placement, all sharing an equal weight. Both shown variants, using observations within the prediction and skipping the evaluation, and, using an unconstrained prediction adding observations during the evaluation, produce the same density in theory – when assuming an infinite number of particles. Due to their limited number of particles, real-world setups will thus deviate significantly. As can be seen in the bottom row of figure 5.8, including the observation within the prediction provides the best result, as all particles are distinct. The prediction without observations, followed by evaluation and resampling (center row), also covered the likely areas. However, the duplication within the resampling yields a less distinct result. Including the observation directly within the prediction thus represents an effective way of preventing *sample impoverishment* [Fet+17].

For the final indoor localization system, step-detection and turn-detection are used solely within the prediction (cf. figure 5.7). Doing so not only prevents sample impoverishment, it also helps the floorplan-based movement prediction to provide better results, based on the additional constraints. Corresponding equations in chapter 2 were provided from the evaluation's point of view, with the prediction's viewpoint derived in chapter 3. While eCompass, barometer and activity could be added to the prediction as well, further increasing its quality, this is only suited for random walks along the navigation grid (cf. section 3.5.2). For the navigation mesh, however, drawing from the resulting mixture distribution is limited by computational complexity. To ensure comparability between both spatial models and remain suitable for embedded use, these three sensors are included as evaluation. While this might seem like a drawback, a prediction based on step and turn-detection already is significantly constrained, and close to the posterior. Similarly, Wi-Fi and GPS are only used for evaluation, as they denote rather broad multimodal densities, also hard to draw random samples from. The resulting setup thus represents a combination of the center and bottom row from figure 5.8, trying to provide a prediction that is close to the posterior, using only some sensors as evaluation, followed by resampling. Due to the quality of the prediction, the computationally inexpensive simple resampling (see section 4.5.2) can be used, and only when necessary, well suited for use on smartphones.

Figure 5.9: When particles (dots) are stopped in front of encountered obstacles, without taking further measures, they eventually get stuck. This not only reduces the quality of the density approximation, but also negatively affects the weighted average estimation (line), clearly lagging behind.

Furthermore, as this results in a tendentially more concentrated posterior with many distinct particles, the number of particles required to provide a stable approximation is significantly reduced. This matches with earlier discussions on the number of samples required for approximating statistical processes in general, conducted in section 3.2. When including the observation within the prediction, and focusing on shorter update intervals, the number of particles can be reduced, and it is sufficient to predict only a single $\overline{q}_{t,[i]}$ for every $q_{t-1,[i]}$, instead of multiples. The number of required particles thus remains constant throughout subsequent filter updates, instead of constantly rising, again increasing suitability for smartphone use.

5.2.3   Handling Impossible Movements

When using an observation-based prediction and the building's floorplan, some peculiarities must be considered. The floorplan can yield negative side effects, independent of whether intersection tests, random walks, or the navigation mesh are used (cf. chapter 3). An example is shown in figure 5.9, where the prediction proceeds to the right, with a large heading uncertainty of $\sigma_{\text{turn}} = 7°$ for visualization. Here, all movements intersecting with a wall are stopped directly in front of it, similar to the random walks, discussed in section 3.4, stopping in front of obstacles, due to the unavailability of walkable edges with high probabilities (cf. figure 3.16). Without further measures, this yields and increasing number of particles to get stuck and ineffective over time, and also affects the estimation negatively. Figure 5.9 uses the common estimation, based on the weighted average of all particles (see section 4.5.3), shown as a black line. Including caught particles causes incorrect results, shifting the estimated whereabouts. The weighted average clearly starts to stay behind the meaningful particles, as soon as the door is reached. This issue is mitigated when using a particle filter and absolute location estimations (e.g. Wi-Fi), which eventually re-weight unlikely particles. Yet, the problem can be prevented beforehand.

One discussed solution was to repeat impossible movements, randomly drawing new values for distance and heading, eventually encountering a possible solution. However, for most cases, the probability of finding a valid solution within a few retries is rather low, especially when directly facing an obstacle. Another approach from literature simply deletes each particle that belongs to an invalid movement [EBS16]. Yet, this causes their number to deteriorate over time,
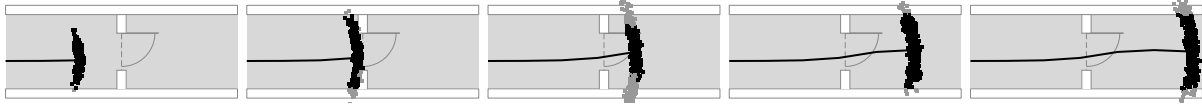
Figure 5.10: Particles (dots) encountering an obstacle are assigned a weight of zero (gray, cf. (3.14)), and thus replaced during resampling. This suppresses impossible walks, and still ensures that the majority of particles remains effective. Compared to (5.9), the estimation (line) is now as desired.

e.g. requiring a prediction that increases the number of particles. Alternatively, the transition itself should directly mark each sample that causes an impossible movement as unlikely, by assigning it a weight of (near) zero (3.14) [Ebn+14]. Within the evaluation, such particles are (almost) nonexistent, and the overall number remains constant. During the resampling step, low-weight particles are replaced by duplicating more likely ones, restoring the number of effective ones. While the duplication reduces distinctiveness, it is viable, as it typically affects only a small fraction of all particles, and is computationally efficient. The result for this strategy is shown in figure 5.10, where zero-weight particles are drawn in gray, directly before they are removed by the particle filter's resampling. The estimation lag present in figure 5.9 (black line) is corrected, all particles define a homogeneous density, and the estimation is located correctly. This strategy was also applied for creating the initial example, shown in figure 5.6.

While down-voting particles that denoted impossible movements was able to address the previous problem, there are constellations where it is unable to, or causes new problems: When receiving erroneous sensor observations, like invalid heading changes due to the pedestrian shaking the smartphone, or invalid Wi-Fi location estimations due to poor signal strength predictions, particles might e.g. be forced into some corner of the floorplan. Depending on how the pedestrian continues to walk, that is, upcoming observations, these particles are either able to leave the corner, or get caught within. An example is shown in figure 5.11, again, solely based on step-detection and turn-detection. Between the first and second prediction, the pedestrian shakes the phone, causing a sensor fault. This yields an incorrect relative heading observation, changing it by approximately $30°$ to the left (counter clockwise). Instead of moving along the corridor, almost all particles are forced into the adjacent room. The following observations are accurate again, representing the pedestrian's walk along the corridor. That is, three steps without change in heading, followed by a $90°$ turn to the left, hereafter walking straight again. As can be seen, this causes all particles to get trapped within the room, unable to leave.

Again, there are various options for addressing this type of error. When using analytical densities, the problem is mitigated, as the room's exterior still remains likely, even if infinitesimally small. However, most analytical approaches are unable to accurately model discontinuous problems, as discussed for the Kalman filter. This type of issue thus represents a compromise that comes with the advantages of the sample-based particle filter. One solution is to (significantly)
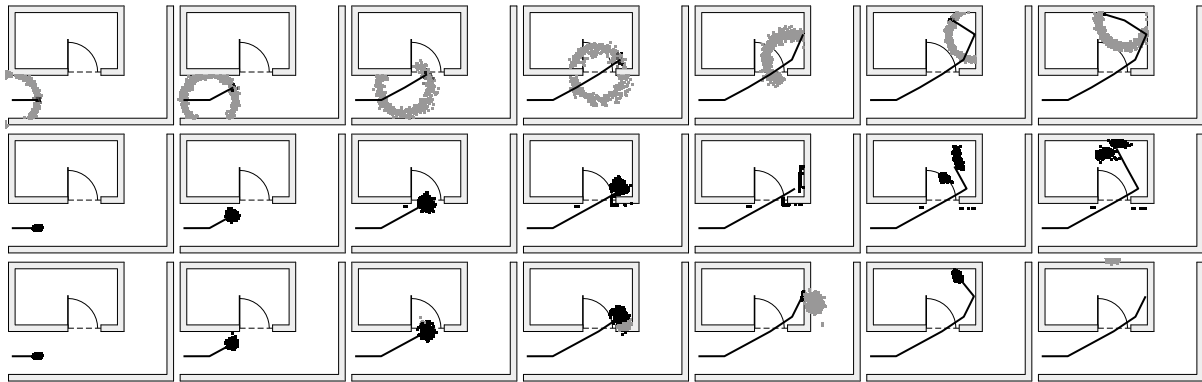
Figure 5.11: Example with a sensor fault between the first and the second prediction. Instead of moving along the corridor, most particles (dots) are forced inside a room, unable to leave. This behavior occurs for all three variants: predicting movements into all directions followed by evaluation and resampling (top row), observation-based prediction and stopping in front of obstacles (center row), and observation-based prediction, down-weighting invalid movements (gray) and removing them during resampling (bottom row). For the latter, the last step contains only particles with a weight of zero, thus having failed.

increase the number of particles, enhancing the quality of the approximation, hopefully causing some particles to remain outside of the room, eventually recovering. Working in theory, this clearly represents a bottleneck for realtime use on smartphones. Alternatively, the uncertainty (here: σ_{turn}) could be increased, ensuring that a decent number of particles remains within the corridor. However, when increasing uncertainties for all predictions, the ones with valid sensor observations are rendered unnecessarily uncertain, causing a broad and growing density. Ideally, the false reading is detectable in some way, increasing uncertainties only when needed. For the presented example, where the fault is caused by the pedestrian shaking the phone, the uncertainty could be adjusted as described in the end of section 2.4.2. A corresponding example is shown in figure 5.12, where the false reading between the first two predictions is detected, and addressed by increasing σ_{turn}. In comparison to figure 5.11, the particles after the second prediction now form a much wider density, including many particles with the correct walking direction. As the following sensor observations are accurate again, all subsequent steps continue from this broader density using a smaller σ_{turn}. Even though some particles enter the room, the majority uses the correct path along the corridor. While all three variants – stopping in front of obstacles, using zero-weights and resampling, using a coarse prediction followed by evaluation and resampling – have improved, only the zero-weights method (bottom row) produces a desirable result. Here, all trapped particles are eventually removed, as they do not match with subsequent observations. The evaluation-based approach (top row), which does not use observations within the prediction, provides almost the same estimation result, but with a less distinct particle set. The variant with the particles stopping in front of obstacles (center row) suffers from the same drawbacks as already indicated by figure 5.9, thus being impractical.
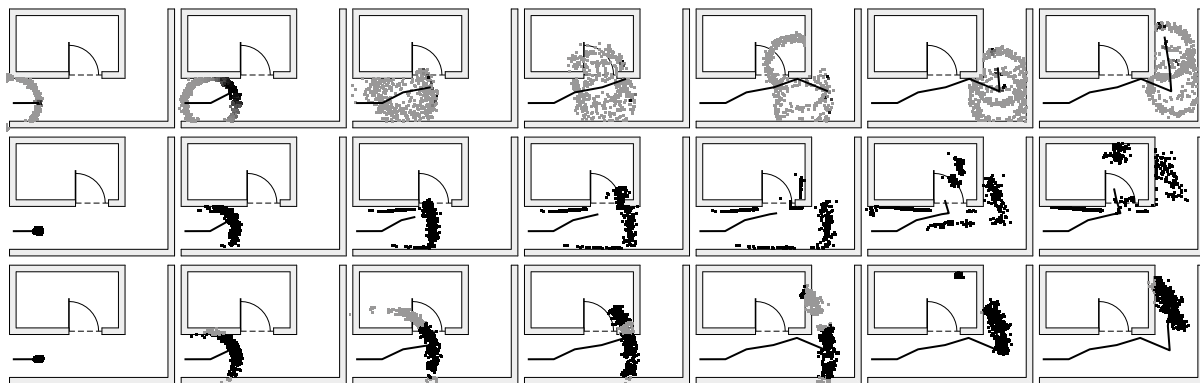
Figure 5.12: Example with a sensor fault between the first and the second prediction. If this fault can be detected, and the sensor-uncertainty is adjusted accordingly (compare against figure 5.11), getting stuck can be mitigated for all three cases: predicting movements into all directions followed by evaluation and resampling (top row), predicting based on the observation and stopping in front of obstacles (center row), and predicting based on the observation, down-weighting invalid movements (gray) and removing them during resampling (bottom row).

5.2.4 Detecting and Handling Deadlocks

However, as already mentioned briefly, not all types of sensor faults are detectable. Especially for the Wi-Fi component, which is the only sensor providing absolute location information indoors, potential error sources are numerous. The pedestrian, holding the phone, might attenuate signals due to the shadowing of the body. Depending on the type of antenna installed within the smartphone, tilting it might alter receivable signal strengths as well. When signal strength prediction models are used, they can differ significantly from real-world readings. For all those cases, there is no clear indication whether some observation is suitable or erroneous. Without adjusting uncertainties accordingly, particles can easily get stuck, depending on their surroundings, requiring other mitigation approaches [Fet+17; Ebn+17].

One option is to detect such deadlock situations, e.g. when the random walk indicates that many unlikely edges were taken, too many particle movements triggered a positive intersection test, or the largest weights after an evaluation are below a certain threshold. The most simple solution is then given by restarting the particle filter. That is, uniformly distributing all particles throughout the walkable area, assigning each one a uniform random heading $q_0^{(\Theta)} \sim \mathcal{U}(0, 2\pi)$. Restarting is basically the same as for every starting situation, where the current whereabouts are unknown. This was shown in figure 5.6, starting with a uniform distribution at $t = 0$, scattering all particles throughout the walkable area, using an unknown heading. Restarting addresses the deadlock, but causes the next few estimations to be unusable. Yet, as within figure 5.6, after several steps, impossible movements will cause many particles to be replaced during resampling. As soon as new Wi-Fi observations are available (as for $t = 3$ in figure 5.6), additional

locations can be ruled out, assuming the sensor works as expected. Until $t = 5$ the density became compact, and starts to resemble the actual ground truth. While taking some time to recover, the deadlock is addressed. When absolute location observations are unavailable, recovery strongly depends on the size and complexity of the floorplan, and the number of particles used. For a large multistory building it is unlikely that a uniform distribution of a few hundred particles contains the actual whereabouts, thus requiring better solutions.

To reduce the time needed for the system to stabilize, and prevent invalid estimations after a restart, a heuristic could be applied. It is unlikely for the pedestrian to have moved across the whole building or several floors within a short timeframe. Thus, instead of distributing all particles uniformly, the stuck density could be scattered within a certain radius. Alternatively, only a fraction of the stuck density could be replaced by uniform samples, approximately keeping the last whereabouts, but allowing for other locations as well. Although, for this case, it is unclear how to weight the old and the new samples, and neither of the two methods represents an analytically profound solution.

One solution is the Interacting Multiple Model Particle Filter (IMMPF) [DB05], where a second, less restrictive, filter is added besides the particle filter with floorplan-knowledge. That is, e.g. a particle filter without floorplan considerations. With the second filter being less restrictive, it can not get stuck, but the quality of its estimation is reduced. The intention is to use both filters in parallel, based on the same input data. When a major difference between both filters is detected, the restrictive version might be stuck. This difference is determined analytically, e.g. by the Kullback-Leibler divergence [KL51], estimating the dissimilarity of two densities. For every filter update, a new set of particles is derived by mixing both based on their divergence. The one with floorplan knowledge is *dominant*, and therefore preferred. When the divergence is large, and the dominant one got stuck, the second, or *support*, is preferred instead. This is achieved by placing an exponential distribution on top of the Kullback-Leibler divergence d_{KL}

$$p_{\mathrm{dom}} \left(d_{\mathrm{KL}} \right) = e^{-\lambda d_{\mathrm{KL}}} \,, \quad p_{\mathrm{sup}} \left(d_{\mathrm{KL}} \right) = 1 - p_{\mathrm{dom}} \left(d_{\mathrm{KL}} \right) \,, \quad \lambda > 0 \,, \tag{5.1}$$

defining the probability for drawing a particle from either of the filters. This implements previous discussions on filter restarts, and how to prevent them from being too strict. The IMMPF can be thought of as providing an alternative density with tolerable quality that can be referred to, when the, usually better, dominant density got stuck. This provides a concept similar to restarting, without scattering particles within an unnecessarily large region. Compared to previous heuristic ideas, this variant is based on the well-researched Kullback-Leibler divergence, and requires only a single empiric value λ, deciding when to mistrust the dominant filter [Fet+17].

5.3 Real-World Considerations

Besides filter and sensor related issues, other aspects, affecting the system's estimation quality and computational performance based on a building's actual architecture, available infrastructure and similar, must be considered as well. Within the following, common pitfalls and potential workarounds are briefly examined.

5.3.1 Sensitive Locations

As discussed in section 2.7, most buildings will contain locations, where the estimations from signal strength prediction models are significantly erroneous. In such places, the evaluation of the Wi-Fi component will not provide viable results, yielding aforementioned drawbacks, of the system potentially getting stuck. Using special signal strength prediction models, limited to the sensitive region, can address such issues by explicitly modeling local signal strength behavior (cf. section 2.7.6). This, however, might yield new problems, in areas where two regional models are adjacent to each other. Due to completely different signal strength estimations within both regions, the evaluation for one of both will often be very unlikely compared to its neighbor. The discontinuity between both models can cause particles to wait within one of the two regions, until the smartphone's readings eventually agree. This yields new problems, when not matching with other sensor observations, such as step-detection or turn-detection [Ebn+17].

Stairwells often represent a sensitive area, with their steel-reinforced concrete significantly attenuating access points. As Wi-Fi reception is not required in stairwells, installing transmitters within is unreasonable. For indoor localization, battery powered supplementary devices, such as Bluetooth beacons, are a viable alternative, providing signal strength-based localization within areas of the building, where Wi-Fi is too weak, or model predictions inaccurate. Yet, they demand for additional maintenance, and can suffer from signal strength depletion based on their battery level. Installing them within the whole building is thus not advisable. For sensitive areas, however, they provide valuable additional localization information [Ebn+15].

Nevertheless, the observations from supplementary transmitters compete with the divergent Wi-Fi-readings, potentially still causing localization errors. If so, Wi-Fi should be considered unstable, relying solely on the supplement, such as aforementioned Bluetooth beacons. Often, more than one access point is affected by local effects within sensitive areas, e.g. yielding generally weak RSSIs, except for a few transmitters. Whenever this constellation is encountered, the Wi-Fi component can be assumed unstable, and temporally excluded from the evaluation.

Situations, where disabling is advisable, can be detected by a quality heuristic [Ebn+17]

$$\text{quality}(\boldsymbol{s}) = \max\left(\min\left(\frac{\mathbb{E}(\boldsymbol{s}) - \tau_{\text{wMin}}}{\tau_{\text{wMax}} - \tau_{\text{wMin}}}, 1\right), 0\right). \tag{5.2}$$

(5.2) maps the mean signal strength of all currently visible transmitters to a range of $0 \leq$ quality$(\boldsymbol{s}) \leq 1$, by using two empiric thresholds τ_{wMax} and τ_{wMin}. If its result drops below a certain value, Wi-Fi is disabled. The three thresholds can e.g. be determined by using empirically chosen values, or by estimating them within the model optimization process, letting the metric produce low outputs, whenever the model estimations are especially erroneous [Ebn+17]. A more profound, yet more complex approach is e.g. given by training an artificial neural network, converting RSSI readings into a quality indication. Training data is e.g. given by the error between reference measurements and model estimations (2.112), or by evaluating the position of the reference measurement against the optimized model (2.111). Again, invisible and non-permanent transmitters must be considered to provide viable results.

Besides completely disabling the Wi-Fi sensor, its impact can be limited. The uncertainty within the evaluation $p_{\text{wifi}}\left(\boldsymbol{o}_t \mid \boldsymbol{q}_t\right)$ can e.g. be increased, to address expected errors. Whenever the Wi-Fi component yields erroneous evaluations, there is a risk of the particle density getting stuck within the building, due to the restricting floorplan. Another alternative is thus given by temporarily disabling obstacles within the transition process, when a low Wi-Fi quality is detected. As the metric is not binary but continuous, both variants, with and without obstacles, can be mixed continuously. Whenever the metric indicates viable Wi-Fi readings, obstacles provide additional localization quality. For weak Wi-Fi signals, they are omitted, ensuring the localization process does not fail by getting stuck. That is, the Wi-Fi quality serves as an additional mixing constraint to the IMMPF, deciding which filter to trust [Fet+17].

5.3.2   Data Acquisition

Another aspect is data acquisition. While this relates mainly to using the system for localizing or navigating with a smartphone, it also concerns the setup phase, e.g. conducting Wi-Fi reference measurements (see section 2.7.6) within a building, where the phone can be used as well.

Sensor sample rates, like for the IMU and barometer, can be controlled to some extend, by selecting from a few pre-defined options (Android). Yet, actual rates can vary, and are hardware dependent. Requesting the fastest sample rate often is a safe choice. The associated arithmetic operations are either inexpensive, or downsampling is easily possible, e.g. by cumulating consecutive readings (gyroscope), or simply discarding some observations (accelerometer, magnetometer). The barometer typically uses low sample rates, not requiring adjustments [Bos15].

For Bluetooth beacons, behavior is slightly different, as the detection is usually performed passively. That is, the smartphone scans without actively requesting data, and beacons actively send advertisements, without being asked for them. With each beacon controlling its own advertisement interval, their sample rate can not be controlled by the phone, and varies among transmitters. Common rates are somewhere near $2\,\mathrm{Hz}$, representing a tradeoff between battery life, accuracy, and not polluting the radio channel [Lin15; Mic18].

As discussed, scanning for nearby Wi-Fi transmitters is performed actively, requesting a response from access points on every supported channel. Achievable sample rates thus strongly depend on the number of channels to scan. While using both, $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$, provides more readings, and thus a potentially better location estimation, including the $5\,\mathrm{GHz}$-band significantly increases scanning times [Ebn+15; Ebn+17]. For some Android devices, $5\,\mathrm{GHz}$ can be disabled within the Wi-Fi settings, to increase the sample rate. If this option is unavailable, there is a chance it is still available as hidden API, accessible via reflection. However, for more modern devices, both options tend to be unavailable. Also, starting with Android 9, the Wi-Fi scanning rate is limited to a few times per minute, considerably reducing usefulness. This can be addressed by reverting to Bluetooth beacons, not affected by this limitation, and supported by most devices, or by using the upcoming FTM (cf. section 2.7.7), if supported by used smartphones and existing infrastructure. Android version 9 thus faces an important restriction, limiting the use of already installed infrastructure, requiring for new hardware, unnecessarily increasing the price, rendering indoor localization less pervasive and affordable. As of today, manual scanning for Wi-Fi transmitters is not allowed by the iOS operating system. Similarly to Android version 9, Bluetooth beacons can be used instead, with future versions of hardware and operating system eventually supporting Wi-Fi FTM as well.

Reference measurements, required for training signal strength prediction models (cf. section 2.7.6), can be conducted similarly, using commodity smartphones. For an adequate sample of real-world behavior, the whole walkable area of a building should be covered [Ebn+17]. By using the semantic floorplan described earlier, actual locations can be added to the map, ensuring the position stored within a database matches with the location in the building. To ensure a stable signal strength estimation, several measurements are conducted at every such location, similar to fingerprinting [BP00; Shu+15; Men+11]. Effects of the phone's antenna can be compensated by using different poses while measuring, e.g. by slowly turning, holding the phone upfront. The resulting database can hereafter be used to estimate the parameters required for a chosen signal strength prediction model. Being used for all upcoming experiments and evaluations, this approach was also applied for participating in the *IPIN 2016 Indoor Localization Competition*, taking around 60 minutes of recording time within the competition building, with the presented system providing the best results among all competitors [EvA16].

5.4   Performance Considerations

The goal is to provide an indoor localization and navigation system that is accurate, easy to set up, inexpensive, and suited for smartphone use. Memory and performance aspects thus play an important role besides financial considerations for required infrastructure, setup and maintenance times. Fingerprints, for example, generally provide the most accurate Wi-Fi results, due to real-world measurements (cf. section 2.7.5). If conducted for every square meter of a building, with an average of ten access points visible at every fingerprint, and only their average signal strength stored as a single byte, a floor of $100\,\text{m}$ by $100\,\text{m}$ in size requires $\approx 100\,\text{KiB}$ of memory. Thus, the amount of data is manageable for most buildings, even when decimal numbers, Gaussian distributions or histograms are stored (cf. section 2.7). Yet, a major issue, besides the time required for conducting fingerprints, are computational requirements within the evaluation, when e.g. a KDE is used to determine the probability for every particle, depending on the likelihood of surrounding fingerprints. Based on the number of fingerprints and particles, the computational complexity can easily exceed the abilities of embedded devices.

5.4.1   Precomputed Model Predictions

When replacing fingerprinting with e.g. the log-distance model from section 2.7.6, computation times to predict the signal strength at the location of each particle, for every currently visible access point, are minimized to a few arithmetic operations. This comes at the cost of reduced localization quality, with the log-distance model being less accurate than fingerprints. A mitigation is given by using the extended log-distance model instead. However, its complexity depends on the number of obstacles included within the prediction process (cf. section 2.7.6). When solely floors and ceilings are considered, there is only a slight overhead compared to the log-distance model. Yet, including all types of walls within each floor significantly increases computation time, requiring intersection tests for all obstacles along the line of sight from each particle towards every visible access point. Depending on the complexity of the floorplan, and the number of particles, embedded devices might fail to perform required computations in time.

Similar to solving the problem of costly shortest path calculations, the spatial models from section 3.4 can also be used to store precomputed signal strengths, at the cost of memory. Within the navigation grid, the result from a signal strength prediction model can be stored directly onto each vertex. The evaluation is then reduced to inexpensive lookups and comparisons, denoted in figure 5.13a. However, the amount of required memory can be significant. When vertices are placed every $25\,\text{cm}$, the $100\,\text{m}$ by $100\,\text{m}$ wide example floor, with ten access points visible per location, already requires $\approx 1.5\,\text{MiB}$, when storing a single byte per transmitter. However,
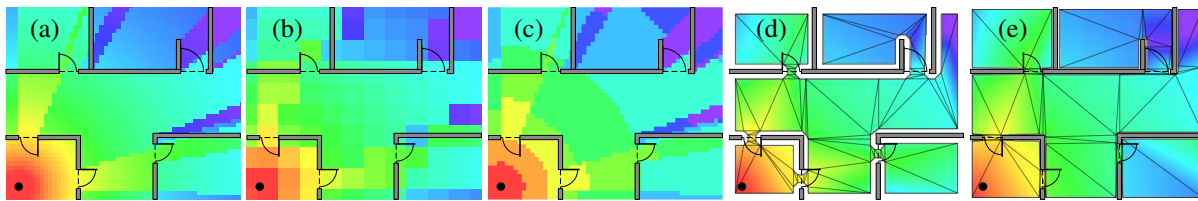
Figure 5.13: Varying strategies for precomputed signal strengths. Storing a value every few centimeters, e.g. directly on grid vertices (a). Using larger distances between predictions, still providing fast lookups (b). Grouping similar predictions, storing references to them on grid vertices (c). Attaching values to triangle edges of the navigation mesh followed by barycentric interpolation (d) and (e).

to maintain the signal strength prediction model's ability to estimate a signal strength for *every* transmitter and location, all access points within a building would have to be stored on every single vertex. This is unfeasible for larger buildings with many transmitters.

Typically, only a few transmitters are actually visible at any location within a building (cf. figure 2.27). One option to address memory consumption is thus given by storing only meaningful transmitters, with a prediction above some threshold. That is, if a transmitter's predicted signal strength for the location of some vertex is below this threshold, it is not stored, but assumed constant, using the value of the threshold. This significantly reduces memory requirements. The error that results from using a constant instead of actual predictions can be limited by choosing this constant to be near typical receiver sensitivity, where signals often are unrecognized by the hardware anyway.

Also, signal strengths do not change substantially within $25\,\mathrm{cm}$, at least when no obstacles are nearby. A less dense data-structure can thus be sufficient, e.g. storing predictions within a separate $1\,\mathrm{m}$ by $1\,\mathrm{m}$ grid. However, this introduces the risk of not aligning with obstacles. Indicated within figure 5.13b, there are many locations where the same signal strength is predicted in front of and behind a wall, potentially causing errors within the evaluation.

Ideally, locations are only grouped together when they share similar signal strength predictions for all transmitters, yielding an irregular data structure. While being feasible for a single transmitter, shown in 5.13c, there is no memory benefit. To the contrary, additional memory is required to store both, each group, and a reference to it for every lookup location. The size of this reference depends on the number of distinct groups, which in turn depends on the number of installed transmitters. The chance for two nearby locations to have a similar signal strength prediction decreases with an increasing number of transmitters, due to physical effects discussed in section 2.7. This potentially causes many small groups, not saving memory at all. Besides, while grouping can reduce the required amount of memory, it introduces discreteness. With each prediction belonging to a larger area, discontinuous jumps between adjacent groups are generated, clearly visible within figure 5.13b and 5.13c, which negatively affect the evaluation.

When the navigation mesh is used as spatial data structure, recuperations are different. Similar to vertices, precomputed RSSIs can be stored onto each triangle-corner within the mesh. Due to using only a few triangles, in contrast to many grid vertices, memory savings are significant. Furthermore, this also reduces the effect of discontinuity, as signal strengths can be estimated by *barycentric interpolation* within the triangle a particle belongs to, increasing evaluation quality. A corresponding example is shown in figure 5.13d and 5.13e. However, interpolated predictions can still be significantly different from actual model values. As RSSIs are calculated only for triangle-corners, results strongly depend on their placement. This can be verified by comparing figure 5.13d and 5.13e, where the lower and upper right side clearly differs. While larger triangles reduce the memory footprint, they also diminish quality, not aligning with or resembling the actual signal strength behavior. When enforcing smaller triangles, this strategy is viable for most locations within a building, well-aligning with obstacles. This type of interpolation can also be used for making fingerprints continuous, without requiring a costly KDE. Yet, conducted fingerprints can not be assumed to align with an algorithm's triangle placement, thus being unclear which RSSIs to assign to every triangle corner. One approach is e.g. given by inverse distance weighting [AC16], estimating the RSSIs for a triangle corner based on nearby fingerprints and their distance. Here, obstacles can be considered, ignoring all fingerprints where the line of sight is occluded. Besides its computational efficiency due to being precomputed, considering obstacles denotes another advantage to the KDE-based approach, which is not directly able to consider walls and other obstacles (cf. section 2.7.5).

As can be seen, the strategy for including the Wi-Fi component strongly depends on required quality, size of the floorplan, number of installed access points, and available computation power. For buildings with many drywalls, signal strength behavior is rather continuous and an extended log-distance model just considering floors and ceilings, is fast and sufficient [Ebn+17]. Besides improving model predictions in surroundings with complex architecture, reference measurements can also be used to determine, whether more complex models and precomputation can provide a benefit.

5.4.2 Code Optimization

Besides engineering or deciding on appropriate algorithms and data structures to perform required calculations, their actual implementation also plays an important role, affecting both, performance and memory requirement, briefly discussed within the following.

Programming Language One aspect is the programming language used for implementing all previously discussed models and calculations for smartphones. Especially when referring to Android-based smartphones, mainly programmed using Java or Kotlin [Lin+11]. Perfor-

mance impact is often workload dependent, subjective, and thus hard to argue scientifically and in general [LZQ15]. Therefore, the following is mainly based on experience, gathered while developing the overall system. The implementation was solely based on native C++ code. A decision also made for portability reason, supporting Linux, Android and iOS. This also allowed for operator overloading, simplifying the code for statistics, filtering, and estimation. Performance and memory considerations were also part of this decision. For Android and Java, early setups indicated significant overheads, especially related to the particle filter. This was also due to the unavailability of plain memory objects, such as C structs, and not being able to store scalar types, like `int` or `float`, within lists or similar data structures.

Particles and Multithreading Besides programming language, the number of particles represents a critical aspect, affecting overall system performance. As prediction, evaluation, and estimation are performed per-particle, their number has a direct influence. While using less particles enhances battery life, it negatively affects the quality of the recursive density estimation. As shown within previous figures, a decent number is required for a proper initialization, and to capture all likely whereabouts. It thus represents a tradeoff between quality, battery life, and available computational power. The presented system used between 5000 and $10\,000$ particles, which is way more than used by competitors [NRP16; KS18; Gui+16; Shu+15]. This was achieved by optimized C++ code, and by leveraging the previously mentioned fact, that the particle filter is parallelizable (cf. section 4.5). Both, the prediction and evaluation step process particles in parallel, using multiple threads, well suited for today's multi-core smartphones.

5.5 Summary

This chapter provided the missing links between the required theory, presented within the three previous chapters. Therefore it concentrated on less scientific and more technical aspects, related to the overall system implementation.

The first section was related to modeling a building's floorplan, suitable for deriving one of the spatial data structures discussed for the movement models in chapter 3. Main focus was on a fast creation process with support for later modifications. It thus referred to 2D drawings per floor. Yet, the result is still suitable to derive accurate 3D representations, including doors and windows, for both, visualization and e.g. ray tracing. Furthermore, it allows including semantic information, like the type of ground, required for movement predictions.

Hereafter, the overall system was assembled, choosing the particle filter as implementation of the Bayes filter for the recursive density estimation process. This part described the way sensors are included within the overall system, using the step-detection as pacemaker, triggering the

filter update procedure. In contrast to typical convention, turn-detection was also used directly within the transition process, providing more robust movement predictions, and suppressing sample impoverishment. To reduce computational complexity, eCompass, activity-detection, Wi-Fi, Bluetooth beacons, and GPS were used within the evaluation step, well suited for resulting multimodalities. While this decision seems like a drawback in terms of density accuracy, this represents a viable tradeoff between quality and required computational complexity.

Furthermore, due to the floorplan, the system's prediction already is rather restrictive. Potential ways for handling impossible movements were discussed. Especially sensor faults, such as an invalid turn-detection, can easily cause all particles to get stuck within a room. When this situation occurs, recovery often is unlikely. Techniques for handling such situations were briefly introduced, including their advantages and disadvantages. While a simple uniform restart is possible, it suffers from various drawbacks, and should thus be avoided.

The next part concentrated on real-world aspects, and how to ensure a decent absolute location estimation from radio transmitters, even within sensitive areas of a building. This was achieved by either adding additional transmitters within those areas, or by adjusting the used signal strength prediction models, using regional limitations or more advanced variants. This also lead to the question of data acquisition on smartphones. The topic was related to both, recording prior reference measurements, and actual observations when the pedestrian uses the localization and navigation system. Both aspects were briefly described, explaining how reference measurements were conducted, and how sensor observations are sampled, adjusted, and included within the filter.

With the quality of the particle filters approximation depending on the number of particles, this lead to the final question on how to handle a decent number of particles on today's smartphones. By using precomputations where applicable, a suitable programming language, and multithreading, several thousand particles are manageable, sufficient for approximating the complex behavior of pedestrians walking indoors.

Chapter 6

Experiments

Previous chapters examined smartphone-based pedestrian indoor localization and navigation from different points of view.

First, individual sensor components and their contribution were discussed in chapter 2. Their observations were used within a probabilistic relation. Either to evaluate potential pedestrian whereabouts, or their changes, based on analytical constraints. Hereafter, pedestrian movement predictions based on a building's floorplan were examined in chapter 3. They were used to estimate the likelihood of certain movements, based on obstacles and the walkable surface. This also allowed for a continuous 3D estimation, when utilizing appropriate spatial models. The link between both viewpoints is recursive density estimation, introduced in chapter 4. Here, several potential variants, as well as their advantages and disadvantages, were discussed. While analytical implementations are computationally inexpensive, they are too restrictive, unable to consider the floorplan. Therefore, the particle filter was introduced, approximating probability densities using multiple samples, well-suited for considering the floorplan. However, at the cost of increased computational complexity. Afterwards, chapter 5 presented several real-world considerations. This concerned the 2D/3D setup of floorplans, and how to handle their obstacles in a probabilistic and computationally efficient manner. Therefore, the filter's update interval, and the inclusion of sensor observations were discussed in detail. Closed by remarks on the programming language's impact of achievable performance.

Finally, this chapter provides an experimental viewpoint of all previous aspects. First, the testbed is described in detail, covering used devices, examined buildings, data acquisition, and ground truth estimation. Following the preceding structure, sensors and movement models are hereafter evaluated separately, estimating individual contributions, and limitations, before examining their combination.

All tests also refer to synthetic and constrained setups, minimizing external influences while focusing only on the aspects that are essential for a component. Additionally, actual real-world scenarios are examined, and, if applicable, compared against the findings from their synthetic counterparts. Hereafter, the overall indoor localization and navigation system is examined, backed by all initial findings. These tests are based on actual pedestrian walks, conducted within multiple buildings, where indoor localization and navigation provide a benefit. To ensure representative results, all experiments were performed with multiple smartphones that vary in price and age, and actual pedestrian walks were conducted by different persons.

6.1   Testbeds and Data Acquisition

This section provides an overview on used devices, and synthetic/real-world scenarios, required for examining all system components. Missing details are provided later, for each test setup.

Devices   In terms of pervasiveness and price, as of today, Android-based smartphones represent the most desirable platform to support initially. Disallowing scanning for nearby Wi-Fi transmitters, iOS-based smartphones lack an important sensor, which is crucial for indoor localization and navigation. Even though supporting Bluetooth beacons, also usable for inferring absolute whereabouts, most buildings are not equipped with the additionally required infrastructure. Thus, iOS-based smartphones were not considered within this work. Based on their relatively low market share, the same holds true for phones using a Microsoft operating system. Without loss of generality, all tests are based on several different Android smartphones, of all brands, prices and ages. The diversity of phone vendors brings varying sensor brands, different internal assemblies, and various operating system versions. The chosen devices thus provide a broad spectrum for the system to deal with, enforcing general solutions, preventing overfitting. Table 6.1 lists all smartphones used within evaluations, their initial date of release, used operating system version, and a brief overview on supported sensors. In general, recent models support Wi-Fi in all relevant frequency ranges, Bluetooth Low Energy – required for Bluetooth beacons to be detected – and a gyroscope for fine-grained heading estimations. Solely the barometer is rarely found. Accuracy and precision of these sensors will be examined later.

Syntehtic and Constrained Testbeds   To examine components individually, special setups were constructed, minimizing external influences, focusing only on important aspects, and providing a well-known ground truth for error estimations. For step-detection, this covers pedestrian walks following a defined pattern, with a known number of steps. For turn-detection by the gyroscope, and absolute heading estimation from the magnetometer, a turntable is used, allowing for exact angular movements, providing an ideal ground truth. Additional pedestrian walks

Vendor	Model	Manufacturer	Year	Android	Acc	Gyr	Mag	Bar	2.4	5	BLE
Motorola	Milestone 2	Motorola	2010	4.4.4 [1]	✓	-	✓	-	✓	-	-
Motorola	Milestone 4	Motorola	2012	7.1.2 [2]	✓	-	✓	-	✓	-	-
Samsung	Galaxy S3 mini	Samsung	2012	5.1.1 [3]	✓	✓	✓	-	✓	✓	✓
Google	Nexus 6	Motorola	2014	7.1.1	✓	✓	✓	✓	✓	✓	✓
Samsung	Galaxy S5 Neo	Samsung	2015	6.0.1	✓	✓	✓	-	✓	✓	✓
Motorola	Moto Z	Motorola	2016	9.0.0 [4]	✓	✓	✓	-	✓	✓	✓
LG	G6	LG	2017	8.0.0	✓	✓	✓	✓	✓	✓	✓
Google	Pixel 2 XL	LG	2017	9.0.0	✓	✓	✓	✓	✓	✓	✓
BlackBerry	KEY2	TCL	2018	8.1.0	✓	✓	✓	-	✓	✓	✓

Table 6.1: Overview on examined smartphones, their date of initial release, currently running operating system version, and supported sensors: accelerometer, gyroscope, magnetometer, barometer, Wi-Fi 2.4 GHz/5 GHz and Bluetooth Low Energy (BLE), required for Bluetooth beacons.
[1] CyanogenMod 11, [2] LineageOS 14.1, [3] CyanogenMod 12.1, [4] LineageOS 16.0

allow for a visual impression on the corresponding behavior under real-world conditions. For the barometer, walks through a stairwell with known altitudes provide the ground truth for error estimation. Similarly, activity-detection was performed based on walks annotated with labels, describing the pedestrian's current activity. For Wi-Fi, brief constrained tests were performed, as external influences can not fully be prevented. These tests refer to general signal strength behavior, providing an insight on the quality of prediction models. For actual model optimization and error estimation, real-world setups are required. Movement prediction and navigation are examined based on synthetic floorplans. While not always being realistic, they provide a visualization of the model's behavior, translatable to real-world scenarios.

Real-World Testbeds   Besides synthetic and constrained testbeds, Wi-Fi predictions, absolute location estimation, and the overall system are examined within real-world scenarios, covering three completely different buildings, varying in age, materials, floor layout, and infrastructure.

The SHL, shown in figure 6.1, is a modern university complex, completed in 2011. It is divided into a main building, with four stacked floors, and a two story annex. While the first floor is directly connected, the 2nd floor is separated by a small outdoor area, shown in the upper right of the figure. The walkable area covers $\approx 12\,000\,\mathrm{m}^2$ when including outdoor areas, and $\approx 8500\,\mathrm{m}^2$ without. Walls are mainly made of drywall, using steel-reinforced concrete only for stairwells, and in places where structural stability is required. The whole complex is equipped with 34 access points, using both, the $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$ range. Due to legal issues, their locations must not be disclosed, and are therefore omitted within the figures.

The Museum 1 (cf. figure 6.2) was built in 1908, with three floors, each split into two halves. That is, there is a small change in altitude between both halves, requiring short stairs. Besides being separated, the building is characterized by a predominant concrete structure, using only a

few drywalls. The walkable area is $\approx 3200\,\mathrm{m}^2$. While a partial Wi-Fi-infrastructure is available, it does not cover the whole walkable area. Therefore, additional Wi-Fi-beacons, based on the `ESP8266`, were installed, yielding 41 transmitters for the absolute location estimation.

The `Museum 2` in figure 6.3 was built as a convent in the 13th century, and serves as a museum since 1936. Besides one main floor, there is no clear structure, and the seven annexes are placed at various altitudes and locations. Being directly attached to the old town wall, the building is characterized by massive concrete walls, using only a few drywalls. When including the garden, $\approx 3000\,\mathrm{m}^2$ are walkable, and $\approx 2400\,\mathrm{m}^2$ without. As there is no existing Wi-Fi infrastructure at all, 45 `ESP8266` Wi-Fi-beacons were installed to provide absolute location estimation.

Required Wi-Fi reference measurements were recorded by a pedestrian, residing at well-known positions, holding the smartphone upfront, slowly turning around in a circle, while the device scans for nearby transmitters. This yields multiple measurements for each single location, reducing the influence of the antenna and the human body. The number and locations of these measurements are presented, when examining the Wi-Fi component.

Within the shown buildings, various walks (see figure 6.4, 6.5 and 6.6) were conducted by several pedestrians, using different smartphones. The ground truth required for error estimation is provided by the pedestrian: All walks are defined by points with exactly known location, connected to a path. Whenever reaching such a point, the pedestrian added a label to the recording. Assuming a constant walking speed, this allows for interpolating a position along the path, for any point in time. Even though being inexact, accuracy is sufficient for the intended use.

Shown in figure 6.4, 6.5 and 6.6, the conducted walks cover completely different scenarios. Four of the five walks within `SHL` are close to a *shortest path from A to B* scenario. Only `walk A4`, a two-story loop using a stairwell, is mainly synthetic. While some walks within `Museum 1` and `Museum 2` resemble a shortest path, most of them were conducted with *exhibition watching behavior* in mind. That is, the pedestrian carrying the smartphone behaved like a normal visitor, sometimes resting in front of exhibits. The vertical lines within `Museum 1` are due to using an elevator to change the current floor. Thus, all examined walks represent various real-world scenarios for an indoor localization and navigation system, examined within the following.
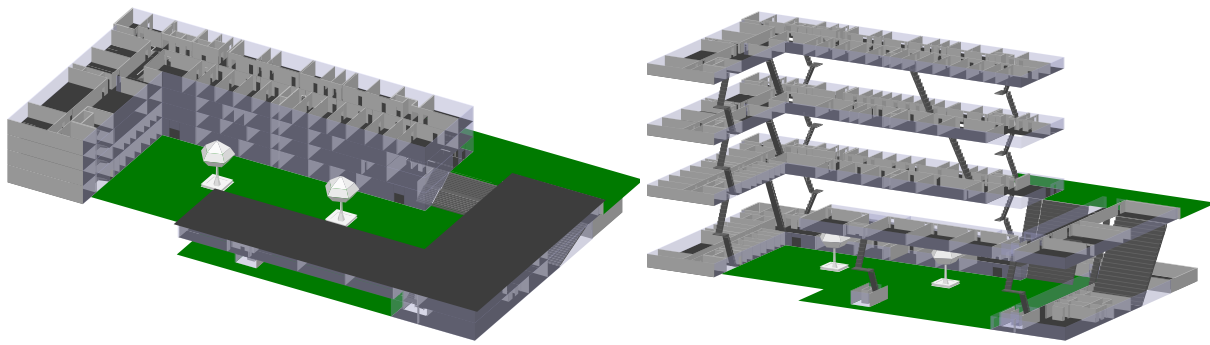
Figure 6.1: SHL – University of Applied Sciences Würzburg-Schweinfurt. Modern building completed in 2011, with four stacked floors, surrounded by a facade of metallized glass, approximately $75 \times 50\,\text{m}$ in size (upper left). Via an outdoor area, the main building is connected to a two story annex (lower right), with an overall size of approximately $110 \times 60\,\text{m}$. The walkable area spans $\approx 12\,000\,\text{m}^2$ when including outdoor areas, and $\approx 8500\,\text{m}^2$ when not. The right half shows an increased distance between floors.
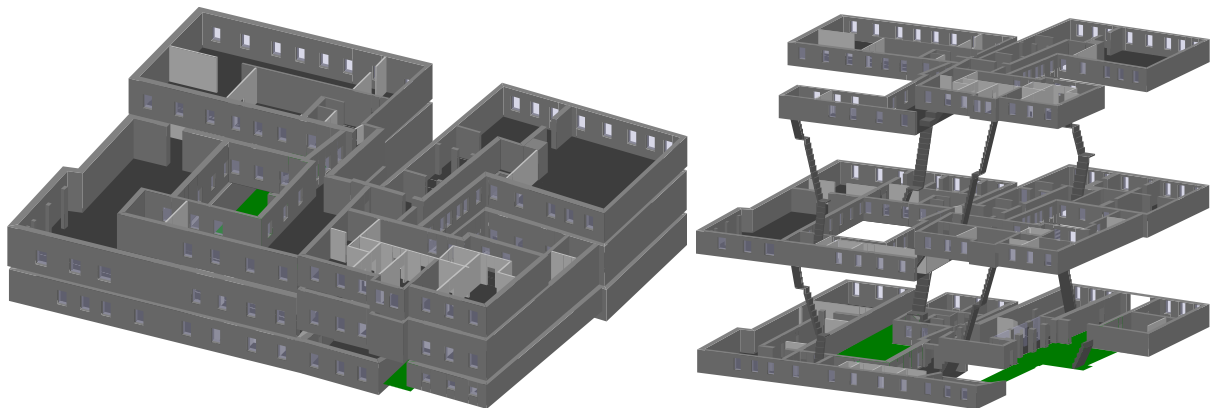
Figure 6.2: Museum 1 – Gäubodenmuseum Straubing[a]. A museum built in 1908, with three floors, each split into two halves (see right figure). Between the two halves, there is a small offset in altitude, yielding an uneven distribution of the floors. The museum's overall size is approximately $55 \times 40\,\text{m}$, with the walkable area being $\approx 3700\,\text{m}^2$. The right half shows an increased distance between floors.
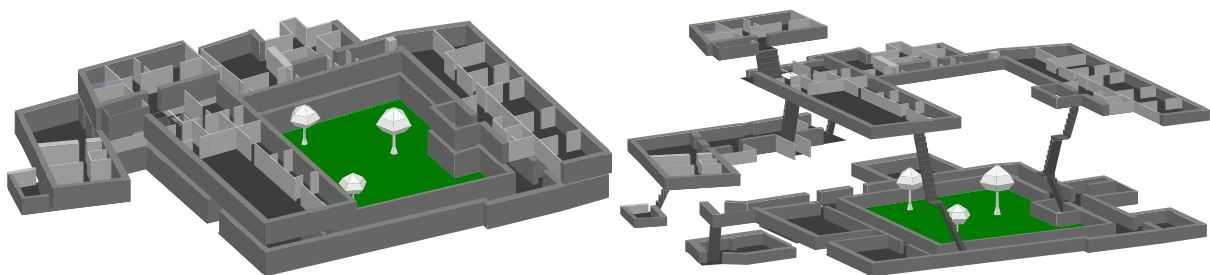
Figure 6.3: Museum 2 – RothenburgMuseum[b]. A museum since 1936, but initially built as a 13th century convent. Floors are arranged in various shapes, sizes, and altitudes, yielding a chaotic layout, with one main area, and seven annexes. Being directly attached to the old town wall, the building is characterized by massive concrete structures. The overall size is approximately $70 \times 50\,\text{m}$, covering $\approx 3000\,\text{m}^2$ when including the garden, and $\approx 2400\,\text{m}^2$ without. The right half shows an increased distance between floors.

---

[a]http://www.gaeubodenmuseum.de/

[b]https://www.rothenburgmuseum.de/
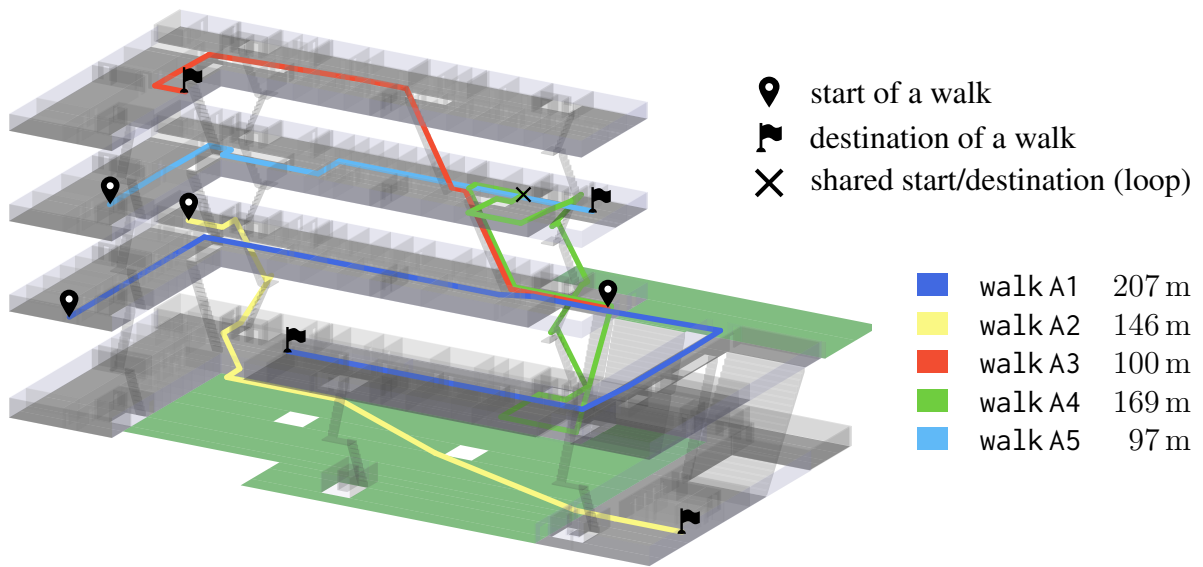
Figure 6.4: Pedestrian walks conducted within SHL.
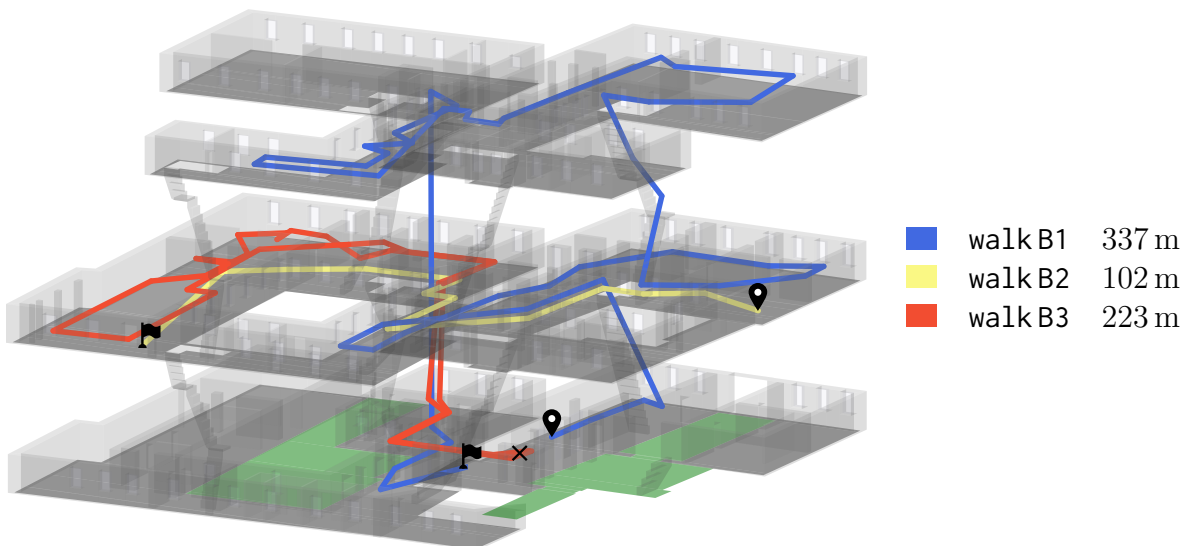
Figure 6.5: Pedestrian walks conducted within Museum 1.
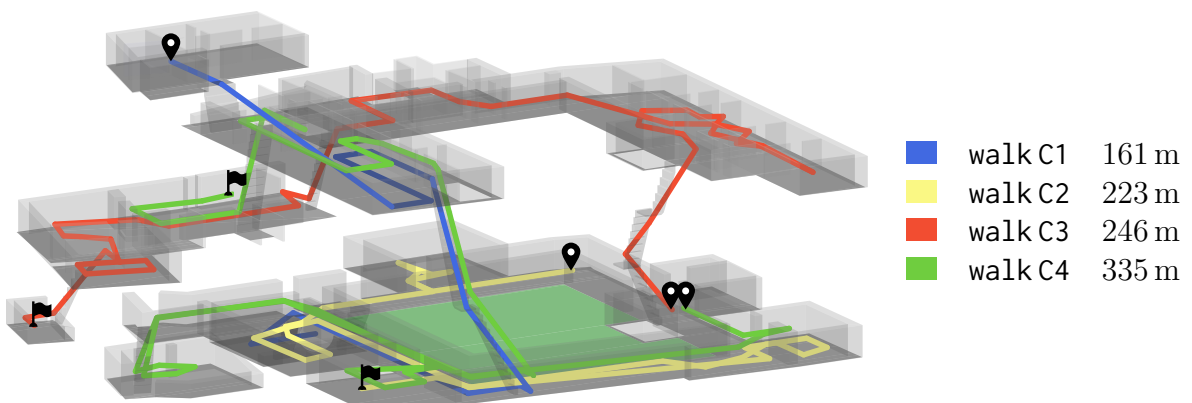
Figure 6.6: Pedestrian walks conducted within Museum 2.

Model	accelerometer		gyroscope		magnetometer		barometer	
Motorola Milestone 2	107.3	33.5	-	-	9.7	0.8	-	-
Motorola Milestone 4	48.0	52.1	-	-	95.2	15.5	-	-
Samsung Galaxy S3 Mini	101.2	17.5	101.4	17.6	49.3	4.5	-	-
Google Nexus 6	227.0	3.5	227.0	3.5	75.7	1.0	32.4	0.1
Samsung Galaxy S5 Neo	200.6	0.6	200.6	0.6	102.1	20.4	-	-
Motorola Moto Z	198.1	0.8	198.1	0.8	49.5	0.1	-	-
LG G6	398.3	2.2	398.3	2.2	99.7	5.5	26.0	0.5
Google Pixel 2	417.2	2.6	417.2	2.6	52.2	0.5	26.0	0.3
BlackBerry KEY2	208.0	0.7	208.0	0.7	50.0	0.2	-	-

Table 6.2: Mean sample rate (in Hz) and standard deviation of each IMU-sensor and the barometer, for all devices listed in table 6.1. A dash indicates a phone's lack of support for a given sensor.

6.2 Evaluation of Sensor Components

All sensor components are examined, following the preceding structure. As the GPS is already well-researched, and only of minor importance for indoor localization, no additional tests were performed. The focus is solely on IMU, barometer and Wi-Fi. As influences on the IMU sensors are numerous, synthetic benchmarks are conducted beforehand, analyzing the general quality of accelerometers, gyroscopes and magnetometers installed within smartphones. These examinations provide an impression on issues concerning accuracy and precision of relative and absolute heading estimations. After evaluating each sensor on its own, combined tests are presented, estimating the quality of pedestrian dead reckoning (PDR), when omitting sensor fusion, and absolute location indications. For Wi-Fi, the first step estimates the model prediction quality, when training them based on real-world reference measurements. Hereafter, the accuracy of location estimation is examined, when using these trained models.

6.2.1 Sensor Overview

Before presenting results for step-detection, turn-detection, eCompass, barometer and activity-detection, general parameters of smartphone sensors are briefly examined. For now, the Wi-Fi component is omitted. Table 6.2 provides an overview on sensor sample rates and the estimated standard deviations for various smartphones, when requesting the fastest rate possible. As can be seen, there are notable differences among the listed devices, concerning both, sample rate and standard deviation. Especially older phones tend to suffer from rather aperiodic sensor readings, potentially causing issues, e.g. within frequency-based filters, relying on fixed sample rates. While the examined modern smartphones provided stable sample rates, individual frequencies varied significantly. To ensure that results among different devices are identical, this must be considered for filters, such as the moving average or complementary filter (see section 2.4).

Type	Variant	Description
FIR	low-pass	gravity removed by subtraction, cut-off at $3\,\text{Hz}$, kernel size $N = 29$
IIR	low-pass	gravity removed by subtraction, cut-off at $3\,\text{Hz}$, $Q = 0.7$
FIR	band-pass	centered at $2\,\text{Hz}$, width of $\pm 1\,\text{Hz}$, kernel size $N = 101$
IIR	band-pass	centered at $2\,\text{Hz}$, $Q = 1.5$ (approximating the width of the FIR)
IIR	HLL-pass	high-pass ($>1\,\text{Hz}$) followed by *two* low-pass ($<3\,\text{Hz}$) filters

Table 6.3: FIR and IIR setups, used for filtering the accelerometer's magnitude (see appendix A.2).
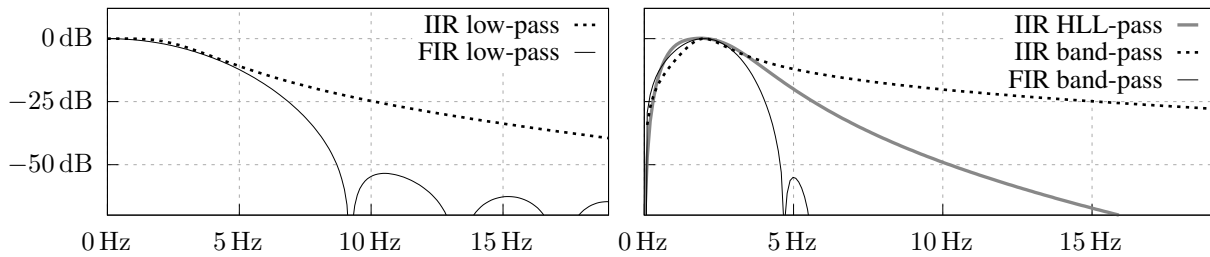
Figure 6.7: Frequency response of the FIR/IIR low-pass and band-pass described in table 6.3.

6.2.2   Step Detection

Discussed in section 2.4.1, readings from the accelerometer contain notable amounts of noise, especially when the smartphone is held upfront by the pedestrian. The contribution of filtering is examined for the two presented candidates, IIR and FIR. Determining the parameters required for (2.25), or a discrete convolution kernel, to create a low-pass, high-pass or band-pass is beyond the scope of this work. Equations and further details can be found in [Smi99; Smi11; WT06; Ror93]. Used setups are shown in table 6.3 and appendix A.2. Besides the IIR band-pass, a chained (1 high-pass 2 low-pass) IIR filter is used, providing more control on the width of the passband. Input to all filters was the accelerometer's magnitude, recorded by a pedestrian walking along a hallway, holding the phone upfront (cf. figure 2.6), resampled to $100\,\text{Hz}$.

Based on the step frequency results from figure 2.7 and [Sau+11], the low-pass filters were configured with a cut-off frequency of $3.0\,\text{Hz}$, also covering pedestrians who walk faster than the average $\approx 1.8\,\text{Hz}$. The IIR's $Q \approx 0.7$ was chosen to resemble a Butterworth filter (see e.g. [Smi99]). For the FIR low-pass, the kernel's size $N = 29$ was selected to provide results similar to the IIR. To also remove the DC component for these two filters, a constant $9.81\,\text{m/s}^2$ was subtracted from the magnitude beforehand. As can be seen in figure 6.7, until $5\,\text{Hz}$, the attenuation provided by IIR/FIR is almost identical. Hereafter the FIR provides a much better damping, at the expense of increased delays and required computational power.

The two band-pass filters were configured similarly, using $2\,\text{Hz}$ as center frequency. For the FIR, a width of $\pm 1\,\text{Hz}$ was chosen, to match with the low-pass filters. The corresponding kernel
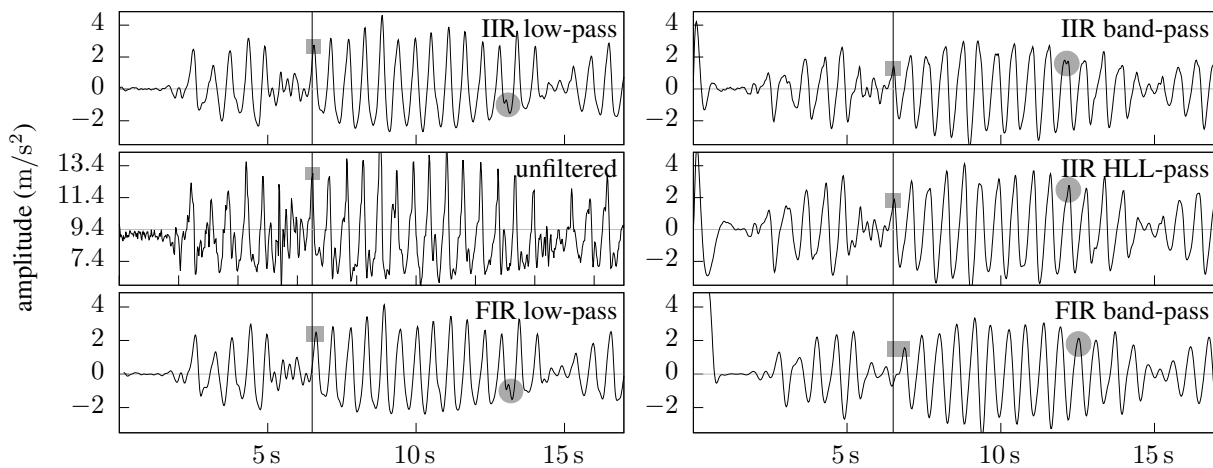
Figure 6.8: Impact of applying the filters from table 6.3 to the accelerometer's magnitude. The vertical line and a rectangle visually indicate each filter's delay. All band-pass-like setups suffer from an initial peak, resulting from the constant gravity contained within the signal. While the FIR band-pass provides the smoothest result, its delay is clearly notable. Circles denote issues, explained within the text.

required $N = 101$ entries, to correctly remove the DC component from the accelerometer's magnitude. With the sample rate of $100\,\mathrm{Hz}$, this causes a delay of $\approx 500\,\mathrm{ms}$. The IIR's $Q = 1.5$ was configured to match the $\pm 1\,\mathrm{Hz}$ bandwidth of the FIR. Due to the large kernel size, the FIR provides a good attenuation, and fast cut-offs. The IIR shows a viable cut-off only for the lower frequencies, with the attenuation of higher ones being less pronounced (cf. figure 6.7). While this can be mitigated by further increasing Q, it also reduces the width of the passband, allowing only for walks at a well known step frequency, affecting the filter's general purpose.

This is addressed by chaining three IIR filters to approximate the FIR band-pass: A high-pass removes frequencies below $1\,\mathrm{Hz}$, and *two* low-pass filters attenuate everything above $3\,\mathrm{Hz}$. This combination, referred to as HLL-pass, creates a filter similar to a $(2 \pm 1)\,\mathrm{Hz}$ band-pass, and is stronger than a single IIR. Shown in figure 6.7, this compromise between the IIR and FIR band-pass provides more control on passband and cut-offs, while retaining viable delay times.

Figure 6.8 shows the results when applying these filters to the magnitude of accelerometer readings for a normally paced walk, sampled at $100\,\mathrm{Hz}$. While peaks are distinguishable for all shown contestants, the FIR band-pass provides the smoothest result. However, as can be seen by comparing the vertical lines, both FIR filters, especially the band-pass with its larger kernel, introduce a significant delay. All three band-pass-like setups need some time to recognize and remove the gravity as DC component from the signal, rendering the first few seconds of output unusable. This can be addressed by pre-feeding the filters with several constant readings of $\approx 9.81\,\mathrm{m/s^2}$. Another aspect that divides low-pass and band-pass is the positive/negative peak ratio. As mentioned in section 2.4.1, the step pattern usually consists of one positive, and one negative peak, where the positive one is slightly more pronounced. This ratio is kept by the low-
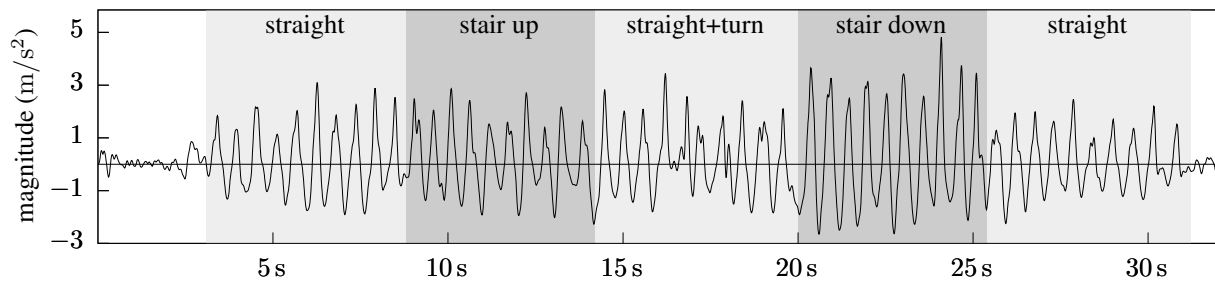
Figure 6.9: Magnitude of the accelerometer, filtered by the IIR low-pass from table 6.3. The device was held upfront while walking 10 steps along a hallway, climbing a stair with 10 treads, making 10 steps including a $180°$ turn, hereafter walking the same way back again.

Name	Input	Filtering	Detection Scheme	Name	Filtering
SD1	magnitude	-	peak + dead time		
SD2a	magnitude	IIR low-pass	local maxima	SD2b	IIR HLL-pass
SD3a	magnitude	IIR low-pass	zero crossing	SD3b	IIR HLL-pass
SD4a	tilt compensated z	IIR low-pass	zero crossing	SD4b	IIR HLL-pass

Table 6.4: Examined step detectors and their configuration according to section 2.4.1.

pass filters, but removed by the band-passes, as it denotes a low frequency. When the desired step detection method relies on this ratio, band-pass filters can not be used.

Despite their abilities, FIR filters are overstated for the step detection problem. Due to both, delay and computational requirements, the following experiments omit the FIR filters, focusing only on IIR versions. For the shown example, the IIR HLL-pass performed slightly better than the IIR band-pass, at a similar delay. However, for faster walks (see figure 6.10), the latter fails, due to aforementioned limitations, while the HLL-pass yields viable results. Being limited to a narrow range of walking speeds, the IIR band-pass is omitted as well. Actual step-detection is performed using the IIR low-pass, the IIR HLL-pass, and no filtering for comparison.

Figure 6.9 depicts the accelerometer's magnitude for a walk of 50 steps in length, including a stair, and one $180°$ turn, filtered by the IIR low-pass from table 6.3. During the turn, at around $17\,\mathrm{s}$, several zero crossings appear within the filtered signal, causing potential issues, depending on the chosen pattern detector. While the magnitude's fluctuation is approximately constant throughout the walk, the steps at $17\,\mathrm{s}$, $26\,\mathrm{s}$ and $28\,\mathrm{s}$ are barely pronounced. In contrast, walking downstairs, between $20\,\mathrm{s}$ to $25\,\mathrm{s}$, peaks increase in both directions. Besides this diversified walk, a simple 50-steps-straight setup is examined as well. To estimate the quality of the step pattern detection algorithms discussed in section 2.4.1, both walks were conducted several times, by multiple persons with varying footwear, using different devices. Table 6.4 lists all examined contestants, including their detection and filtering strategy. Within recordings, the step
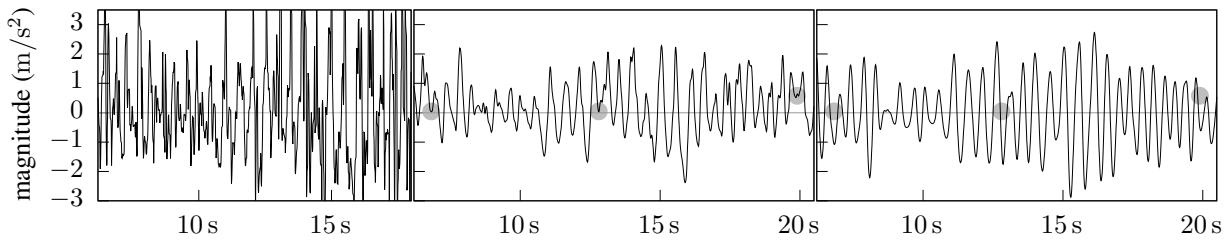
Figure 6.10: Accelerometer's magnitude for a fast paced walk. The raw data (left) contains significant amounts of noise, and does not allow for a robust step detection. Applying the IIR low-pass (middle) yields a noticeable improvement, but indicates several issues (filled circles), due to the present low and high frequency noise. Using the HLL-pass instead (right), enhances the overall result.

Dataset	Walks	SD1	SD2a	SD2b	SD3a	SD3b	SD4a	SD4b
straight only	22	50.0	51.7	51.8	50.0	50.6	50.0	50.6
straight only, fast	12	49.8	52.5	50.6	49.9	50.6	49.8	50.7
typical mix	25	47.1	48.9	52.8	49.4	50.1	49.1	49.9
typical mix, fast	20	40.0	41.2	48.5	44.8	48.8	43.9	48.5
overall	79	46.5	48.3	51.1	48.5	50.0	48.2	49.9

Table 6.5: Number of detected steps for all detectors presented in table 6.4, averaged by dataset. The correct number of steps is 50 for all cases.

frequency resided somewhere around $\approx 1.8\,\text{Hz}$. For this input, all strategies provided viable results. Even when no filtering was used (SD1), deviations were only marginal.

Additionally, faster walking speeds were examined as well. When increasing the step frequency beyond $2.2\,\text{Hz}$, most detection schemes started to fail, missing up to $15\,\%$ of the taken steps. While the IIR low-pass used within SD2a-SD4a mitigated this problem, the overall number of detected steps remained too low. Analyzing the raw accelerometer magnitude for those walks (see figure 6.10) reveals that the amount of noise increased significantly for both, high and low frequencies. The low-pass filtered signal (middle) thus fluctuates at a very low frequency, removing some zero crossings. An IIR band-pass (not shown in the figure) provided slightly better results and reduced the number of missed steps, still with room for improvements, due to its slow cut-off. The IIR HLL-pass addresses this issue with its two chained low-pass filters, and one high-pass, effectively removing all noise components from the raw signal (right).

Table 6.5 contains detailed results for all presented detectors and datasets. For normally paced walks, all detectors provided viable results. As expected, the straight walks yielded the best overall results. When variations, like stairs, are introduced, the risk for missing steps during the detection process increases. For highly paced walks, the HLL-pass, used in SD2b, SD3b and SD4b, provides better detection rates than the single low-pass used in other setups.
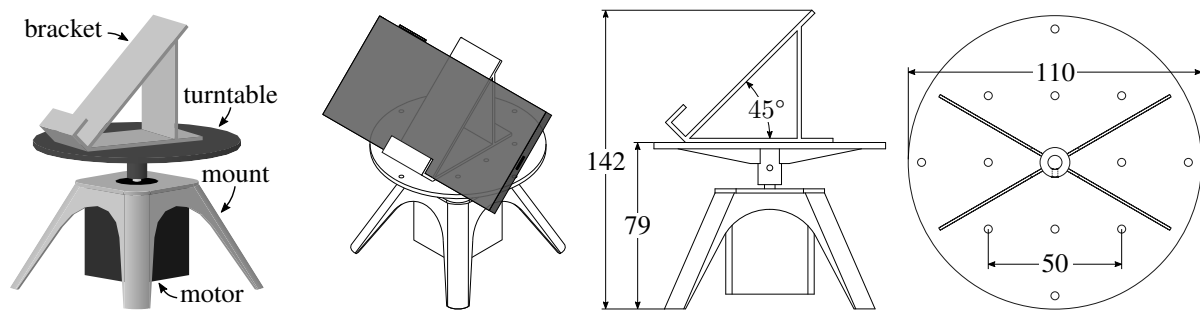
Figure 6.11: Layout of the turntable used for synthetic IMU tests. The direct-drive stepper motor ensures a valuable ground truth. The bracket is used to place the smartphone into portrait and landscape mode, with a constant $45°$ tilt, required for tilt estimation tests. All distance units are given in mm.

6.2.3   Relative and Absolute Heading Estimation

The suitability of a smartphone's IMU for turn-detection with the gyroscope, or absolute heading estimation by the eCompass, is examined synthetically, using a 3D-printed turntable (cf. figure 6.11). It is driven by a 17MH5417 stepper motor, with a physical resolution of $0.9\,°/\mathrm{step}$, and executes requested rotations with accuracy and precision down to $1°$. The resolution is further increased by a TMC2100 stepper driver, configured for $1/16$ microstep interpolation, with a full $360°$ turn thus requiring 6400 steps. To ensure that the motor does not skip any steps, and the smartphone keeps its enforced alignment, an acceleration and deceleration phase is used.

Tests examine three aspects, under controlled conditions: estimating accuracy and drift of the gyroscope, testing the magnetometers suitability for absolute heading estimation, and determining whether tilt compensation works as expected. Thus, experiments used three different device orientations: The first places the phone flat onto the turntable, with its z-axis pointing upwards, and a rubber mat in between ensuring the device remains in place. To examine tilt compensation, the phone is also placed in portrait and landscape mode, both with a $45°$ tilt angle, similar to a pedestrian holding the device upfront, fixed by a bracket shown in figure 6.11.

While the turntable is aligned parallel to the earth's surface, a phone placed on top can not be ensured to read an exact $0°$ angle, due to slight misalignments. All devices were manually aligned, ensuring that accelerometer readings match $0°$ as closely as possible. Angular values shown within the following experiments will thus vary slightly.

Results - Overview   To determine the quality of tilt compensation and turn-detection, all three poses were examined using three different turning rates: $62.7\,°/\mathrm{s}$, $122.5\,°/\mathrm{s}$ and $234.0\,°/\mathrm{s}$. Table 6.6 shows the results for measuring this turning rate, by using the median of the gyroscope's magnitude (see (2.38)), and the results for the placement angle, determined by the average dot product between normalized accelerometer readings and $(0, 0, 1)^T$ (see (2.46)).

	0° flat			landscape at 45°			portrait at 45°		
	62.7 °/s	122.5 °/s	234.0 °/s	62.7 °/s	122.5 °/s	234.0 °/s	62.7 °/s	122.5 °/s	234.0 °/s
S3 Mini	1.7°	1.4°	1.7°	45.8°	46.2°	47.5°	45.0°	44.7°	45.1°
	57.9 °/s	112.9 °/s	215.5 °/s	57.9 °/s	112.9 °/s	215.2 °/s	58.4 °/s	114.1 °/s	217.8 °/s
Nexus 6	0.9°	0.9°	1.6°	47.3°	47.1°	48.0°	46.5°	46.2°	45.3°
	62.4 °/s	121.9 °/s	232.7 °/s	62.2 °/s	121.4 °/s	231.9 °/s	62.0 °/s	121.4 °/s	232.1 °/s
Moto Z	1.5°	2.0°	3.4°	44.9°	44.9°	44.6°	45.1°	44.8°	43.5°
	62.6 °/s	122.2 °/s	233.4 °/s	63.0 °/s	123.1 °/s	234.8 °/s	62.0 °/s	121.5 °/s	233.0 °/s
LG G6	1.3°	1.5°	3.2°	45.7°	46.3°	48.1°	45.4°	45.5°	45.8°
	63.0 °/s	123.2 °/s	235.3 °/s	63.2 °/s	123.2 °/s	235.0 °/s	62.5 °/s	122.8 °/s	235.7 °/s
Pixel 2	1.9°	1.5°	3.1°	45.3°	45.9°	47.7°	45.9°	45.6°	45.8°
	62.6 °/s	122.3 °/s	233.6 °/s	62.6 °/s	122.3 °/s	233.6 °/s	63.7 °/s	124.5 °/s	237.7 °/s
KEY2	0.9°	1.4°	2.9°	45.4°	45.5°	45.9°	44.9°	44.2°	42.2°
	62.6 °/s	122.5 °/s	233.8 °/s	62.6 °/s	122.1 °/s	233.0 °/s	62.9 °/s	123.0 °/s	235.6 °/s

Table 6.6: Results of a turntable experiment, using three different device placements, each with three turning speeds. For every phone, the first row shows the tilt angle, estimated by the average dot product between $(0, 0, 1)^T$ and the normalized readings from the accelerometer (see (2.46)). The second row contains the measured angular velocity, estimated by the median of the gyroscope's magnitude.

For all devices, except the S3 Mini, turn rates are viable, and, due to using the magnitude, independent of the phone's orientation. The S3 Mini is one of the first containing an IMU with gyroscope (see table 6.1). Besides comparatively poor sensor resolution, neither its accelerometer nor its gyroscope seemed to be correctly calibrated. Within all tests, the accelerometer's z-axis was off by a constant $+0.13\,\mathrm{g}$, and the gyroscope's x and z were off by $+0.015\,\mathrm{rad/s}$ and $+0.005\,\mathrm{rad/s}$, respectively. While the latter seems small, it denotes a drift of $17.19\,°/\mathrm{min}$, significantly affecting turn-detection. The same holds true for tilt compensation. Depending on the way this device is held, the tilt-vector deviates, due to the constant z-offset. While the accelerometer of the Milestone 2 behaved similarly, all other examined smartphones did not show major offsets for their accelerometers or gyroscopes. Yet, for all contestants, the estimated angle indicated a dependency on the current turning rate, examined within the following.

Results - Pose Estimation Discussed in section 2.4.2 and 2.4.3, readings from gyroscope and magnetometer are orientation dependent. The following examines the required tilt compensation, using the turntable as controllable testbed, before evaluating actual pedestrian walks.

Figure 6.12 shows a setup, with the LG G6 placed flat onto the turntable, examining the influence of the rotation axis, by moving it into six different positions: Centered within the turntable where the rotation axis matches the *phone's* center, aligned with the left or right boundary of the turntable, moved to the top or bottom, and with the rotation axis matching the actual location of the *IMU*. This location, shown for the LG G6, is model dependent, was determined by hardware teardown, and confirmed by experiments. The turntable was configured to perform ten turns counterclockwise (CCW), a short break, and ten turns clockwise (CW). The ten turns are
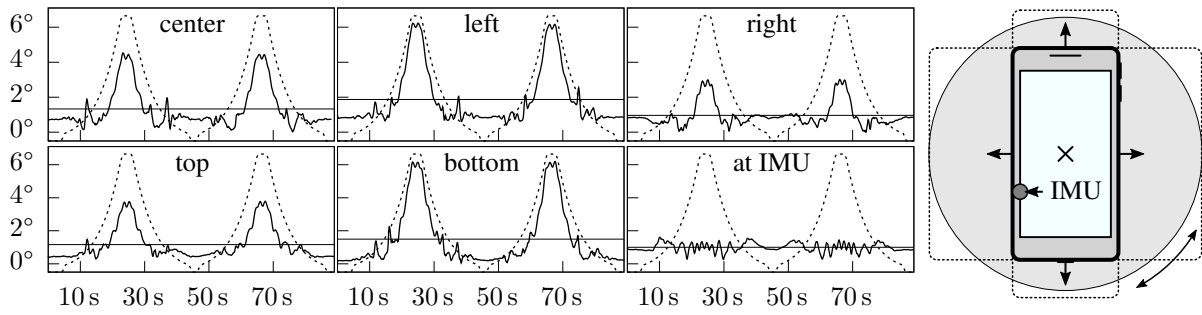
Figure 6.12: Behavior of accelerometer and gyroscope readings from the LG G6, based on the location of the rotational axis, and rotation speed. The estimated angle between the z-axis and the readings from the accelerometer (solid lines) should be $\approx 0°$, as the phone is aligned parallel to the earth's surface. Its average is indicated as horizontal line. The absolute turn-rate measured by the gyroscope's z-component (dashed lines) is equal for all six variants, showing the same output with a maximum of $\approx 234°/\text{s}$.

divided into three sections: four turns of acceleration, spinning two turns with a target speed of $234.0°/\text{s}$, followed by four turns of deceleration. With the phone placed flat onto the table, the angular velocity is measurable solely within the gyroscope's z-component, depicted as dashed line in figure 6.12. For visualization reasons, the absolute of this value is depicted, yielding two positive turn-rate-peaks. Both, accelerometer and gyroscope readings were filtered using a $1000\,\text{ms}$ moving average filter, to remove spikes, visualizing only the important aspects. As can be seen, the turn-rate in z is unaffected by the position of the phone. For the accelerometer, results are different. The impact of the rotation axis is determined by calculating the angle between $(0, 0, 1)^T$, and the readings from the accelerometer (see (2.46)). On a first impression, this angle should remain stable while turning. Yet, this only holds true if the IMU is located directly within the rotation axis. If not, readings are influenced by a centripetal force, depending on the current angular velocity, and the IMU's distance towards the rotation axis. This is confirmed by the plots shown in figure 6.12. For the *left* and *bottom* variant, the LG G6's IMU is moved farther away from the rotation axis, increasing the centripetal force, thus the accelerometer's readings for x and y, and thus the angle with respect to the z-axis. For *right* and *top*, the IMU is nearer to the rotation axis, reducing the centripetal force, and thus the angle.

This result is important, as the additional force, besides gravity, affects the tilt compensation, and thus turn or heading estimations. Depending on the angular velocity, faster turns will cause larger errors. As the location of the IMU depends on the device, effects vary between models, explaining the results in table 6.7, where the error between actual and estimated angle increased differently for faster turning rates, dependent on the current orientation. When the phone is held by a pedestrian upfront, this effect intensifies, as the distance towards the rotation axis of the turning pedestrian increases, thus affecting the tilt compensation.
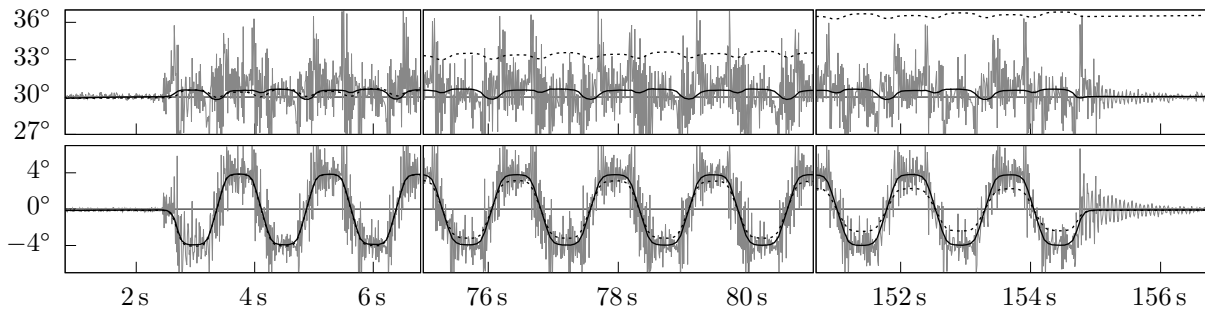
Figure 6.13: Consecutive $\pm 15°$ CCW and CW rotation of a turntable tilted by $15°$, with the phone in $45°$ portrait mode. The two rows denote the tilt angles in x (top) and y (bottom), estimated from the accelerometer (gray), gyroscope (dashed black) and complementary filter with $\kappa = 0.998$ (solid black). By combining both, the filter compensates noise from the accelerometer, and drift of the gyroscope.

The complementary filter (cf. section 2.4.2) provides a partial solution. By applying a low-pass filter to the accelerometer, temporal anomalies are compensated. Yet, this only addresses small timeframes, unsuitable for longer turns, or the synthetic tests on the turntable. While the complementary filter can be configured to rely even more on the gyroscope, when increasing the timeframe of compensation, drift becomes notable, causing new issues. These effects, and the impact of the filter, are shown in figure 6.13, where the LG G6 was placed in $45°$ portrait mode, onto the turntable, which was tilted backwards by $15°$, causing a $30°$ angle between phone and ground. The turntable was configured to perform $\pm 15°$ turns around this initial position, that is, $15°$ CCW once, followed by a repetition of $30°$ CW and $30°$ CCW, alternating around the initial center. This results in a shaking movement, similar to a walking pedestrian, holding the smartphone upfront. Figure 6.13 shows the estimated rotation for the phone, around its x and y axis (see (2.39) and (2.49)). As the gyroscope measures only relative changes, its cumulation was initialized using the first few estimations from the accelerometer. As can be seen, the accelerometer provides an absolute estimation, but contains significant amounts of noise. While the gyroscope does not show notable noise, it suffers from cumulating drifts, with the $30°$ angle reaching $36°$ after $\approx 130\,\mathrm{s}$. This is addressed by a complementary filter with an empiric $\kappa = 0.998$, combining accelerometer and gyroscope, deriving a stable, noise-free estimation, suited for tilt compensation. While results were depicted and discussed for a single phone, all other devices showed similar behavior, yet, dependent on the location of their IMU.

Results - Turn Angles The quality of gyroscope-based turn-detection is examined by the turntable performing ten turns CCW, followed by ten turns CW, using the three mentioned speeds and orientations. Ground truth is the cumulated angle after the first ten, and all 20 turns, which should read $3600°$ and $0°$, respectively. It is estimated using four different strategies: Integrating the gyroscope's z-component (2.36), using the gyroscope's signed magnitude (2.38),

performing tilt compensation solely by the accelerometer using (2.49) with $\kappa = 0$, and, applying tilt compensation by the complementary filter using (2.49) with an empiric $\kappa = 0.998$.

Results are listed in table 6.7. As expected, just integrating the gyroscope's z-component (each 1st row) provides valid results only when the smartphone is placed parallel to the earth's surface. The $45°$ landscape and portrait modes should read approximately $\sin(45°)\,3600° \approx 2546°$. While using the gyroscope's signed magnitude (each 2nd row) addresses this issue, this is only valid for the presented turntable tests, where the induced rotation has a single source axis. When a pedestrian holds the smartphone while walking, shaking additionally affects the magnitude, invalidating the cumulated angle. For pedestrian walks, tilt compensation (each 3rd and 4th row) should be applied to the gyroscope measurements, prior to the integration step. While the accelerometer is able to provide this compensation on its own (each 3rd row) previous experiments indicated that using a complementary filter (each 4th row) is advisable.

This is again confirmed by the values of the $45°$ portrait setup in table 6.7. When the bracket holds the smartphone in portrait mode (see figure 6.11), each device picks up a notable amount of noise, resulting from a disadvantageous center of mass. When using solely the accelerometer for tilt compensation (3rd row), this noise affects the estimated tilt, and thus the resulting cumulated angle, clearly indicated by the divergent results. The complementary filter is able to address this issue by filtering the accelerometer, and including the gyroscope.

Counterintuitively, the KEY2 and Moto Z show larger errors for the flat placement, than for the other two orientations. Analyzing the raw readings from the gyroscope indicated that the sensor within these devices provides slightly divergent turning rates for CCW and CW rotation, at least, for readings of the z-axis. Even though this difference is marginally, it cumulates over time, causing notable deviations after the 20 turns. As it is less pronounced for faster turning rates, that is, shorter test-runs, it might be caused by a minor, constant calibration offset.

While repeatability is good for most devices, all absolute readings are slightly off. That is, the measured turn-rate (see table 6.6) does not match with the actual one, yielding the cumulations (see table 6.7) to be off as well. Especially the Pixel 2 in portrait mode deviates more than $1\,\%$ from the actual turn-rate. Yet, for all examined devices, results were viable with relatively small errors for shorter timeframes, well suited for assisting coarse absolute headings.

Results - ECompass   Even though the results for gyroscope-based turn-detection are promising, they remain relative. When not knowing the pedestrian's initial heading, and/or suffering from cumulating drift, absolute indications are helpful, even if they are less accurate.

To ensure the magnetometer points towards geomagnetic north, a brief test compared the estimated angle (2.60) with the one from an analog compass, when placing each device flat onto the ground. Being inexact, results are not provided as actual angular values. Nevertheless,

	0° flat			landscape at 45°			portrait at 45°		
	62.7°/s	122.5°/s	234.0°/s	62.7°/s	122.5°/s	234.0°/s	62.7°/s	122.5°/s	234.0°/s
S3Mini	3337.0 / 37.9	3328.4 / 21.1	3322.5 / 10.8	2114.2 / 36.5	2103.5 / 15.8	2102.5 / 11.4	2193.6 / 33.8	2196.2 / 15.0	2191.7 / 11.5
	3347.0 / 44.3	3335.1 / 27.0	3327.6 / 15.1	3378.5 / 107.9	3350.3 / 56.0	3339.5 / 33.8	3372.2 / 33.5	3369.2 / 18.3	3362.1 / 14.6
	3334.4 / 35.1	3327.7 / 20.1	3322.6 / 10.4	3358.0 / 103.5	3339.3 / 55.1	3334.4 / 33.3	3355.8 / 30.7	3345.4 / 7.9	3348.1 / 9.5
	3335.9 / 36.0	3327.9 / 20.0	3322.0 / 10.2	3361.4 / 103.9	3339.3 / 55.5	3331.4 / 32.2	3358.8 / 30.4	3350.0 / 8.1	3350.9 / 10.1
Nexus 6	3579.7 / -4.5	3582.9 / -1.4	3579.9 / -1.9	2437.1 / 10.9	2450.9 / -0.5	2450.1 / -0.4	2466.5 / -2.6	2469.3 / -3.7	2473.6 / -1.3
	3579.9 / -4.8	3583.2 / -1.4	3580.1 / -1.9	3569.6 / 0.9	3569.6 / 0.8	3568.5 / 0.0	3583.1 / -2.6	3569.7 / -2.1	3571.6 / -1.5
	3579.5 / -4.5	3582.6 / -1.4	3578.3 / -0.9	3567.2 / 4.7	3568.4 / 0.4	3565.9 / 0.2	3557.1 / -11.4	3566.1 / -3.8	3569.0 / -1.4
	3579.6 / -4.6	3582.7 / -1.4	3579.4 / -1.7	3568.7 / 2.6	3568.8 / 0.6	3566.9 / -0.2	3569.7 / 0.3	3568.5 / -1.9	3570.7 / -0.4
Moto Z	3598.8 / 14.1	3594.7 / 5.8	3593.2 / 3.1	2531.7 / 11.4	2530.3 / 7.7	2529.2 / 4.5	2509.3 / 9.0	2507.9 / 4.9	2507.6 / 2.4
	3599.6 / 14.4	3595.4 / 6.0	3594.0 / 3.3	3619.1 / 8.0	3617.7 / 5.9	3615.8 / 3.3	3599.8 / 12.9	3591.1 / 5.7	3587.6 / 2.6
	3598.8 / 14.2	3593.7 / 5.7	3586.8 / 3.3	3615.7 / 6.9	3614.2 / 4.7	3610.3 / 1.8	3543.4 / -31.8	3555.5 / -13.7	3573.2 / 0.5
	3599.0 / 14.2	3594.4 / 5.8	3589.9 / 3.9	3617.1 / 7.4	3615.4 / 5.0	3613.4 / 2.8	3590.3 / 10.1	3587.6 / 5.3	3581.6 / 3.7
LG G6	3619.7 / -0.0	3619.8 / -0.3	3620.0 / -0.1	2557.3 / 0.2	2558.6 / -0.2	2557.3 / -0.1	2553.9 / -0.7	2554.8 / -0.4	2555.9 / -0.2
	3619.9 / -0.1	3619.9 / -0.3	3620.1 / -0.2	3617.7 / 0.1	3617.1 / -0.2	3617.1 / -0.5	3633.5 / -0.3	3626.6 / -1.2	3625.7 / -0.3
	3618.5 / -0.3	3618.1 / -0.1	3609.4 / -0.6	3611.8 / -0.8	3612.3 / -0.5	3602.0 / -0.9	3567.1 / -40.4	3593.0 / -9.1	3612.5 / -2.2
	3619.5 / -0.0	3619.2 / -0.3	3612.4 / 0.1	3616.1 / 0.1	3615.2 / -0.2	3606.5 / 0.3	3624.0 / -0.7	3624.2 / -0.4	3619.9 / -0.0
Pixel 2	3594.4 / -0.2	3594.5 / -0.1	3594.1 / -0.1	2525.5 / -0.8	2527.2 / -0.6	2528.2 / 0.2	2527.9 / -0.6	2529.8 / 0.0	2530.9 / -0.3
	3594.6 / -0.2	3594.6 / -0.1	3594.2 / -0.0	3595.0 / -1.2	3593.9 / -1.6	3594.1 / -1.0	3665.5 / -5.9	3657.4 / -0.8	3657.0 / -1.2
	3591.6 / 0.1	3592.7 / 0.3	3584.6 / 0.3	3590.1 / -2.4	3591.8 / -1.5	3583.1 / 0.2	3622.5 / -2.4	3644.8 / -0.9	3644.2 / 1.9
	3594.2 / -0.2	3593.6 / -0.1	3586.4 / 0.2	3593.5 / -1.2	3592.9 / -1.6	3586.8 / -0.5	3655.2 / -1.1	3654.4 / -0.3	3648.1 / 0.0
KEY2	3590.6 / -11.6	3593.4 / -6.0	3595.1 / -3.4	2503.7 / -0.1	2504.8 / 0.5	2504.7 / -0.0	2532.7 / -0.7	2525.4 / -0.5	2526.4 / -0.4
	3591.1 / -12.1	3594.0 / -6.4	3595.7 / -3.7	3591.8 / 9.9	3589.2 / 6.0	3587.4 / 2.0	3631.7 / -10.7	3623.6 / -5.0	3624.3 / -2.7
	3590.3 / -11.0	3590.8 / -6.0	3582.7 / -3.7	3587.7 / 11.4	3586.7 / 7.2	3577.4 / 3.6	3595.0 / -22.4	3617.5 / -5.8	3609.0 / -3.2
	3590.7 / -11.4	3592.3 / -5.9	3587.4 / -1.2	3591.4 / 11.2	3588.2 / 7.1	3580.3 / 4.6	3619.7 / -10.3	3620.9 / -4.8	3615.9 / 1.3

Table 6.7: Results of the turntable experiments, performing ten CCW turns, followed by ten CW turns, using three different smartphone placements, each with three turning speeds. For every phone, the cumulated heading change after the first ten CW turns (left value, should be 3600°) and after additional ten CW turns (right value, should be 0°) is determined using four different strategies: Integrating the gyroscope's z-component as-is (2.36) (1st row), using the gyroscope's signed magnitude (2.38) (2nd row), performing tilt compensation solely by the accelerometer using (2.49) with $\kappa = 0$ (3rd row), and applying tilt compensation by the complementary filter using (2.49) with $\kappa = 0.998$ (4th row).
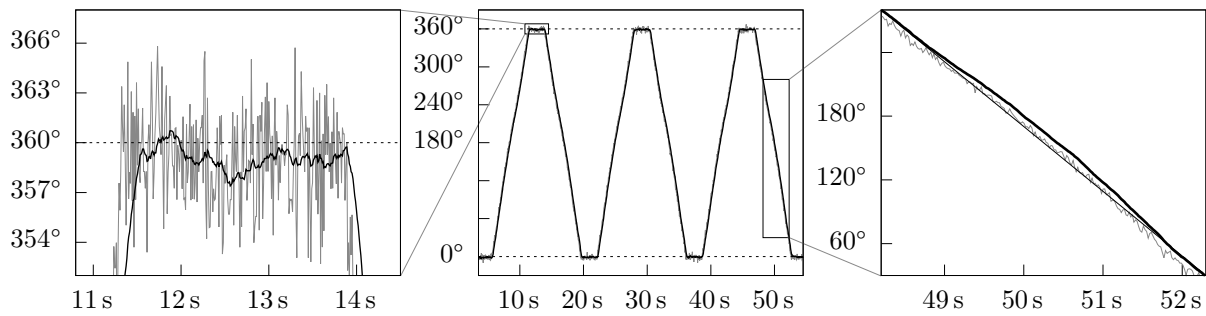
Figure 6.14: Angle estimated by (2.60) for the `LG G6`, after offset-removal and unwrapping for visualization reasons. The gray line denotes the angle derived from the raw sensor data, the black line resulted from prior low-pass filtering. Shown in the left, the unfiltered result contains significant amounts of noise, mitigated by the filter. The right denotes a minor nonlinearity throughout the rotation.

all examined errors remained within $\pm 20°$, and the majority within $\pm 10°$. While not ideal, this is still suited for a probabilistic absolute heading, backed by a fine relative turn-detection.

Potential accuracy, precision, and lag were examined by placing each device flat onto the turntable, using a spacer of approximately $250\,\mathrm{mm}$ in between, to prevent the motor from influencing sensor readings. The turntable's pattern is changed to three repetitions of, $360°$ CCW followed by $360°$ CW, matching the value range of the magnetometer. Acceleration and deceleration were disabled, to visualize the linearity of the sensor. However, this limits testing to the slowest speed of $62.7\,°/\mathrm{s}$, ensuring the phone stays in place, and the motor not skipping steps. To allow comparing against a known ground truth, all angles calculated from magnetometer readings are adjusted to be relative to the initial one, eliminating offsets towards magnetic north. The result is unwrapped to remove jumps, e.g. between $0°$ and $360°$, allowing for continuous drawing in figures. Additionally, a $250\,\mathrm{ms}$ moving average filter is applied to each of the three magnetometer axes, before estimating the angle (2.60). Filtered and raw results are shown in figure 6.14. As can be seen, without filtering, a significant amount of noise is present within the estimated angle (left). Independent of filtering, the behavior while turning indicates a minor nonlinearity, deviating from a straight line (right), observable for all tested devices. While sources for this behavior are numerous, it is most likely related to minor calibration issues.

Mentioned earlier, the magnetometer is affected by hard iron and soft iron effects, requiring (re)-calibration. For all examined devices (see table 6.1) except the `Milestone 4`, the magnetometer calibrated itself when being active, and the smartphone rotated several times around all three axes. The quality of this calibration is examined by fitting an ellipse onto the (x, y) readings, which, for the conducted test, should form a circle around $(0, 0)$. The fitting is performed by solving an over-determined set of linear equations, determining six ellipse-parameters to match with all measured samples, based on the general ellipse equation $ax^2 + bxy + cy^2 + dx + ey + f = 0$. It is hereafter converted to the *geometric form*, with a
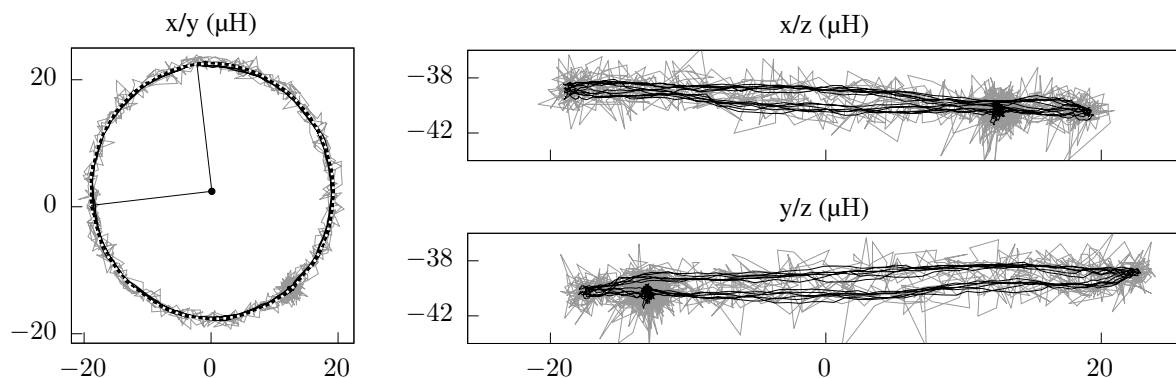
Figure 6.15: Raw (gray) and filtered (black) magnetometer readings for the LG G6, that lead to figure 6.14. The fitted ellipse (dotted white) indicates an offset from $(0, 0)$, including a minor unroundness. Similarly, both plots on the right reveal a minor sensor-tilt, affecting the roundness as well.

center, length of major/minor axis, and rotation [WGN15; Ber17]. The result for the LG G6 is shown in figure 6.15, containing the raw and filtered readings for all three magnetometer axes. As can be seen, the fitted ellipse is slightly off-center, and shows a minor unroundness, affecting the linearity, as depicted earlier in figure 6.14. While this can be addressed using the presented calibration (2.62), differences in accuracy were negligible for most devices.

Table 6.8 presents an average out of several repetitions, for each examined smartphone. The quality of automatic calibration is given by the average magnitude of the magnetometer readings (1st column), and the center of the estimated ellipse (2nd and 3rd column), which should be $(0, 0)$. While not being ideal, automatic calibration is sufficient for absolute heading estimation. Only for the Milestone 4, the calibration was performed manually using (2.62). Besides the center of the ellipse, the ratio between the size of its minor and major axis (see figure 6.15) is calculated (4th column), indicating the *roundness* of the sensor readings, which should be 1.0 for an ideal circle. While some of the devices show a tendency towards an ellipsoid, their sensor readings are still viable for heading estimations. Sensor-tilt (5th column) also affects the roundness and angular results. The listed value denotes the angle between the x-axis, and a line defined by the range of the magnetometer's z-readings. For the presented turntable pattern, the latter should also yield a horizontal line, and thus an angle of $0°$ (see figure 6.15 right). While some devices indicated a slight tilt, results remained within a viable range, with no significant impact on the estimated angles.

The right half of table 6.8 shows the angles estimated in between the turntable's actions. They are based on the same values, as have been used for figure 6.14. That is, filtered, offset-corrected, and unwrapped. Entries are derived by searching for a minimum/maximum at each of the $0°/360°$ plateaus. As can be seen, while the angles do not exactly conform with an alternating pattern of $0°$ and $360°$, they are well within range, even for older smartphone models.

	Magnitude/Center (µH)			Roundness		Angular Values (°)						
	mag	x	y	ratio	tilt (°)							
Milestone 2	46.41	0.54	2.13	0.98	0.93	-2.3	363.5	-2.3	362.6	-0.9	361.9	-2.4
Milestone 4	-	-	-	0.95	0.24	-2.0	360.4	-1.2	360.0	-1.6	360.1	-2.2
S3 Mini	42.63	0.37	-0.68	0.92	2.31	-2.0	359.8	-2.5	359.5	-3.0	357.9	-3.6
Nexus 6	44.46	0.62	0.58	0.99	0.37	-1.4	361.5	-0.7	361.5	-0.8	361.0	-0.7
S5 Neo	42.38	1.84	3.07	0.97	0.90	-0.3	361.0	-0.6	360.0	-2.1	360.4	0.5
Moto Z	45.88	0.10	-1.87	0.98	1.74	-2.2	360.7	-3.5	361.3	-0.9	361.9	-2.1
LG G6	44.14	0.12	2.44	0.95	1.43	-2.1	362.9	-0.0	362.1	-0.1	362.2	0.9
Pixel 2	43.90	1.07	-1.13	0.96	0.61	-1.0	360.8	-0.8	360.6	-0.8	360.5	-0.7
KEY2	46.19	-0.54	2.33	0.94	0.65	-0.6	362.3	-0.9	361.1	0.1	361.6	-1.2

Table 6.8: Results of the turntable experiments, performing three times a sequence of: $360°$ CCW and $360°$ CW at $62.7°/s$, with the smartphone placed flat onto the turntable. Besides the seven estimated angles (right), calibration-related metrics are shown (left), covering the average magnitude, offset from $(0,0)$, and the result's roundness, explained within the text.

Tests requiring tilt compensation are presented only briefly, for two reasons. On the one hand, the turntable setup is not suited for a $45°$ portrait/landscape alignment with enough space between the device and the motor, without introducing misalignment and shaking. On the other hand, previous results for the gyroscope have already shown the validity of the tilt compensation. Yet, examinations conducted with additional spacing indicated results similar to the ones from the gyroscope, with tilt compensation working as expected.

Not shown within figures, the angle estimated from the magnetometer was also compared against the one determined from the gyroscope, indicating no significant delay between both, when using the raw magnetometer data. Even for devices where the sensor seems to perform internal filtering (S3 Mini, S5 Neo and Moto Z), lag was moderate, as low as a few milliseconds. Especially for the S3 Mini, the magnetometer was significantly more stable than its gyroscope, which suffered from aforementioned drifts. Thus, the conducted synthetic tests indicate that absolute heading from the magnetometer is suited for supporting the gyroscope.

6.2.4 Pedestrian Dead Reckoning

The actual quality of IMU-based turn-detection, and absolute heading estimation is determined by a pedestrian walking while holding the smartphone upfront. For an impression of the estimation results, the previously examined step-detection is included as well, yielding a pedestrian dead reckoning setup, depicting individual aspects for each of the two heading methods. It also provides an impression on the capabilities of smartphone-based PDR. That is, expected localization quality, when not using sensors for absolute location estimations, omitting probabilistic approaches, recursive density estimation, and constraints from an underlying floorplan.
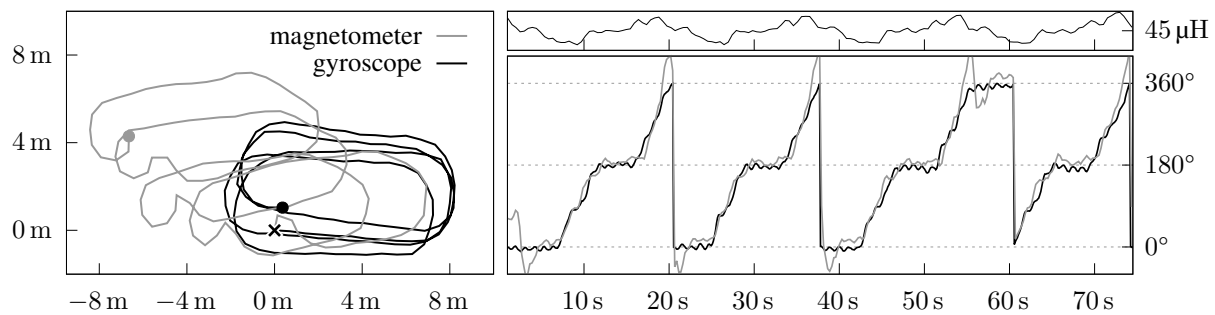
Figure 6.16: PDR for walking four times around a $7.5 \times 2.5\,\text{m}$ railing, consisting of wood and metal, holding the `Pixel 2` upfront. Even though their heading estimations (lower right) are often similar, the estimated path (left) is significantly different between using the cumulated gyroscope (black), and using the magnetometer (grey) as heading indicator. The impact of the environment is also indicated by the behavior of the magnetometer's magnitude (upper right, showing $(45 \pm 20)\,\mu\text{H}$).

The walking path is estimated by starting at $(0, 0)$, hereafter adjusting this location whenever a step is detected, moving $70\,\text{cm}$ into the direction indicated by the tilt compensated magnetometer (2.60), or the cumulated and tilt compensated gyroscope (2.51). Due to previous results, tilt compensation uses the complementary filter (2.49). Magnitude-based turn-detection (2.38) was omitted after brief tests, where shaking, induced by holding the phone, accumulated rapidly, causing large drifts, rendering this variant unusable for pedestrian walks.

To ensure comparability, the heading from the magnetometer is adjusted to start at $0°$, independent of the local declination, the building's orientation (2.60), and the offset for the device being held in portrait mode (see table 2.1). Within the plotted paths, the correct orientation is used, matching with the building's floorplan, and might thus deviate. Additionally, estimated angles are modified, adding $\pm 360°$ to reduce the difference between gyroscope and magnetometer. The latter was re-calibrated before every walk. Even when figures depict the results from a single smartphone only, all walks were conducted using multiple devices. They all captured the same local influences, with only minor deviations, due to hardware differences.

For the first walk, the pedestrian held the smartphone upfront, walking four times around a $7.5 \times 2.5\,\text{m}$ railing, made of wood and metal. Results for the `Pixel 2` are shown in figure 6.16. For visualization, the gyroscope's cumulation is wrapped to $0°$ when reaching $360°$ after one complete turn. As can be seen, the estimation from the gyroscope is close to the actual path, with start (cross) and end (dot) being almost identical. Even though the drift is only minimal (see right half), the result clearly shifts with every subsequent turn. The wavy pattern, visible within the estimated heading when walking straight, is caused by the pedestrian shaking the smartphone, slightly altering its pose. While the heading derived from the magnetometer often is similar, its PDR estimation diverges significantly. The overall shape remains recognizable, but some estimations are notably different, due to outliers, caused by the metal railing. En-
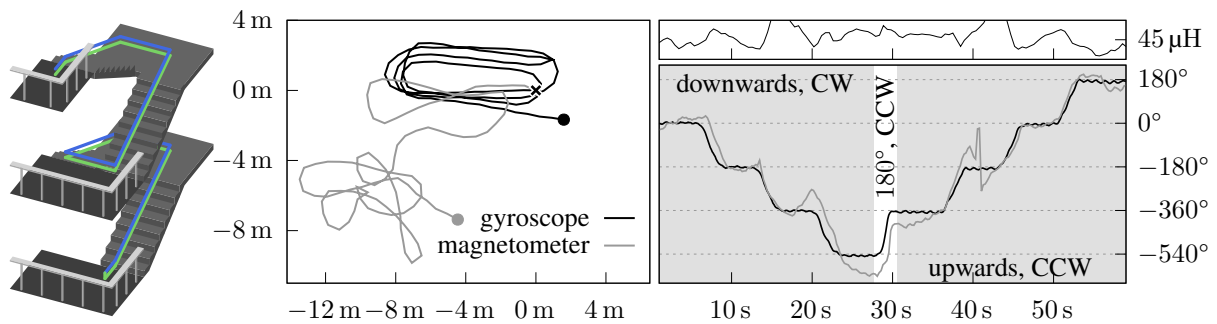
Figure 6.17: PDR for holding the `Pixel 2` upfront, walking two floors downwards in CW direction, turning by 180° CCW in place, walking two floors upwards in CCW direction, reaching the origin. The stairwell's environment affects the magnetometer significantly, indicated by changes in magnitude (upper right, showing $(45 \pm 20)\,\mu H$), causing heading inversions in some locations (lower right plot). Consequently, the path estimated from the magnetometer's heading (left plot) is rendered unusable.

vironmental influences are also notable within the sensor's magnitude, depicted in the upper right plot, showing a range of $(45 \pm 20)\,\mu H$. While the presented example seems like an ideal candidate for the complementary filter, compensating gyroscope drift with a low-pass filtered magnetometer, this does not hold true in general. Temporal environmental effects can yield heading offsets with unknown duration and impact, uncorrectable by the complementary filter.

This can be seen within a second setup, with a pedestrian walking two floors downwards in CW direction, performing a 180° turn in place, walking back to the origin, shown in figure 6.17. The environment is characterized by steel-reinforced concrete, a facade made of glass and metal-bars, and large radiators for heating the stairwell. Shown within the plots, local influences, like the radiators, can cause the indicated heading to be the opposite of the real direction, uncorrectable by initial calibration. The magnitude of the magnetometer readings, shown within the upper right plot, also indicates the effect of surrounding metal objects. However, for the depicted walk, there is no direct temporal correspondence between changes in magnitude, and heading errors. Besides heading, the estimated distance also unveils important results. While the plot indicates a walking distance in x-direction of $\approx 8\,m$, the actually walked length measures only $\approx 4\,m$. This is due to assuming a constant step length of $70\,cm$, which is incorrect while taking stairs. Here, the step length is given by the size of the stair's treads, which was $\approx 30\,cm$. As mentioned in chapter 3, this will be addressed later, by including the building's floorplan, adjusting the step length, whenever the underlying ground denotes treads of a stair.

A third walk is used to determine the long-term stability of step-detection, gyroscope and magnetometer, when walking mainly straight forward. For this setup, the pedestrian held the smartphone upfront, walking $\approx 60\,m$ straight through the building, taking one 90° CCW turn, proceeding straight for $\approx 40\,m$. Results are depicted in figure 6.18, for two different devices, the `Pixel 2` (solid lines), and the `LG G6` (dashed lines). For both, the estimated walking distance
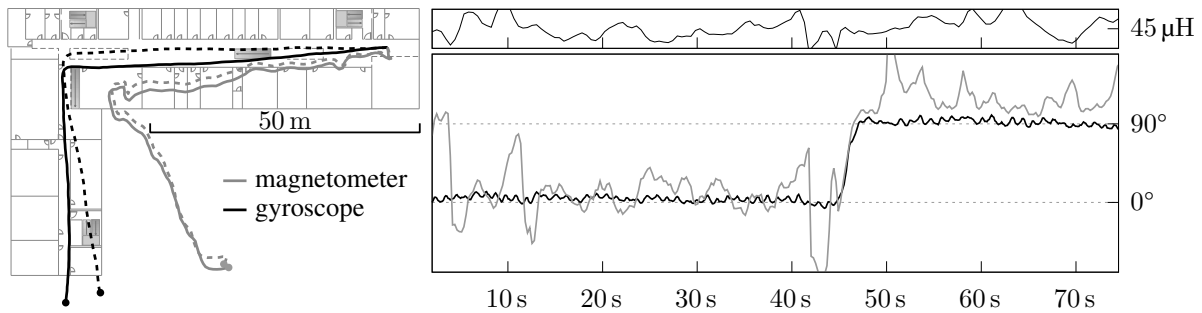
Figure 6.18: PDR for holding the `Pixel 2` upfront, walking ≈ 100 m straight, with one $90°$ CCW turn in between. The magnetometer's magnitude (upper right, showing (45 ± 20) µH) changes significantly throughout the walk, so does its estimated heading (lower right). The result for both smartphones – the `Pixel 2` (solid), and the `LG G6` (dashed) – slightly overshoots the actual destination (lower left). While the gyroscope-based PDR diverges between both, the magnetometer-based variant is notably similar.

is slightly larger than the actual value, overshooting by approximately 2 m, due to an incorrect step length constant. Concerning turn-detection, the two devices seem slightly biased, visible as individual bending of their estimated paths. For the magnetometer's estimation, both results are notably similar, showing the same local effects along the walking path, even though both were recorded one after another. Yet, besides being notably similar, and the overall shape recognizable, there are major drawbacks for the magnetometer heading's accuracy. The first half of both walks is off by $\approx 15°$ from the horizontal ground truth. Within the second half, this error increases to $\approx 25°$ from the vertical ground truth. The difference between the halves clearly indicates that this can not be addressed by a constant angular calibration offset.

The examined walks thus yield several conclusions. Similar to previous experiments, step-detection is accurate, but walking distance estimations can vary, especially when invalid assumptions on the pedestrian's step length are made. In some cases, like the depicted overshooting, this can be mitigated by including the building's floorplan, preventing impossible walks.

The gyroscope-based turn-detection is almost free of noise, accurate within shorter time-frames, but suffers from cumulating drifts, notably affecting PDR-based location estimation. Furthermore, the amount of drift, and its direction, vary between the tested devices. Again, these issues can be mitigated by including a floorplan, limiting impossible walks.

The magnetometer can provide a coarse absolute heading, at least in some parts of the building, but might suffer from unpredictable, large errors in other regions. As there was no direct correspondence between changes in magnitude and observed errors, predicting outliers seems difficult. Only within measurements belonging to straight walks, or when compared against turn-detection, assumptions on potential outliers can be made. Even after outlier filtering, the accuracy of the estimated heading remains unclear (see (6.18)), influenced by the surrounding environment. However, based on the presented results, including the magnetometer probabilis-
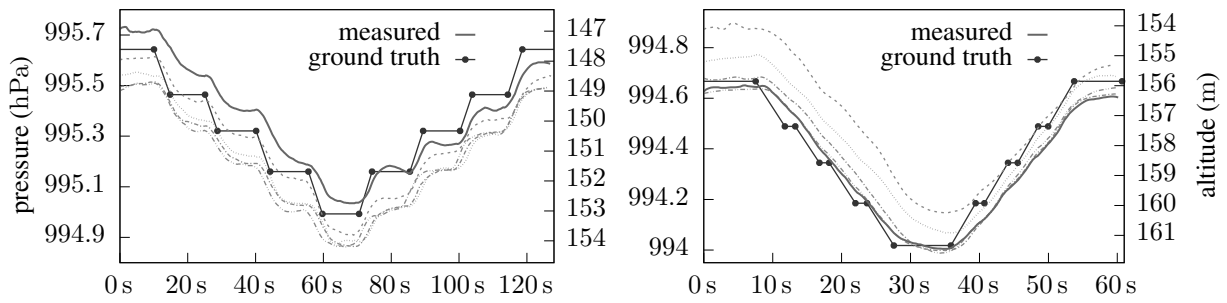
Figure 6.19: Measured pressure over time, for a pedestrian walking upstairs by two stories, then downstairs again, using the LG G6. At each plateau, the pedestrian either rested shortly (left), or kept walking (right). Solid lines denote the first walk, dashed lines all repetitions, made within a ten minute timeframe. Between the five walks on the left, and the ones on the right, one hour has passed. The ground truth was determined by (2.71), based on the building's altitude, a measured reference pressure, and the stair.

tically for a coarse heading estimation, seems feasible (cf. section 2.4.3). Corresponding results will be discussed when examining the overall system, including sensor fusion by recursive density estimation, and constraints imposed by the building's floorplan.

6.2.5 Altitude Estimation

To evaluate the contribution of smartphone barometers, discussed approaches, and issues, several walks with varying hardware, location, and ambient conditions were conducted. The experiments focus on accuracy, precision, delays, environmental influences, and consequences.

Repeatability of measurements was examined by conducting a two story walk (first up, then down) several times, within a timeframe of ten minutes, using the LG G6, during cloudy weather conditions. The difference in altitude between both stories was estimated using a *plumb bob* and is $\approx 5.54\,\mathrm{m}$. While walking, four stairs, each $\approx 1.38\,\mathrm{m}$ in height, and five plateaus (including start and end) were taken. Figure 6.19 shows the results for five consecutive first-up-then-down walks. At each plateau, the pedestrian rested either $\approx 10\,\mathrm{s}$ (left half) or kept walking (right half). The solid lines describe the first of five repetitions, all others are denoted by dashed lines. Between both variants, with and without resting at plateaus, one hour has passed, and, as can be seen, the atmospheric pressure changed by about $1\,\mathrm{hPa}$, equal to $8\,\mathrm{m}$ in altitude. Even between the repetitions, there is a difference in altitude of up to $1.0\,\mathrm{m}$, and the atmospheric pressure reading varies notably. This clearly indicates a major drawback when using absolute pressure indications. Besides, figure 6.19 also indicates a poor repeatability during each single walk, as there is no symmetry between both halves (walking upstairs/downstairs). However, compared with the $1\,\mathrm{hPa}$ difference, deviations throughout the walk were less severe.

As can also be seen, sensor readings appear unnaturally smooth, most probably due to filtering inside the sensor or operating system. This causes the plateaus, where the pedestrian

	Altitude Change		Seconds per Stair		Steps/Treads per Second	
	μ	σ	μ	σ	μ	σ
stair↑	31.5 cm/s	3.99 cm/s	4.38	0.66	1.85	0.23
stair↓	36.2 cm/s	3.78 cm/s	3.84	0.47	2.09	0.22

Table 6.9: Average stair walking speeds among several pedestrians for the staircase from figure 6.19.

was standing still, to barely appear as a flat line. For the non-resting walk, they are completely unrecognizable. That is, all measurements appear like a low-pass filtered version of the ground truth, yielding a curved and delayed output. This delay was first estimated visually, by moving the measurements to the left until they match the ground truth, indicating $\approx 3\,\mathrm{s}$. Likewise, the delay was determined mathematically via numerical optimization of an error function, given by the vertical distance between measurements and ground truth, indicating a delay of $\approx 3.4\,\mathrm{s}$.

To estimate the impact of this delay, the pedestrian's average change in altitude per second, and the time needed for climbing the examined staircase, are determined. For this, several pedestrians walked within the staircase from figure 6.19, while taking the exact time at the beginning and end of each individual stair. Results of this analysis are shown in table 6.9. As can be seen, delays presented by the LG G6's barometer are almost equal to the average time it takes a pedestrian to climb each stair, that is, half a floor. Even though the barometer indicates the first changes in atmospheric pressure slightly before the estimated delay, this still causes issues when evaluating the probability for potential pedestrian movements.

Figure 6.20 depicts the resulting evaluation, comparing barometer readings and ground truth probabilistically. For the *absolute* variant (2.73), the current atmospheric pressure and altitude above mean sea level are required for the examined stairwell. Both values together are used to convert the readings from the barometer into an *expected* altitude. This result is comparable against the walk's ground truth, using (2.73) with $\sigma_{\mathrm{alt}} = 1\,\mathrm{m}$. The uncertainty was determined empirically, based on previous evaluations. It implies that $> 99\,\%$ of all sensor readings are expected to stay within a $\pm 3\,\mathrm{m}$ boundary, or approximately one floor level. When using the *relative* approach (2.75), by using the first barometer reading as 0-reference, no absolute values are required, comparing solely relative pressure and altitude changes.

Figure 6.20 compares both variants. The *absolute* approach shows a poor repeatability. Due to changing ambient conditions, the conducted repetitions are significantly different. Thus, the absolute evaluation of barometer readings does not provide a viable solution to estimate the pedestrian's altitude. This is effectively addressed by the *relative* strategy, where all repetitions start with the same likelihood, due to the initial pressure reference. However, even the relative variant suffers from changing weather conditions, as they invalidate the initial reference even-
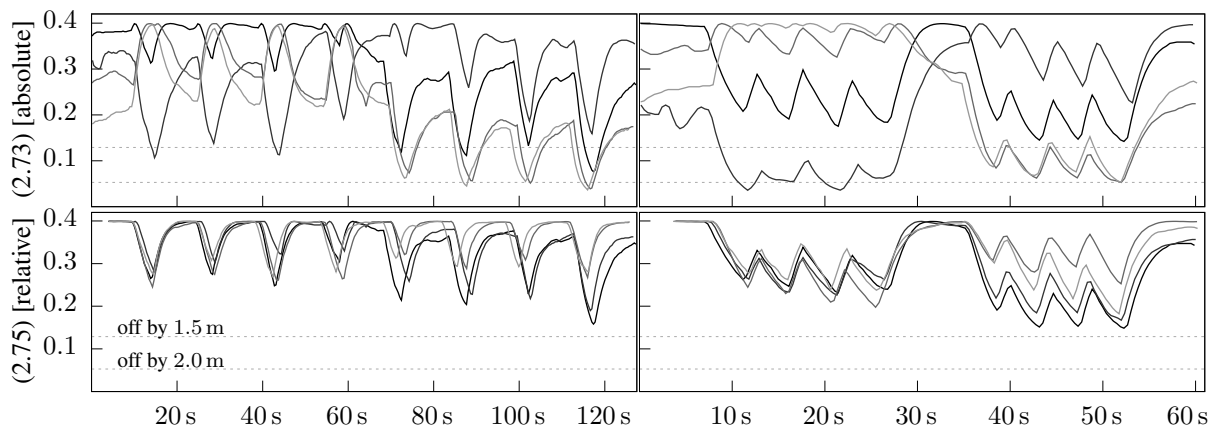
Figure 6.20: *Absolute* (2.73) and *relative* (2.75) evaluation of barometer readings for the data from figure 6.19. The absolute variant clearly lacks repeatability, due to the ambient pressure slightly changing between the conducted walks. Spikes within all results are caused by the sensor's delay. For the walks where the pedestrian rested at each plateau (left) the probability is able to recover. For the non-resting walks (right), the delay becomes even more apparent, with more and wider spikes.

tually. It is thus advisable, to use a time-limited approach, comparing the pressure and altitude change *during the last few seconds*, instead of *since start*.

For both variants, the sensor delay yields noticeable drops in likelihood whenever taking a stair, with spikes similar to being off by half a floor. When resting for $10\,s$ at each plateau, the probability is able to recover. For the non-resting variant, however, additional spikes appear and blend together, causing a period of reduced probability. Not shown within the figure, when compensating the sensor delay manually, the spikes are significantly reduced, yielding almost a horizontal line, that is, a constant probability, indicating the general viability of the sensor.

Both, the presented delays and repeatability, are sensor dependent attributes. Conducting a similar walk with a `Google Nexus 6`, equipped with a barometer from `Invensense`, yields the results shown in figure 6.21. While the direct output of the sensor is less smooth, due to a decreased number of significant digits, delays are smaller, repeatability is better, and response is more direct. Even when the pedestrian is not resting at the plateaus, they are visible within the data provided by this sensor. Furthermore, starting and ending pressure are almost identical, yielding a good repeatability of the measurements. Finally, the delay between ground truth and indicated readings is notably smaller, compared to the `LG G6` with a `Bosch BMP280` barometer. Optical and numerical analysis of the delay indicated $\approx 2.1\,s$. Thus, even though the output from the `Nexus 6` looks less pleasing, it is more useful for indoor localization and navigation.

Experiments also indicated a phenomenon, concerning the initial readings of the barometer. Shown in figure 6.21, in the beginning, the readings change rapidly by about one complete floor, stabilizing hereafter. While not being present within all recordings and/or hardware components, it was observed several times. Likewise, neither the direction nor the amount of change
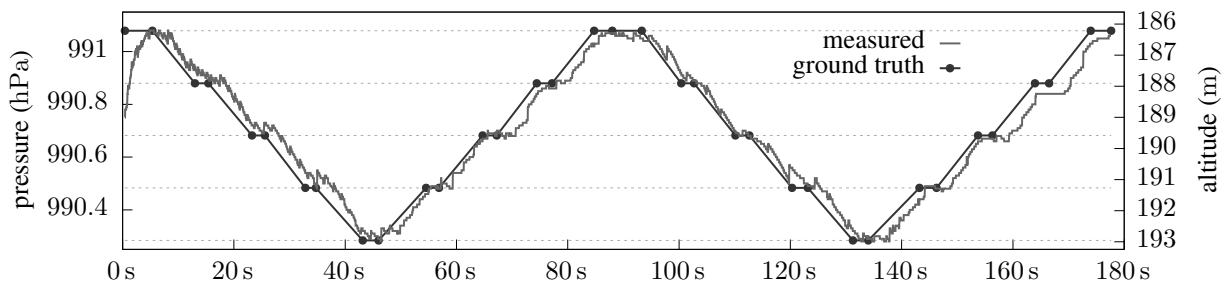
Figure 6.21: Measurements from the barometer installed within the `Google Nexus 6`, held upfront by a pedestrian, while walking a staircase up and down, twice.

were the same throughout tests. Even the timeframe the effect occurred for was varying, but ranged somewhere around $5\,\text{s}$. While the source for this behavior is unknown, it might be related to filters, installed within the sensors [Bos15; Bos18]. As this phenomenon strongly affects the relative evaluation (2.75), it must be addressed by omitting the initial readings of the barometer, until they are assumed stable, e.g. by using the first reading after several seconds as reference.

To summarize, while the barometer is able to provide hints on the current altitude, these are only stable when referring to smaller timeframes, which reduce the impact of environmental changes. The sensor is thus unsuited for absolute altitude estimations, but e.g. well suited for determining whether the pedestrian is currently taking stairs, that is, activity-detection.

6.2.6 Activity Detection

To determine the potential and accuracy of activity recognition, a tagged dataset containing all activities was recorded for several pedestrian walks with varying devices. $10\,\%$ of this dataset was used as training data, to estimate one 2D normal distribution per activity, based on the accelerometer's variance (2.77), and the barometer's delta (2.78), both within a certain timeframe. A new 2D feature pair consisting of those two values is calculated for every incoming barometer reading. Aforementioned distributions are estimated using the mean and covariance of all pairs belonging to the same activity. While smaller timeframes introduce less delays, they are expected to be less accurate due to sensor noise. Corresponding results can be seen in figure 6.22, where three different timeframes, $250\,\text{ms}$, $500\,\text{ms}$ and $1000\,\text{ms}$, are depicted. The focus of the figure is on separability. Actual values are thus omitted, and listed in table 6.11. For each activity, the estimated distribution is visualized using one ellipse for 1σ, and one for 2σ around the mean, containing $\approx 68\,\%$ and $\approx 95\,\%$ of all samples, respectively.

As can be seen, a timeframe of $250\,\text{ms}$ is too narrow to provide a viable separation between activities. Unexpectedly, however, this is not only due to the barometer, but also the accelerometer. The pedestrian slightly shaking the phone while holding it changes the short term variance
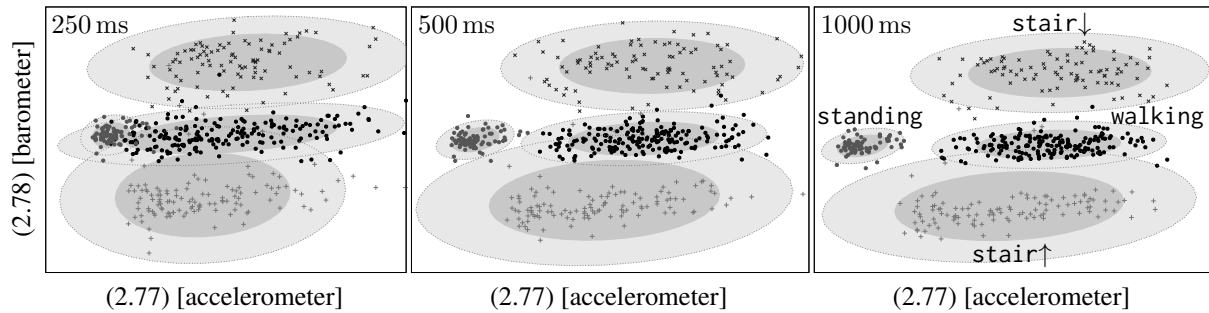
Figure 6.22: Estimated normal distributions for activity detection using the accelerometer's variance (2.77), and the barometer's delta (2.78) within three different timeframes. Dark and light gray ellipses denote the 1σ and 2σ region around the mean. For visibility, scales are omitted, and listed in table 6.11.

real \ Ω	standing	walking	stair↑	stair↓	real \ Ω	standing	walking	stair↑	stair↓
standing	95.8 %	4.2 %	0.0 %	0.0 %	standing	97.4 %	1.5 %	1.1 %	0.0 %
walking	4.2 %	91.7 %	2.2 %	1.8 %	walking	0.0 %	97.0 %	1.4 %	1.6 %
stair↑	0.0 %	3.1 %	95.1 %	1.8 %	stair↑	0.0 %	3.1 %	95.6 %	1.3 %
stair↓	0.0 %	1.4 %	0.0 %	98.6 %	stair↓	0.0 %	0.4 %	0.0 %	99.6 %

Table 6.10: Accuracy of activity recognition using one normal distribution for each activity Ω based on (2.77) and (2.78) with a 250 ms (left) and 500 ms (right) data window. The class Ω is determined by the distribution with the highest probability (2.82).

of the magnitude in a similar way the step pattern does. Therefore, standing and walking can not be well distinguished. Using 500 ms timeframes increases the separability. While the 2σ regions (light gray ellipses) of the distributions are still overlapping, the 1σ regions (dark gray ellipses) are clearly separated. Increasing the timeframe even further to 1000 ms provides only marginal improvements compared to 500 ms, and addresses the impact of noise as well as other effects on the barometer. For older barometer sensors with a lower sample rate and higher noise levels, the larger timeframe will yield improved results, at the cost of increased delays. However, as the sensors themselves also introduce lag (see section 6.2.5), additional delays introduced by the recognition step should be as short as possible.

The actual accuracy is determined by the amount of misclassifications. Therefore, each feature pair not used for training is compared against the estimated distributions, to determine the best matching one, based on the highest probability (2.82). The number of classifications for each class is then converted to a percentage. Results for the 250 ms and 500 ms timeframe are listed in table 6.10. Values for the 1000 ms timeframe are almost identical to 500 ms, and thus omitted. Similar to the overlapping seen in figure 6.22, misclassifications between standing and walking are notable for the 250 ms timeframe. As the 500 ms timeframe provides a viable tradeoff between delay and misclassifications, further examinations focus only on this setup.
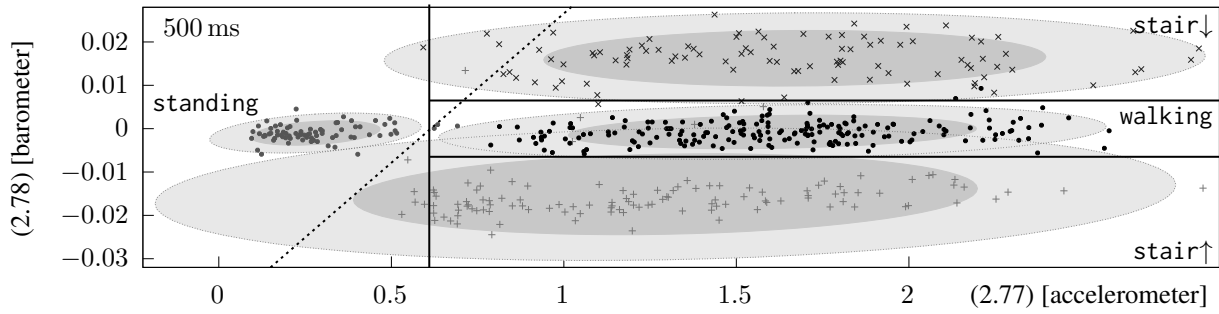
Figure 6.23: Heuristic thresholds τ_{accel} and τ_{baro} (solid lines) for the decision tree classifier. Ideally, the vertical split should be diagonal (dashed), to correctly distinguish between stair↑ and standing.

Ω	(2.77)		(2.78)		Ω real	standing	walking	stair↑	stair↓
	μ	σ	$\mu \times 10^{-2}$	$\sigma \times 10^{-3}$					
standing	0.28	0.12	−0.10	1.86	standing	97.0 %	3.0 %	0.0 %	0.0 %
walking	1.60	0.39	−0.08	2.54	walking	0.1 %	98.1 %	0.4 %	1.4 %
stair↑	1.29	0.60	−1.51	6.16	stair↑	3.2 %	3.4 %	92.1 %	1.3 %
stair↓	1.67	0.48	1.62	4.23	stair↓	0.5 %	1.7 %	0.0 %	97.8 %

Table 6.11: Mean and standard deviation of the estimated distributions for the $500\,\text{ms}$ window (left), and corresponding classification results when using a decision tree derived from these values (right).

In the following, the two thresholds τ_{accel} and τ_{baro} for the decision tree are estimated. This can be done visually, by dividing the distributions in figure 6.22, or mathematically, using mean and covariance of the distributions. The discussed decision tree (cf. figure 2.17) only allows for horizontal (barometer) and vertical (accelerometer) splits. When deriving τ_{accel} and τ_{baro} mathematically, only the center diagonal of each covariance matrix needs to be considered. The corresponding μ and σ for each activity are listed in table 6.11. Figure 6.23 shows the splits τ_{accel} and $\pm\tau_{\text{baro}}$, placed by optically separating the distributions. The resulting values, $\tau_{\text{accel}} = 0.61$ and $\tau_{\text{baro}} = 0.0065$, are similar to using $\approx \mu \pm 2.5\sigma$ for the values from table 6.11. The listed average pressure change of $0.016\,\text{hPa}$ per $500\,\text{ms}$, when walking upstairs or downstairs, equals a change in altitude of approximately $0.25\,\text{m/s}$, and depends on the pedestrian's walking speed, and the height of the stair's treads. Also shown in figure 6.23, the decision tree will cause classification errors between standing and stair↑, as a correct split would require a diagonal (dashed line). The results, using the $500\,\text{ms}$ window with the decision tree are listed in table 6.11, clearly denoting the mentioned classification error.

In some cases, the mentioned numeric precision errors (cf. section 2.6), that can e.g. occur when estimating the average value of many sensor readings within a certain timeframe, were encountered. Even though all errors were relatively small, they did affect results when being used within (2.77) to determine the variance. Without compensated summation, this often resulted in $\mathbb{E}(\mathcal{X}^2) - (\mathbb{E}(\mathcal{X})^2) < 0$, and thus an invalid square root for the standard deviation.

Model	2.4 GHz only				2.4 GHz & 5 GHz			
	Connected		Disconnected		Connected		Disconnected	
Motorola Milestone 2	0.46	0.011	0.53	0.004	-	-	-	-
Motorola Milestone 4	1.35	0.055	1.85	0.450	-	-	-	-
Samsung Galaxy S3 Mini	1.06	0.042	0.85	0.015	0.25	0.003	0.28	0.001
Google Nexus 6	2.74	0.131	1.84	0.066	0.44	0.769	0.34	0.001
Samsung Galaxy S5 Neo	1.11	0.027	0.88	0.039	0.25	0.002	0.29	0.002
Motorola Moto Z*	-	-	-	-	0.07	0.042	0.07	0.042
LG G6	1.11	0.081	0.97	0.072	0.25	0.041	0.30	0.024
Google Pixel 2*	-	-	-	-	0.07	0.036	0.07	0.036
BlackBerry KEY2	-	-	-	-	0.19	0.031	0.45	0.013

Table 6.12: Mean sample rate (in Hz) and standard deviation of the Wi-Fi component within each phone listed in table 6.1. Data was collected with both, a currently active Wi-Fi connection, and a disconnected component. Scanning was conducted using the 2.4 GHz range only, and 2.4 GHz & 5 GHz combined, if possible. Dashes indicate that some mode is unsupported by the phone's hardware or operating system.

To summarize, activity-detection based on accelerometer and barometer provides a viable accuracy for a smartphone held upfront, independent of using a Bayes classifier, or a decision tree. Additional delays, besides the ones from the sensors themselves, can be as low as $500\,\text{ms}$. While even lower values are possible, they affect accuracy, and are thus not recommended.

6.2.7 Wi-Fi Location Estimation

Within the following, the quality of Wi-Fi-based absolute location estimation is examined. This covers a brief overview on Wi-Fi hardware installed within smartphones, to determine data acquisition speed, crucial for instantaneous system behavior. Hereafter, signal strength prediction accuracy is examined, by optimizing the discussed signal strength prediction models, using reference measurements recorded within the buildings from section 6.1. Based on the trained prediction models, a Wi-Fi-only absolute location estimation is performed, determining its accuracy, potential drawbacks within real-world scenarios, and how to address them.

Data Acquisition   Mentioned in section 2.7, one potential drawback of the Wi-Fi-component is the time required to scan for nearby transmitters. It depends on the actual implementation within the hardware component and its driver, whether the smartphone is currently connected to some transmitter, and the to-be-scanned frequency range. For some devices, the latter can be limited by software, scanning only the 2.4 GHz band, ignoring the 5 GHz range, providing faster scanning times, but reducing the number of visible transmitters. Table 6.12 lists a brief comparison for all examined smartphones, testing both, the connected and disconnected state, and, when supported, a scan limited to the 2.4 GHz range.

Strategy	Model	Obstacles	Setup
oELD	log-distance model	none	empiric, one model per AP
oLD	log-distance model	none	optimized, one model per AP
oLDC	extended log-distance model	ceilings only	optimized, one model per AP
oLDCW	extended log-distance model	ceilings & walls	optimized, one model per AP
oLDCPF	extended log-distance model	ceilings only	optimized, one model per AP and floor

Table 6.13: List of all strategies used for optimizing signal strength prediction models.

As can be seen, the scanning rate varies significantly among all examined devices, ranging form $0.46\,\mathrm{Hz}$ to $2.74\,\mathrm{Hz}$ when using only $2.4\,\mathrm{GHz}$, and from $0.07\,\mathrm{Hz}$ to $0.44\,\mathrm{Hz}$ when scanning both ranges. However, the $0.07\,\mathrm{Hz}$ is due to the software limitation introduced with Android 9, restricting the allowed scanning rate for the devices marked with an asterisk. The listed results also denote a few peculiarities. When examining only $2.4\,\mathrm{GHz}$, scanning is often faster, when the device is currently connected to some access point. When examining both frequency ranges, however, the opposite is true, increasing the scanning speed, when disconnecting from the base station. While using both ranges increases the number of measurements, assuming $5\,\mathrm{GHz}$ transmitters are installed within a building, it significantly increases scanning times, causing delays. Therefore, it is advisable to limit scanning to the $2.4\,\mathrm{GHz}$ band, if supported, to increase the responsiveness of the Wi-Fi component. If limiting is supported, Wi-Fi can provide absolute location information approximately every second, similar to typical GPS receivers.

Signal Strength Prediction Strategies   Mentioned in section 2.7.5 and 2.7.6, to perform a probabilistic Wi-Fi location estimation, either a database of fingerprints, or a signal strength prediction model is required. As fingerprinting is time-consuming, this work focuses on prediction models only. Within the following, several models and optimization strategies (see table 6.13) are examined and compared regarding their prediction quality for real-world scenarios. This covers: an instantaneous setup, when the locations of all installed transmitters are known, and aforementioned optimization strategies based on a few reference measurements.

oELD   If the locations of all transmitters installed within a building are known, the fastest setup strategy uses the log-distance model (2.87) with these locations, and an empiric choice for P_0 and γ, yielding one model per transmitter. If all transmitters use identical hardware, the same P_0 can be used. γ, however, is chosen based on the transmitter's surroundings. For buildings with homogeneous interior, the same value can be used for all transmitters, ranging somewhere between 2.0 (mainly free space attenuation), and 3.0 (typical drywall/concrete mixtures). As no reference measurements are required, the setup time is reduced to a minimum, at the cost of accuracy, with unknown prediction quality, therefore referred to as empiric log-distance (oELD).
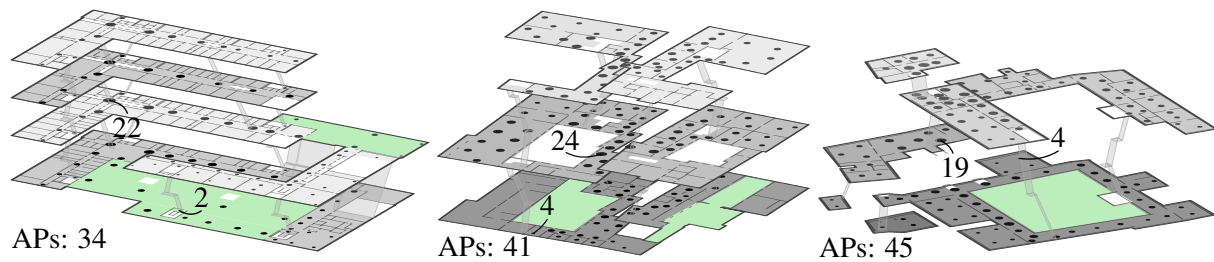
Figure 6.24: Location of reference measurements within the three buildings from figure 6.1-6.3, indicating the number of visible transmitters by sized dots, and their total number in the lower left.

oLD   For some buildings, the locations of installed transmitters might be unavailable, or undisclosed. Here, reference measurements can be used, to optimize the log-distance model (2.87), estimating each transmitter's location, P_0 and γ. For single-floors and drywalled interior, this strategy can provide sufficient accuracy, and will be referred to as log-distance (oLD).

oLDC   As steel-reinforced concrete strongly attenuates radio signals (cf. section 2.7.1), floors and ceilings should be considered within multistory buildings, when estimating signal strengths for transmitters installed within adjacent floors. Similar to oLD, this strategy uses reference measurements, but trains the extended log-distance model (2.88). It includes an additional attenuation factor for each transmitter's model, to consider the impact of floor/ceiling attenuation, based on the building's floorplan, therefore referred to as log-distance with ceilings (oLDC).

oLDCW   When the building's interior relies on a mixture of materials, such as drywalls, glass and concrete, including all individual obstacles can be required. Here, reference measurements are used to train an extended log-distance model (2.88), not only including floors/ceilings, but also all wall-obstacles, using an individual attenuation factor for every material. As mentioned earlier, while this offers an improved prediction quality, it significantly affects performance, requiring numerous intersection tests to estimate a single signal strength. Within the following, this strategy is referred to as log-distance with ceilings and walls (oLDCW).

oLDCPF   The complexity of oLDCW can be reduced by a tradeoff, where only floors/ceilings are considered, and the model for each transmitter is not trained for the whole building, but for individual regions within it, e.g. on a per-floor basis. Based on the reference measurements for one floor, one extended log-distance model (2.88) including only floors/ceilings is trained for every visible transmitter, referred to as log-distance with ceilings, per floor (oLDCPF).

Within the following, all discussed strategies are applied to the buildings introduced in section 6.1, providing a detailed examination of achievable results. The empiric oELD is provided only for SHL and Museum 2, as the required transmitter locations were unknown for Museum 1. Mentioned earlier, reference measurements, required for the other strategies, were recorded by

Building	APs	FPs	oELD	oLD	oLDC	oLDCW	oLDCPF
SHL	34	121	8.7 dB	6.3 dB	4.7 dB	4.0 dB	3.1 dB
Museum 1	41	139	-	4.8 dB	4.2 dB	3.6 dB	2.6 dB
Museum 2	45	130	11.2 dB	5.4 dB	4.3 dB	3.3 dB	2.7 dB

Table 6.14: RMSE (2.113), but among all access points of each examined building (see section 6.1 and figure 6.24), and chosen optimization strategy (cf. table 6.13).

holding a smartphone upfront, slowly turning around in a circle, while scanning, yielding multiple measurements for each transmitter visible at a location. Here, only static transmitters, belonging to the building's Wi-Fi infrastructure, were considered. That is, all dynamic access points, such as smartphone hot spots, smart presenters, Wi-Fi-equipped televisions, and similar, were omitted. Figure 6.24 depicts all reference measurements, with their dot-size representing the number of distinct transmitters visible at each location. While this number is not a direct indicator for prediction and localization quality, it can provide hints on sensitive areas, where the number of visible transmitters is too low to provide a stable localization.

The quality of the trained models is quantified by the RMSE among all transmitters, similar to (2.113). That is, the square root of the average squared difference between model prediction and reference measurement, for all fingerprints, access points, and repetitions. While this metric does not indicate outliers or regionally limited issues, it yields a brief quality estimator, shown in table 6.14. A corresponding probabilistic evaluation is provided by a cumulative distribution function, indicating the percentage of estimations that stay below a certain absolute error value

$$P(\mathcal{X} < x), \quad \text{with } \mathcal{X} = \overbrace{\left\{ \left| P_{\mathrm{mdl}} \left(\mathbf{pos}_{\mathrm{xyz}} \left(\mathrm{fp} \right), \psi_{\mathrm{ap}} \right) - \varsigma_{\mathrm{fp,ap},n} \right| \mid \forall \mathrm{fp}, \mathrm{ap}, n \right\}}^{\text{absolute differences between all predictions and measurements}}, \quad (6.1)$$

shown in figure 6.25. As can be seen in table 6.14 and figure 6.25, an empiric choice, using the same P_0 and γ for all transmitters, yields the highest RMSE. Here, the same $P_0 = -40\,\mathrm{dBm}$ and $\gamma = 3$ were used for Museum 2 and SHL, with significantly varying error results. This is due to the number of transmitters installed within Museum 2 being more dense, and the architecture of both buildings being completely different, requiring smaller values for γ.

For all examined cases, the RMSE and the corresponding CDF fully agree. Smaller RMSEs are due to improvements throughout the whole error spectrum (see figure 6.25). The quality of the optimization strategies follows the order they were presented in. As expected, optimizing one log-distance model per transmitter based on reference measurements (oLD) yields a lower RMSE than a pure empiric choice, using the same P_0 and γ for all access points (oELD). Additionally including floors/ceilings (oLDC) further increases the prediction quality, especially for
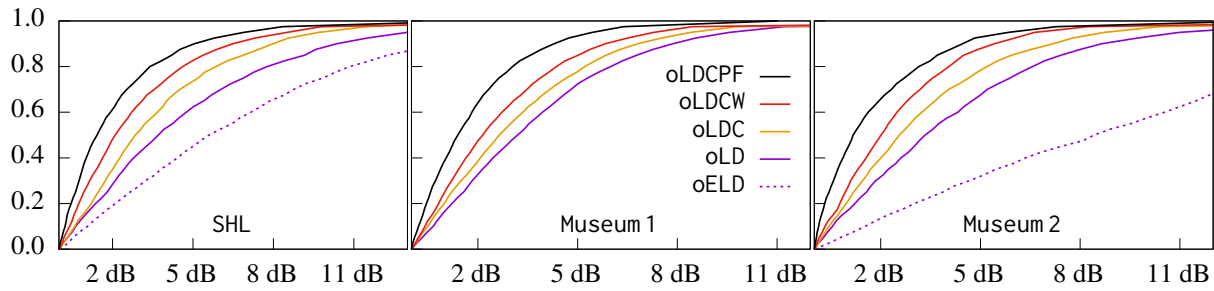
Figure 6.25: Result of the cumulative distribution function (6.1), for the absolute difference between all reference measurements and corresponding model predictions, based on the examined building, chosen model, and optimization strategy. As expected, the empiric variant oELD yields the largest errors.

the SHL and Museum 2. Intuitively, the amount of potential improvements strongly depends on the building's architecture. This can be seen by additionally including wall obstacles (oLDCW), where Museum 2 gained the most improvements, due to its massive concrete walls. Intended as a compromise, the per-floor optimization without wall-knowledge (oLDCPF) provided the best results for all examined buildings. This is due to the complex nature of radio signals, being affected by far more influences than are addressed by the presented models, such as reflection and refraction. By regionally limiting the signal strength prediction model, the number of effects to be dealt with is reduced, thus increasing the chance for better predictions. While using even smaller regions than *per-floor* is possible, it is not always advisable. Mentioned within the theoretical discussion, training requires a decent amount of reference measurements to be stable and converge. With smaller regions, the chance for potential issues, and poorly optimized models for some transmitters, increases.

Locally limited issues, e.g. within concrete stairwells, can be visualized by calculating the average signed, and the maximum absolute error, for every location where reference measurements were conducted. That is, at every such location, each received RSSI for every known transmitter is compared against the corresponding model prediction for the same location

$$\begin{aligned}\varepsilon_{\mathrm{avg}}\left(\mathrm{fp}\right) &= \mathbb{E}(\mathcal{X}) \\ \varepsilon_{\mathrm{max}}\left(\mathrm{fp}\right) &= \max(|\mathcal{X}|)\end{aligned} \quad \text{with} \quad \mathcal{X} = \overbrace{\left\{ P_{\mathrm{mdl}}\left(\mathbf{pos}_{\mathrm{xyz}}\left(\mathrm{fp}\right), \boldsymbol{\psi}_{\mathrm{ap}}\right) - \varsigma_{\mathrm{fp,ap},n} \mid \forall \mathrm{ap}, n\right\}}^{\text{differences between predictions and measurements, for one fp}}. \quad (6.2)$$

The average signed error provides an impression whether a model tends to predict too weak or too strong signal strengths for a location. Similarly, the maximum absolute error yields an overview on the model's worst-case behavior. The results for both metrics, and all examined buildings are shown in figure 6.26, 6.27, and 6.28.

As can be seen, while the behavior of the maximum absolute error is approximately comparable between the buildings, the signed average behaves differently. The SHL (cf. figure 6.26)
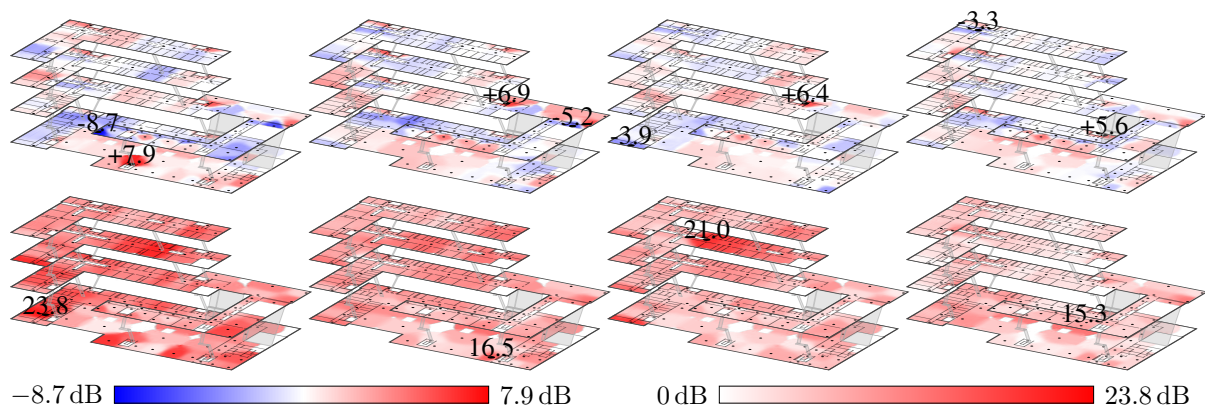
Figure 6.26: `SHL` (cf. figure 6.1) – average signed (top) and maximum absolute (bottom) deviation between model predictions and reference measurements, when using `oLD`, `oLDC`, `oLDCW`, `oLDCPF` (left to right). Within the top row, blue indicates too weak, and red too strong model predictions.
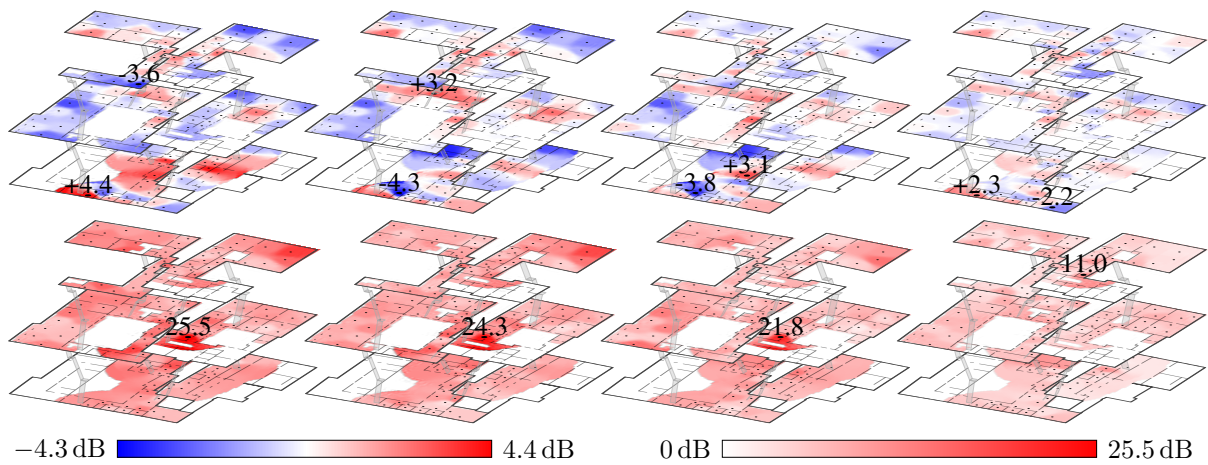
Figure 6.27: `Museum 1` (cf. figure 6.2) – average signed (top) and maximum absolute (bottom) deviation between model predictions and reference measurements, when using `oLD`, `oLDC`, `oLDCW`, `oLDCPF` (left to right). Within the top row, blue indicates too weak, and red too strong model predictions.
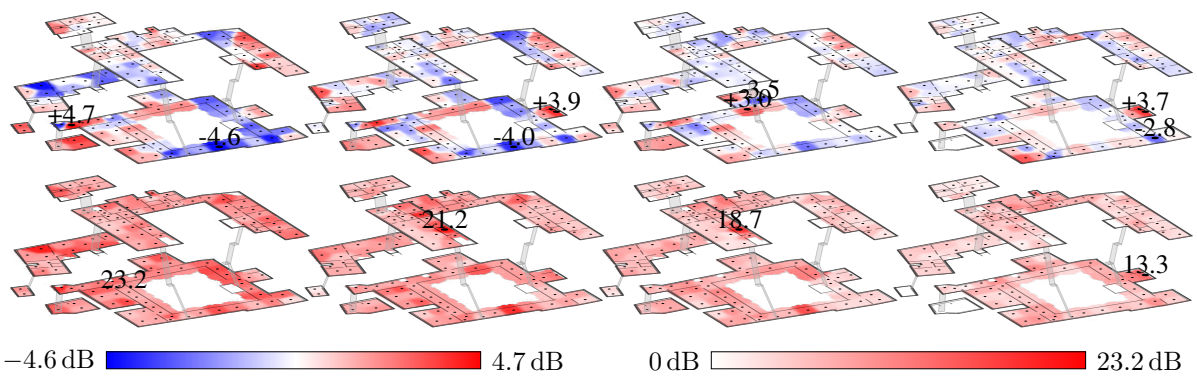
Figure 6.28: `Museum 2` (cf. figure 6.3) – average signed (top) and maximum absolute (bottom) deviation between model predictions and reference measurements, when using `oLD`, `oLDC`, `oLDCW`, `oLDCPF` (left to right). Within the top row, blue indicates too weak, and red too strong model predictions.

denotes the highest errors, which is due to a lower number of transmitters per building size. With Museum 1 (figure 6.27) and Museum 2 (figure 6.28) being smaller, regional effects become more pronounced, visualized by red and blue alternating within several areas. Yet, for all buildings, regional errors follow previous findings, decreasing in the order of the examined optimization strategies. Similar to the RMSE, including wall-obstacles improves only some regions of the SHL, while using oLDCPF enhances predictions throughout the whole building, indicated by less saturated colors for both, signed average, and absolute maximum. Museum 1 is similar, showing most improvements for oLDCPF. Within Museum 2, however, including wall-obstacles clearly enhances prediction quality, by considering the building's massive concrete exterior.

Effects of certain materials on radio signal propagation is best explained within SHL, where the indoor and outdoor areas of the first floor (cf. figure 6.1) are separated by a facade of metallized glass. This causes attenuations of $10\,\mathrm{dB}$ and above, visible within the average signed errors of figure 6.26, where the outdoor area is predicted too strong (red), and indoor areas too weak (blue). While this can be mitigated by considering such obstacles (oLDCW), required intersection tests are costly. Thus, the prediction from oLDCPF, with an average error of $\approx 3\,\mathrm{dB}$ is an ideal candidate for a computationally efficient, model-based location estimation.

Concerning the number of reference measurements required for a stable model estimation, there is no rule of thumb. For simple models, like oLD, half of the shown reference measurements can be removed, without significantly affecting the prediction quality of the resulting models. For more complex variants, especially oLDCW, a decent number of reference measurements is mandatory for a stable optimization. This number thus represents a tradeoff between accuracy and setup time. Ideally, the whole walkable area is covered uniformly with measurements, to prevent regional overfitting. Similarly, neuralgic regions, like stairwells, should be included as well, to provide a general signal strength prediction solution.

Location Estimation Based on the optimized prediction models, Wi-Fi location estimation can be performed. Within the following experiments, the whereabouts are estimated for every Wi-Fi scan s from the recorded walks, and compared with the corresponding ground truth (cf. figure 6.4, 6.5 and 6.6). Whereabouts are determined by two different strategies:

lBest The first strategy estimates the current location by numerically optimizing (2.111) with $\sigma_{\mathrm{rssi}} = 8$, determining the best matching result. To increase the chance of finding a global maximum, the optimization process is repeated $10\,000$ times, each starting from a different random location, which belongs to the walkable area of each building.

lFilter The second strategy is based on the particle filter, and starts by uniformly sampling the walkable area, using 5000 particles. Each particle is evaluated, assigning a weight given by (2.111) with $\sigma_{\mathrm{rssi}} = 8$. The weighted average of all particles represents the first location

estimation (see section 4.5). For every new Wi-Fi scan, the process continues from the previous set of particles, after scattering it by (3.10), with $\mu_{\text{walk}} = 2.5\,\text{m}$ and $\sigma_{\text{walk}} = 1.0\,\text{m}$. In other words, potential whereabouts are modeled by random samples. Between two Wi-Fi scans, these samples are relocated, but only within aforementioned constraints. After this spreading, they are re-weighted by the current Wi-Fi observation. Afterwards, their weighted average denotes the next location estimation. Eventually, resampling is applied, to ensure a decent amount of particles remain efficient. For the next Wi-Fi scan, the process starts all over. The resulting behavior is similar to (3.8), prevents the Wi-Fi estimation from causing large jumps in location, and supports multimodalities unsupported by the Kalman filter.

While the first strategy (lBest) returns estimations that are independent of the previous ones, the particle filter, used for the second strategy (lFilter), implicitly includes previous measurements, mitigating the impact of outliers, and regionally poor signal strength predictions. As no floorplan is included for now, its behavior is continuous, similar to a Kalman filter.

An example for one repetition of walk A1 is shown in figure 6.29. Even if examining only a single walk, this is without loss of generality, as all others indicate similar effects. Comparing the shown setups, the benefits of the particle filter (lFilter) become visible. Without filtering (lBest), the location estimations based on the oLD optimization strategy are noisy, with large regional errors. The particle filter suppresses outliers by restricting location changes between two measurements via (3.10). This yields smoother results, with reduced errors. Yet, filtering only improves temporal effects, but is unable to address long-term issues, e.g. based on invalid signal strength predictions, occurring around $10\,\text{s}$ and $20\,\text{s}$.

These errors can be addressed by using more complex signal strength prediction models, like the depicted oLDC. Shown within the figure, this reduces the error between $10\,\text{s}$ and $20\,\text{s}$ from $30\,\text{m}$ to $12\,\text{m}$, as the model provides better predictions. Also shown within the figure, oLDC significantly improves the location estimation in z, when compared with oLD. Mentioned earlier, oLDC considers floors and ceilings within signal strength predictions, by including discrete attenuations. Predicted signal strengths will thus vary notably between adjacent floors, which matches with the actual real-world behavior. This also holds true for oLDCW and oLDCPF, not shown within the figure. Most of the time, their predictions match best with the correct floor, rendering adjacent floors unlikely, as shown within the figure. However, for all presented instances, the upper right outdoor area, labeled as (3), clearly causes significant problems. These errors are examined by using another visualization of potential whereabouts.

Figure 6.30 depicts a probability heat map (2.111) for two points in time of walk A1. It denotes the likelihood for residing at any location within the building, for the Wi-Fi measurements received at $t = 14\,\text{s}$ directly after the start (top row), and at $t = 97\,\text{s}$ having reached the outdoor area (bottom row). For the latter, all examined signal strength prediction models fail. Several
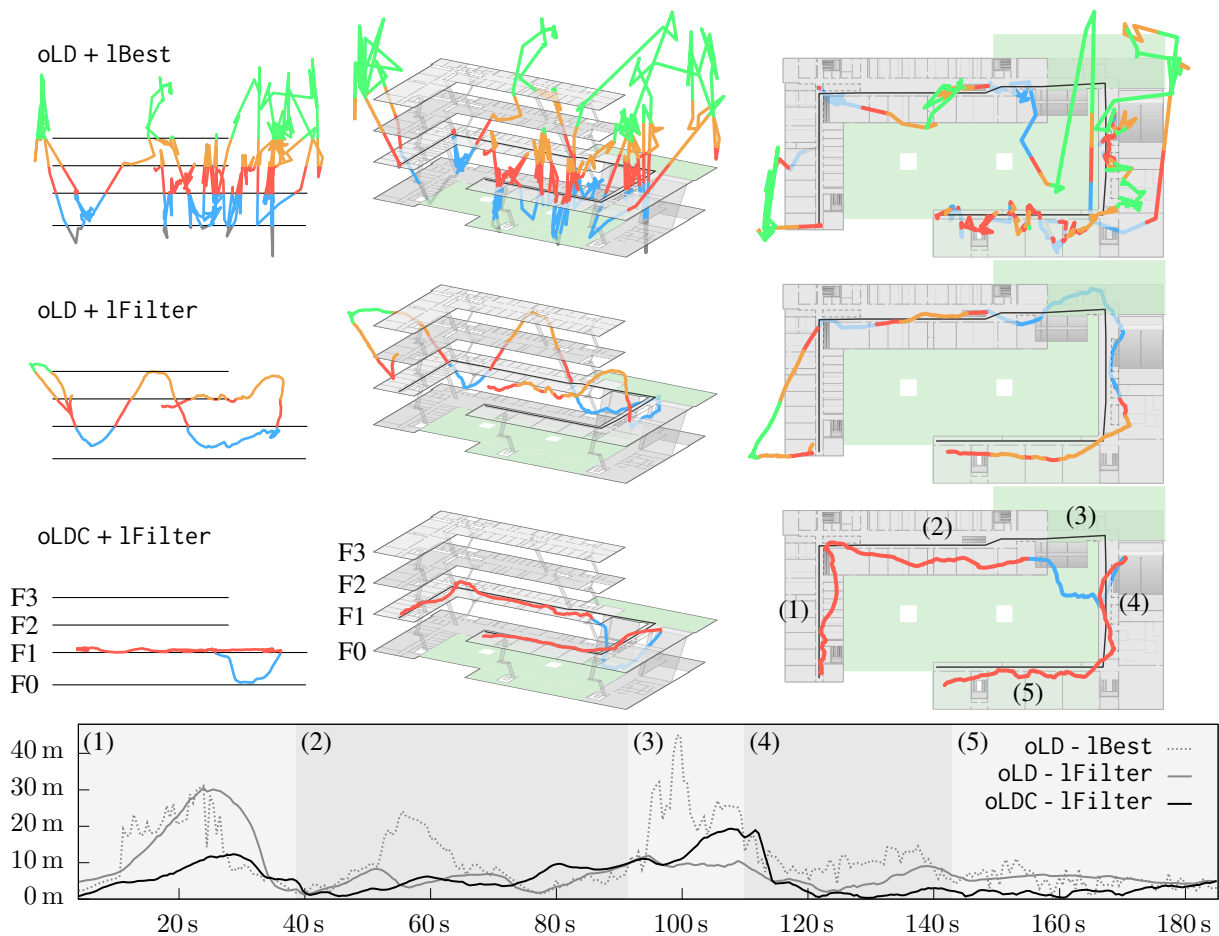
Figure 6.29: Location Estimation (top) and 3D error over time (bottom), for one repetition of walk A1 (see figure 6.4), when using oLD + lBest, oLD + lFilter, and oLDC + lFilter. The three columns show the side-view (left), perspective (center), and top-view (right) of each estimation result.

areas within the building are more likely than the actual ground truth (black dot). This is due to the building's facade made of metallized glass, attenuating radio signals, thus leaving a blind spot. In such situations, even the accurate fingerprinting can fail, as the number of receivable transmitters is low, and their RSSIs are all rather similar.

Figure 6.30 also denotes the impact of including attenuating ceilings and obstacles. For oLD, the predictions for adjacent floors are similar, as they are only a few meters apart, not causing major free space attenuation. Not being able to provide floor estimations is a major drawback of oLD's simplicity. For all other methods, the floor estimation is mostly accurate. However, within unusual buildings like Museum 2, the estimation can fail, due to their irregular floor layout.

Besides the shown benefits, including obstacles causes unwanted discontinuous behavior of the signal strength prediction. As soon as an obstacle intersects the line of sight, it causes a drop in signal strength. Due to physical effects like reflection and diffraction, the real-world behavior deviates, is smoother, and less discontinuous. However, these effects are not considered by most
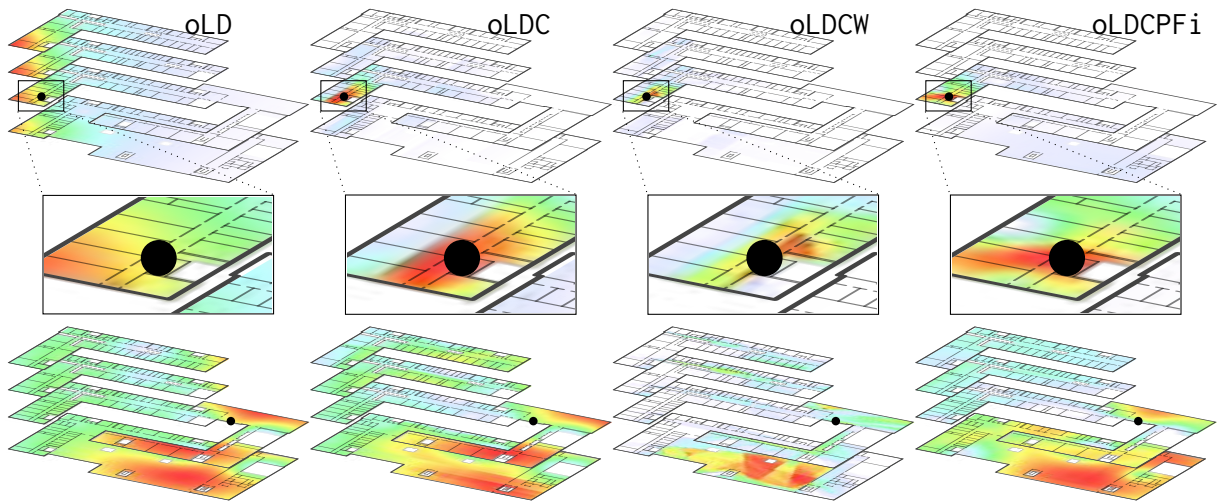
Figure 6.30: Probability heat map (2.111), from warm (likely) to cold (unlikely), showing potential whereabouts for walk A1 at $t = 14\,\text{s}$ (top) and $t = 97\,\text{s}$ (bottom), based on the chosen optimization strategy. The ground truth is indicated by a dot. The additional oLDCPFi is explained within the text.

prediction models. This aspect also influences the probability for residing at certain locations, which will show the same discontinuous drops. For oLDC, this behavior is expected between adjacent floors only, where ceilings are considered. For oLDCW, the impact is also expected along each floor, as walls are considered as well. However, shown in the magnified region of figure 6.30, oLDC also shows discontinuous behavior along floors, which is unexpected. The effect is caused by the optimization, which sometimes places access points *outside* of the building. This affects the behavior of ceiling intersection tests. When transmitters reside outside of the building, there is not necessarily a ceiling along the line of sight, even when the transmitter is on another floor. This is addressed by oLDCPFi, a modified version of oLDCPF – which is similar to oLDC concerning this behavior – forcing AP positions to remain *inside* the building, by adding a penalty to the target function. As can be seen, this mitigates discontinuous behavior. Even though the resulting signal strength prediction resulted in slightly increased errors ($0.2\,\text{dB}$), the location estimation is less erroneous, as the density is more continuous. Yet, this is not a complete fix, as irregular building shapes, as in SHL and Museum 2, can still contain constellations with no ceiling along the line of sight from a transmitter to a location on another floor.

Figure 6.31 depicts one cumulative distribution function (CDF) per building, containing the localization errors of all walks. These are given by the 3D distance between each position estimated from a Wi-Fi scan s, using either lBest and lFilter, and the ground truth interpolated for the scan's timestamp (see section 6.1). Additionally, the error in z is determined separately, as correctly indicating the current floor is a significant benefit. Matching with previous discussions, the three buildings behave differently, due to architectural and infrastructural differences.
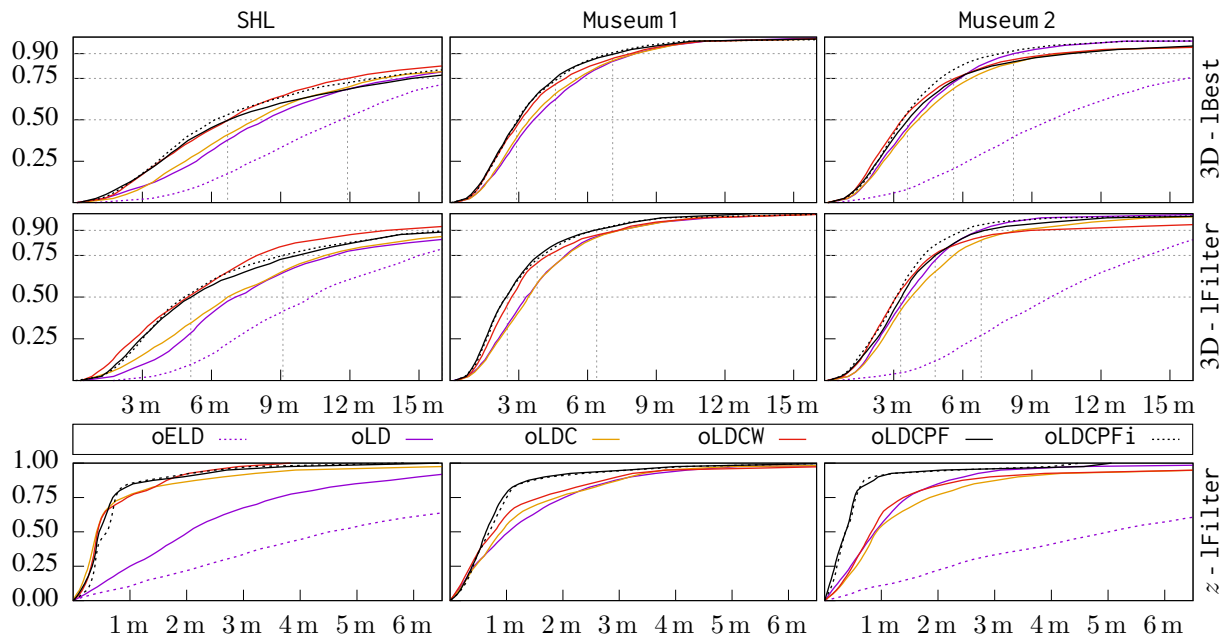
Figure 6.31: CDF of the 3D/z localization error, for the walks from section 6.1, based on the previously introduced optimization strategies, plus one additional variant, described within the text. The difference between the `lBest` (top) and `lFilter` (center) location estimation strategy is clearly visible.

The top row of figure 6.31 shows the CDFs based on all walks within each building, when using the `lBest` localization strategy. As can be seen, `SHL` causes the largest errors, with $50\,\%$ of them being worse than $\approx 6.6\,\mathrm{m}$. This is due to a comparatively low number of installed transmitters, covering approximately $350\,\mathrm{m}^2$ per AP. For `Museum 1` and `Museum 2`, this value is $90\,\mathrm{m}^2$ and $67\,\mathrm{m}^2$, respectively, thus yielding better results. Furthermore, some of the walks conducted within `SHL` used the outdoor area, which is not well-covered by radio signals, due to the facade made of metallized glass. These outdoor parts yield large localization errors (see figure 6.29), and cause $10\,\%$ of all 3D location estimation errors to be greater than $\approx 20\,\mathrm{m}$. For the two other buildings, this value is as low as $\approx 7\,\mathrm{m}$ and $\approx 10\,\mathrm{m}$, respectively.

The center row of figure 6.31 depicts the CDFs when using `lFilter` as location estimation. Compared with `lBest`, the resulting errors are notably reduced, which matches earlier findings. While the lower $50\,\%$ of all errors remain almost unaffected, the last $25\,\%$ and $10\,\%$ denote significant enhancements. This behavior is as expected, with the recursive density estimation mainly suppressing outliers, by preventing major jumps within the estimated location, based on the described movement restriction. The depicted improvements, especially for `SHL`, clearly indicate the possibilities offered by recursive density estimation.

The bottom row of figure 6.31 depicts the CDFs for the error in z, when using `lFilter` as location estimation. For `SHL`, the results are as expected. As soon as ceilings are considered by the model, z-estimations become viable. For the other two buildings, however, z-errors are
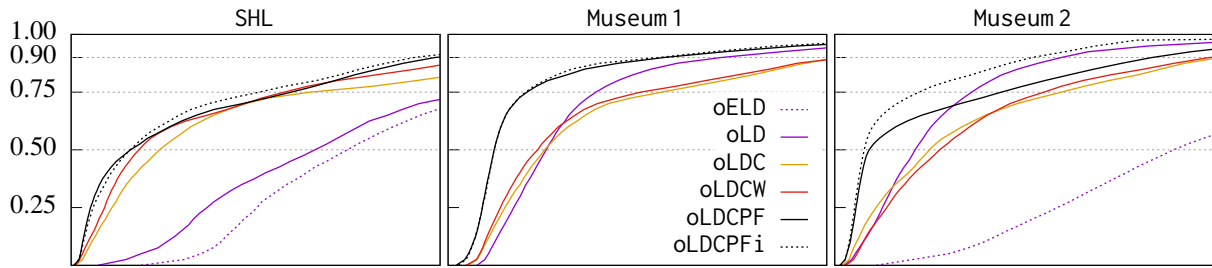
Figure 6.32: CDF of the Kullback-Leibler divergence (KLD), for the walks from section 6.1, based on the previously introduced optimization strategies, plus one additional variant, described within the text. As the KLD's value does not convey a quantifiable meaning, the x-axis is unlabeled.

moderate even when ceilings remain unconsidered. This is due to the number of densely packed transmitters, generally providing a higher accuracy, also affecting z.

When comparing figure 6.25 and 6.31, it becomes clear that better signal strength prediction models do not necessarily yield better location estimations, and that location errors are relatively similar among most models. While oLDCPF provides the best estimations for SHL and Museum 1, it does not for Museum 2, where $20\,\%$ of the results denote larger errors than the ones provided by the simple oLD. While this is mitigated by lFilter, it still remains present. Forcing all estimated transmitter positions to remain inside the building (oLDCPFi) not only addresses discontinuities, but also suppresses additional outliers, yielding the best overall results.

Location Probability Figure 6.31 determined Wi-Fi localization quality by comparing the most likely whereabouts against the ground truth. While this is a common strategy for error estimation, it is not ideal for the intended use case. Within the overall localization and navigation system, Wi-Fi is meant to weight particles on whether their state (location) is likely, or unlikely, based on the current signal strength readings from the smartphone. In other words, Wi-Fi is not used to directly infer a location (x, y, z), but to derive *probabilities* for arbitrary whereabouts. When used like this, two aspects are implied: Ideally, the location of the current ground truth, and its vicinity, should be as likely as possible, whereas all other locations should be unlikely.

Besides visually inspecting figure 6.30, both requirements can be quantified by the Kullback-Leibler divergence (KLD), measuring the (dis)similarity between the Wi-Fi *density* (2.111), and ground truth. For the latter, any empiric distribution, modeling the likelihood of the ground truth and its vicinity, can be chosen. For experiments, a multivariate normal distribution with $4^2, 4^2, 0.75^2$ along the diagonal of Σ was used, to favor floor accuracy over (x, y). Both densities are compared by generating $50\,000$ random samples throughout the whole walkable area of each building. Similar to an error distance, the KLD yields a single scalar result, describing the dissimilarity between the density around the ground truth, and the density of potential whereabouts, based on the current Wi-Fi measurements. All resulting scalars are combined into

a CDF, containing all walks within a building, shown in figure 6.32. As can be seen, results are similar to figure 6.31, which depicted the CDF for the error in m. However, as the density chosen for the ground truth enforces a higher z-accuracy, oELD and oLD yield inferior results. Again, among all examined walks and buildings, oLDCPFi provided the best estimation, well suited for a computationally efficient evaluation within the recursive density estimation process.

6.3   Evaluation of Movement Models

After examining all sensors for the overall indoor localization and navigation system, and providing a first PDR example, the impact of adding a floorplan in combination with probabilistic movement prediction is examined. The discussions from chapter 3 and 5.2 already included synthetic examples for the propagation of each movement prediction density. Therefore, this section focuses on real-world setups, the amount of memory required for storing a spatial floorplan model, its probabilistic aspect, the capability of performing 3D estimations based on 2D sensor data, and including additional knowledge/sensors, using semantic information provided by the floorplan. As discussed in section 5.2, to perform probabilistic predictions on a navigation grid with random walks (cf. section 3.5.2), or on a navigation mesh (cf. section 3.6.2), sampling is required. All experiments thus focus on a recursive density estimation using the particle filter (see section 4.5), including transition, evaluation, estimation and resampling.

6.3.1   Spatial Floorplan Representation

Before evaluating the advantages of a floorplan-based probabilistic movement prediction, the quality of the two discussed spatial representations, the navigation grid and the navigation mesh, is briefly examined. The focus is on spatial accuracy for real-world scenarios, and required memory consumption, essential for use on embedded devices.

Memory Consumption   To determine memory consumptions for both models based on their configurable parameters, the three floorplans from section 6.1 are used, supplemented by four additional buildings, not examined in terms of conducted walks, but concerning their floorplan:

UAH – is the *University of Alcalá de Henares'* polytechnic building, with four floors and $130\,\mathrm{m}$ to $130\,\mathrm{m}$ in size, where the presented system participated in the *IPIN 2016 Indoor Localization Competition*, which required the floorplan. One of its floors is shown in figure A.1.

CAR – also belongs to the *University of Alcalá de Henares*, was part of the *IPIN 2016 Indoor Localization Competition*, and has one floor, $65\,\mathrm{m}$ to $40\,\mathrm{m}$ in size, surrounded by a large outdoor area, shown in figure A.2.

| | Navigation Grid | | | | Navigation Mesh | | | |
	20 cm	30 cm	40 cm	50 cm	Triangles	Quads	Pentagons	Hexagons
SHL	518 k	228 k	128 k	82 k	4164	2527	2187	2035
Museum 1	96 k	42 k	23 k	14 k	1852	1155	980	903
Museum 2	83 k	33 k	17 k	11 k	857	531	448	411
UAH	955 k	424 k	238 k	152 k	3206	2035	1788	1662
CAR	105 k	46 k	25 k	16 k	801	479	424	387
Townhall	182 k	80 k	45 k	28 k	2876	1806	1542	1425
Museum 3	76 k	34 k	19 k	12 k	1223	748	633	581

Table 6.15: Number of vertices/primitives required to describe building floorplans, when using a navigation grid with varying grid size gs, or a navigation mesh with different primitives.

| | Navigation Grid | | | | Navigation Mesh | | | |
	20 cm	30 cm	40 cm	50 cm	Triangles	Quads	Pentagons	Hexagons
SHL	30 MiB	13 MiB	7 MiB	5 MiB	248 KiB	191 KiB	172 KiB	164 KiB
Museum 2	5 MiB	2 MiB	1 MiB	636 KiB	51 KiB	40 KiB	35 KiB	33 KiB
Museum 1	5 MiB	2 MiB	1 MiB	811 KiB	110 KiB	87 KiB	77 KiB	73 KiB
UAH	55 MiB	24 MiB	14 MiB	9 MiB	191 KiB	155 KiB	141 KiB	134 KiB
CAR	6 MiB	3 MiB	1 MiB	930 KiB	48 KiB	36 KiB	33 KiB	31 KiB
Townhall	10 MiB	5 MiB	3 MiB	2 MiB	171 KiB	136 KiB	121 KiB	115 KiB
Museum 3	4 MiB	2 MiB	1 MiB	687 KiB	73 KiB	57 KiB	51 KiB	48 KiB

Table 6.16: Memory required to store building floorplans, when using a navigation grid with varying grid size gs, or a navigation mesh with different primitives.

Townhall – is the townhall of Würzburg, with four floors, but only two of them were modeled, approximately $60\,m$ to $100\,m$ in size, including a large courtyard for the first floor.

Museum 3 – models the *Deutsches Hutmuseum* in Lindenberg, which is a museum with five floors, approximately $40\,m$ to $20\,m$ in size, shown in figure A.3.

For the navigation grid, the memory consumption directly depends on the number of vertices and edges needed to model the building's floorplan, which in turn depend on its size and number of floors, and the chosen grid size, where smaller values yield a better spatial quality at the cost of memory. For the navigation mesh, there is no direct correspondence between the size of each floor and required memory, as the data structure is irregular, able to model large areas with a single primitive. The number of primitives depends on the architectural obstacles found within each floor, dividing larger primitives into multiple smaller ones. Similarly, the type of primitive (triangle, quad, ...) also affects the required number, and thus memory.

Setting-dependent results for both models are shown in table 6.15 and table 6.16. The largest among all examined buildings, UAH, requires most vertices for the navigation grid, yielding a memory requirement of $55\,MiB$ for $gs = 20\,cm$. When using the navigation mesh, the smaller SHL requires more primitives and memory. This is due to architectural differences, with SHL
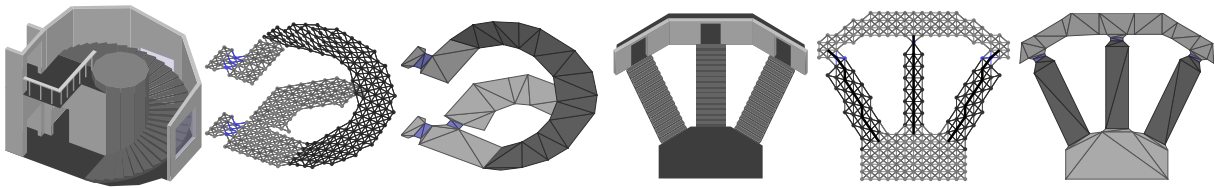
Figure 6.33: Comparison of navigation grid (gs = 40 cm) and navigation mesh, for advanced floorplans. While both produce viable results for complex circular stairs (left), elements not aligned to multiples of 45°, like two of the three stairs in the right setup, indicate potential drawbacks of the navigation grid.

consisting of many small rooms, whereas UAH has larger rooms, thus requiring less primitives. Independent of such differences, the navigation mesh consumed less than $1\,\mathrm{MiB}$ for all examined buildings. For the navigation grid, this is only possible when using large grid sizes of $50\,\mathrm{cm}$ and above, providing only a coarse representation of the building. Nevertheless, even a gs = 20 cm navigation grid of a large building is still manageable by a modern smartphone.

However, the numbers from table 6.16 refer to the smallest possible amount, storing only vertices and edges, and in a very dense manner. As soon as additional attributes are involved, such as shortest path estimations (see section 3.5.3 and 3.6.3), or pre-computed Wi-Fi signal strengths (cf. section 5.4), numbers might exceed the manageable range: Storing the 20 strongest access points for each vertex would require at least $40\,\mathrm{byte}$ per vertex (id and RSSI), adding additional $40\,\mathrm{MiB}$ to the UAH. Which model is applicable for a specific use case, thus strongly depends on the building's architecture, and additional performance/quality requirements.

Even though triangles require slightly more memory than other primitives (cf. table 6.16), they allow for interpolation and fast inclusion checks, and thus are preferred.

Spatial Quality   Besides differences in memory consumption, the spatial representations also vary in quality. While the navigation mesh is considered general purpose, the navigation grid's regular structure is mainly suited for axis-aligned floorplans, with obstacles oriented in multiples of 45° or, even better, 90°, discussed in section 3.5.1. While this requirement holds true for many modern buildings, like SHL, it does not for older ones, or more complex architecture.

Figure 6.33 used complex architecture to compare a navigation grid with gs = 40 cm against a navigation mesh with triangles, constructed by the algorithms discussed in section 3.5.1 and 3.6.1, respectively. For a wide circular stair (left), both produce results that appear viable. Even though the navigation grid's exterior is more discrete, compared to the navigation mesh, this is still within limits, and the stair is covered by a multitude of walkable nodes and edges, even when using gs = 40 cm. The semantic information, indicated by different colors (gray, black, blue), e.g. required for reducing the pedestrian's step length throughout the stair, also works for both spatial models. However, one of the three doors is not correctly labeled for the navigation

grid. This is due to the chosen spacing not yielding vertices within the door's center, but only directly before and behind it. This can be mitigated by using a smaller grid size.

Within the right half, three stairs with varying orientation and adjacent doors are shown. While the axis-aligned stair in the center is well suited for the regular navigation grid, the two others are not. Even though visually fine, the direction of edges is not ideal for predicting movements along the stair, potentially yielding the depicted zig-zag pattern. The same applies to the circular stair on the left, often not providing edges that follow the ideal path. Such cases, where the to-be-predicted walking direction does not match with the direction of edges, can not be addressed by using smaller grid sizes, and are a general problem of the navigation grid.

Furthermore, figure 6.33 indicates a significant number of nodes and edges required for the navigation grid, even when using a large grid size of $\text{gs} = 40\,\text{cm}$. In contrast, the navigation mesh is able to model fine details, adjusts to different alignments, while combining large connected areas into a few primitives. From the viewpoint of both, memory consumption and spatial quality, the navigation mesh thus is more promising than the navigation grid.

6.3.2 Navigation

One benefit of including a floorplan, discussed in section 3.5.3, is navigation. Not only for guiding the pedestrian towards a desired destination, but also to favor movements approaching this location, increasing the quality of the overall system. The impact is examined using simulations of random walks along a navigation grid, using algorithm 1. It is started at a given location, and executed until the given destination is reached. The algorithm is thus executed with an unlimited walking distance d_{walk}. Only the destination-approaching metric (3.32) is used to determine the probability for each edge, that is, the heading is completely unconstrained. Thus, edges are solely chosen on whether they approach the destination, or not. The weight of the edges used within Dijkstra's algorithm is given by either (3.29) or (3.31), depending on whether walls are to be avoided, or not. For a robust estimation of the random walk's behavior, this simulation is repeated 5000 times. Results are visualized using a heat map, indicating how often every vertex of the graph was visited among all repetitions. Cold colors (blue) denote fewer visits than warm colors (red). In case of no color, the vertex has not been visited at all.

Figure 6.34 shows the impact of different values for κ_{dest}, indicated by the heatmap of 5000 random walks through a synthetic maze. κ_{dest} affects the probability of favoring edges approaching the destination over edges that depart from it. A value of $\kappa_{\text{dest}} = 0.50$, shown in the left of figure 6.34, implies that all edges are equally important, and the destination does not affect the probability for taking an edge at all. Such a truly random walk will require a significant amount of time, until the requested destination is reached by chance. As can be
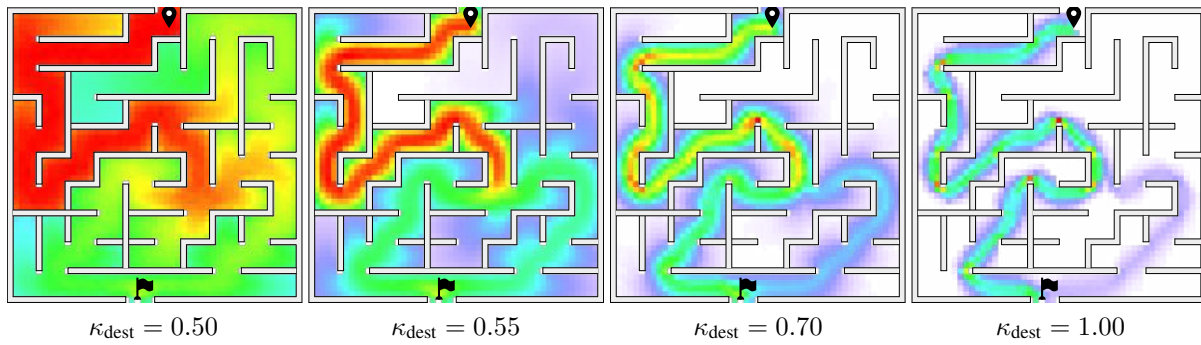
Figure 6.34: Impact of κ_{dest} on random walks approaching the destination via (3.32), weighted by (3.29) running simulation algorithm 1 along a $\text{gs} = 20\,\text{cm}$ grid. The heatmap denotes how often each vertex of the grid was visited during 5000 random walks, starting from the upper center, each running without distance limitation, until reaching the destination in the lower center.

seen, the first few meters of the walk share the same color, indicating a similar number of visits. As soon as the maze allows for more than one walking direction, the colors get colder, and the random walks split into several directions. Approaching the destination, colors are getting warmer again, as all random walks have to reach the destination. Being accessible from two sides, and the option to walk backwards, colors surrounding the destination are colder than the ones around the origin. When the heuristic is slightly increased to $\kappa_{\text{dest}} = 0.55$, the effect of favoring destination-approaching edges is already significant. Most locations within the maze are rarely visited (blue), or not visited at all (white). Furthermore, when the maze splits into three potential walking directions, only two of them are considered, as the third one is a dead end. The impact is even more pronounced for $\kappa_{\text{dest}} = 0.70$, where most regions remain unvisited, and the left of the two valid paths is frequented more often, as it is slightly shorter than the alternative on the right. For $\kappa_{\text{dest}} = 1.00$, the simulation ignores all edges departing from the destination, leaving very narrow trails, with some vertices visited during almost every of the 5000 repetitions, shown as red spots. While higher values for κ_{dest} yield a more direct approach of the destination, they allow for less deviations from the shortest path, ignoring alternatives, and the pedestrian not following the system's recommendation. Estimating the correct value is thus an empiric choice, as tradeoff between a closely directed route, and allowed variation.

As can be seen in figure 6.34, all routes closely adhere to walls, slightly reducing the distance of the shortest path. Discussed in section 3.5.3, this is atypical for pedestrian walking behavior. By artificially increasing the distance for edges near obstacles, the shortest path becomes more realistic. The impact of this adjustment is shown in figure 6.35. When obstacles are not considered, the result is unnaturally close to all obstacles, and only the left of two potential routes is used, as it is several meters shorter than using the stair on the right. Artificially adjusting the distance between edges using (3.31), the second route along the right stair is more attractive,

$\kappa_{\text{dest}} = 0.75$, not avoiding walls (3.29) $\kappa_{\text{dest}} = 0.75$, avoiding walls via (3.31)
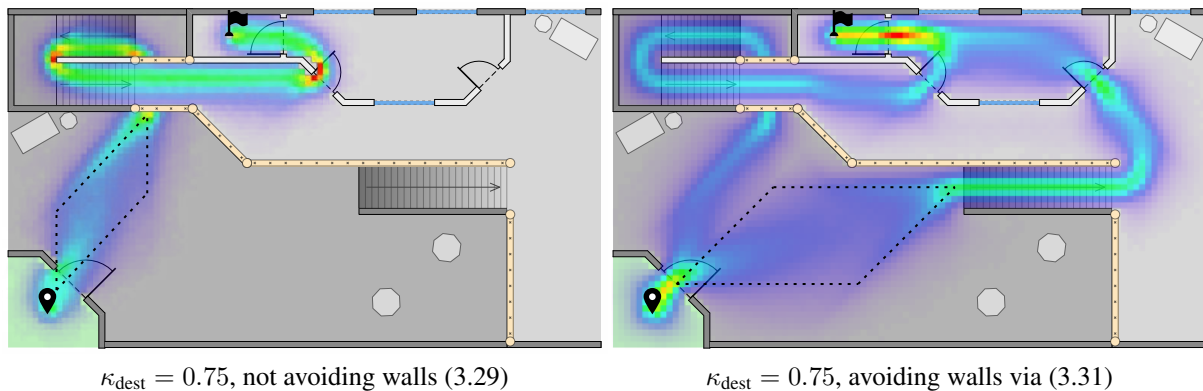
Figure 6.35: Impact of the wall avoiding metric (3.30) on random walks approaching the destination via (3.32), weighted by (3.31) running simulation algorithm 1 along a $\text{gs} = 20\,\text{cm}$ grid. The heatmap denotes how often each vertex of the grid was visited during $20\,000$ random walks starting in the lower left, each running without distance limitation, until reaching the destination in the upper center.

being a few centimeters shorter. Furthermore, both routes are more natural, keeping some distance towards obstacles, walking through the centers of doors and narrow passages. Yet, the negative impact of the discrete regular grid is clearly visible. For segments of the shortest path, that have a direction which is not a multiple of $45°$, there often is no single best choice that yields the shortest path, and several equidistant ways connect the segment's ending points. This is highlighted in figure 6.35, where dashed lines denote equidistant connections between two points. The heatmap around such segments appears much wider, covering all potential paths, indicated by larger cold (blue) regions. When the direction is a multiple of $45°$, a single shortest connection exists, shown by all narrow warm (green/red) streams within the figure. This effect could be addressed by using more than eight adjacent neighbors, also considering vertices further away to introduce new walking directions, such as $22.5°$. However, as those edges are even longer than $\sqrt{2}\,\text{gs}$, the distance error discussed in section 3.5.3 becomes more pronounced, and memory requirements are increasing, potentially imposing issues on embedded use.

While the navigation mesh does not suffer from these discrete limitations, it potentially requires more computations to include the destination information correctly. Discussed in section 3.6, the most efficient movement prediction for the mesh directly calculates a potential transition based on distance and heading from the origin, including some noise (3.35). In contrast to probabilistic random walks along a navigation grid, the only way to also include a navigation likelihood, is by using particles, and adjusting their weight accordingly. However, as the result of (3.35) is very constrained, covering only a small area, the resulting density might lack particles in areas that are likely in terms of navigation. This is a typical limitation of the particle filter, discussed in section 4.5. The posterior is modeled by both, placement and weight of the particles. If the placement (proposal distribution) does not cover areas likely in terms
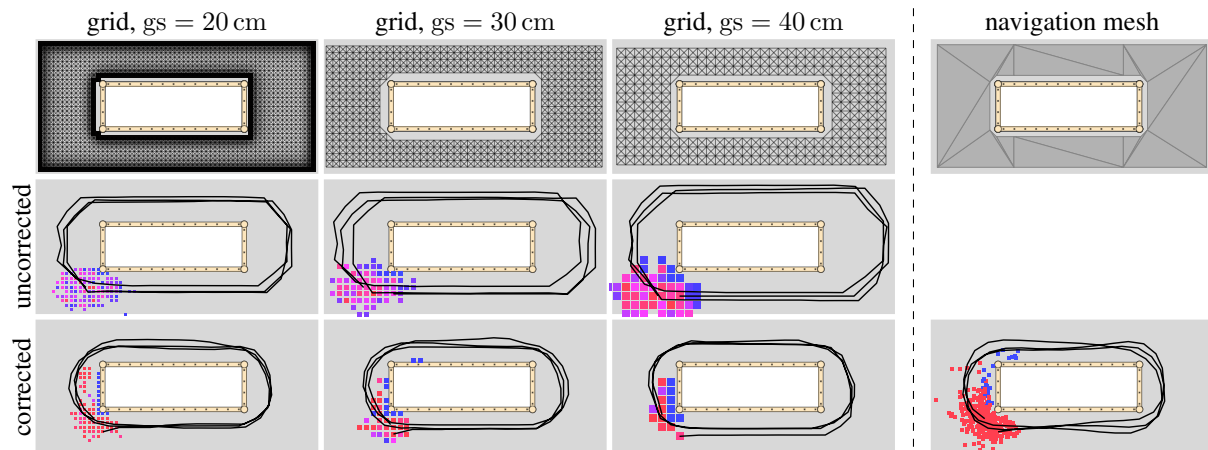
Figure 6.36: Comparison of navigation grid and navigation mesh, for one of the PDR examples from section 6.2.4 (see figure 6.16). For the navigation grid, different grid sizes gs were used, also examining the impact of distance and heading error compensation (cf. algorithm 1).

of weight, the result is not optimal. This can be mitigated by using the full sampling approach shown in figure 3.24, covering large areas with samples, hereafter weighting them based on *all* metrics, that is, matching distance, heading, and navigation. While this variant ensures that the whole area of importance is sampled, it requires more samples and thus computations to do so, unsuited for use on embedded devices. When navigation is the only prediction constraint, as in figure 6.35, full sampling is the only option, as no limiting heading information is available. As can be seen, in terms of flexibility, the navigation mesh is thus more constrained than random walks along a navigation grid.

6.3.3   Floorplan-Based Probabilistic Pedestrian Dead Reckoning

Advantages and disadvantages of including a building's floorplan and probabilistic movement prediction are examined for the pedestrian dead reckoning setups from section 6.2.4. Although regionally limited, they include every essential behavior, and allow for focusing on the important aspects, without loss of generality. The floorplan not only prevents impossible walks, it also allows for 3D estimations, and its semantic information enables including probabilistic assumptions, additional prior knowledge, as well as new smartphone sensors. Besides step-detection, turn-detection, and absolute headings, which are typical for PDR, activity-detection and navigational knowledge can now be considered as well. Additionally, differences between the two presented spatial models are examined, emphasizing distinctions between discrete random walks along a navigation grid and continuous estimations on the navigation mesh. All of the following experiments refer to a probabilistic simulation using the particle filter.

Graph and Navigation Mesh A comparison, of the navigation grid's and navigation mesh's behavior, is shown in figure 6.36. Its results are based on one of the previous PDR examples (see section 6.2.4), walking four times around an obstacle. Similar to earlier, initial whereabouts and heading are well-known. A transition is performed whenever a step was detected, and proceeds with $\mu_{\text{step}} = 70\,\text{cm}$ into the direction of the heading updated by the gyroscope. In contrast to earlier, navigation grid and navigation mesh now allow for a probabilistic transition, each using 1000 particles to include an uncertainty in step length $\sigma_{\text{step}} = 10\,\text{cm}$ and turning $\sigma_{\text{turn}} = 1°$, both applied for every filter update, also preventing impossible movements, by down-weighting particles that encountered an obstacle, or would represent an impossible transition.

For both models, the fastest calculable transition, well-suited for smartphone use, was chosen: Within the navigation mesh, each transition is determined by directly calculating the destination based on walking distance and estimated heading (3.35), including some uncertainty for both (3.37). If a calculated destination is reachable (contained within an adjacent triangle), it is accepted, otherwise the particle's weight is reduced to zero (see section 5.2.3). Within the navigation grid, edges are weighted based on whether their direction matches with the estimated heading (3.25), and drawn randomly, until the walking distance of d_{walk} is reached. Discussed in section 3.5.2, this approach causes discretization errors based on the graph's grid size, and multiples of $45°$ for walkable edges, requiring compensation, introduced in algorithm 1.

Figure 6.36 depicts the necessity for this error compensation. Without it, walking distances are longer than requested, dependent on the underlying grid size, visible as increased radius within the three depicted results. Likewise, the $45°$-structure becomes apparent, with the estimated path resembling an octagon. Yet, the overall results, when not using error compensation, appear consistent in their size and shape. This is due to the discretization, preventing minor angular changes. When all conducted turns are multiples of $45°$, this can be considered a benefit. In general, however, this suppresses minor changes, or discretizes them into larger ones, causing issues for walks and buildings not aligned to multiples of $45°$. Also visualized in figure 6.36, compensation from algorithm 1 addresses this issue, reducing the size of the estimated path, and producing correctly rounded turns, even for larger grid sizes. The right shows the same result when using the navigation mesh. It is similar to the navigation grid with error compensation, but shows slightly more variation, due to its continuous approach.

Discussed in section 5.2.3, detecting impossible walks is one important aspect the floorplan model must provide. For the navigation mesh this is identified when the calculated destination is not covered by a triangle, or its triangle is too far from the starting point. For random walks along the navigation grid, however, this is more complex (see section 3.5.2). As random walks follow adjacent edges, drawn by weighted random sampling, they always produce a prediction, reachable from the starting point. To prevent particles from getting stuck in front of an obstacle,
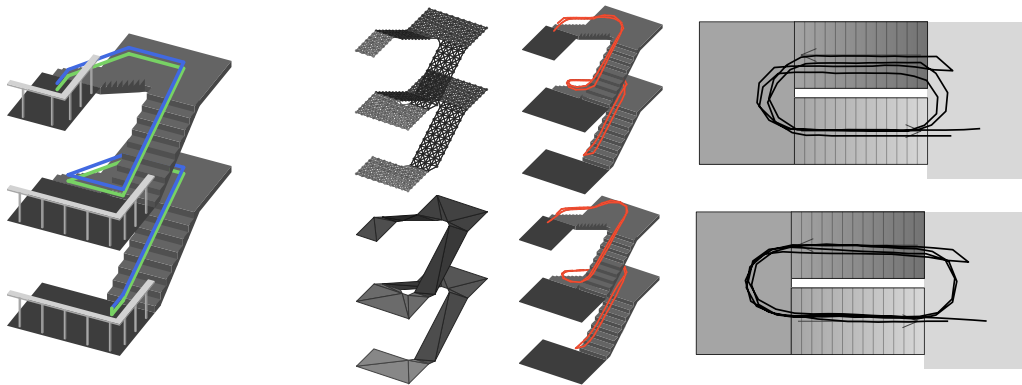
Figure 6.37: PDR with step and turn-detection, for the data from figure 6.17, now using a probabilistic estimation, including the building's floorplan based on navigation grid (top) and navigation mesh (bottom). The result for both spatial models is similar, corrects gyroscope drift, and enables 3D estimations.

they must be identified. For the depicted example this was addressed by the obstacle-prevention (3.30). All nodes with less than eight neighbors are flagged, and particles residing at such nodes are down-weighted. In figure 6.36, those nodes are shown in black for $gs = 20\,cm$. The particle down-weighting is indicated by their color, shown for $t = 54\,s$, where blue indicates a low, and red a high weight. As can be seen for both, navigation grid and navigation mesh, particles near obstacles are unlikely, compared to others in their vicinity. This prevents impossible movements from affecting the estimation, and resampling eventually replaces such particles by more likely ones, ensuring stability throughout the walk. Comparing the depicted results against the raw PDR from figure 6.16, the advantages of including a floorplan become apparent.

3D Component   Another benefit of including the floorplan is the ability to allow 3D walks, modeling the walkable surface by some spatial structure, which not only prevents impossible walks, but also introduces possible movements in z. This is shown for another result from the previous PDR examples (figure 6.17), walking downwards by two stories within a stairwell, and then upwards again. As can be seen in figure 6.36, both, navigation grid and navigation mesh, allow for 3D walks, even when only relative 2D sensors (step and turn-detection) are used. For the navigation mesh, each particle, not shown within the figure, is moved in the 2D plane, by calculating a potential destination, hereafter determining its z-component based on barycentric interpolation (cf. section 3.6.2). For the navigation grid, the random walk follows adjacent edges, updating the particles z-component with every edge. The result of both spatial models is similar. Shown in the 2D top view on the right, the floorplan not only adds the z-component, it also addresses drifts, discussed and shown previously for the raw PDR in figure 6.17.

Including Additional Knowledge   Discussed in chapter 3 and 5, the quality of movement prediction increases with every added information (origin, step length, heading, activity, des-
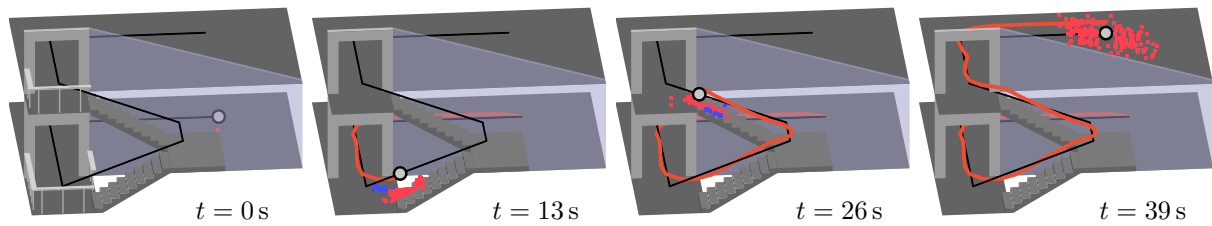
Figure 6.38: Probabilistic PDR based on the building's floorplan, with location and heading of the initial state q_0 well known, hereafter using step and turn-detection with the navigation mesh.
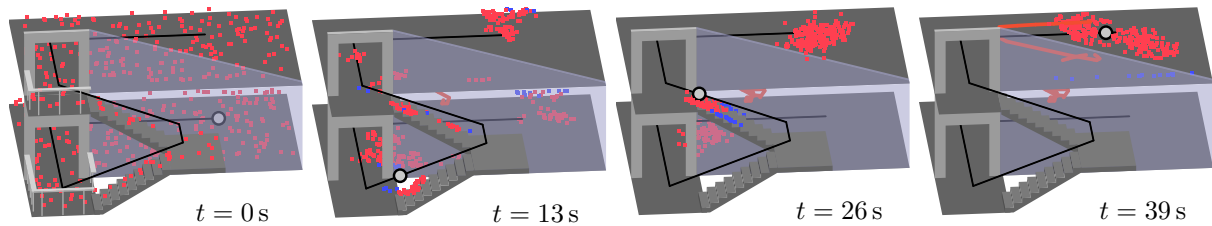
Figure 6.39: Probabilistic PDR based on the building's floorplan, with location and heading of the initial state q_0 unknown, modeled by a uniform distribution, hereafter using step and turn-detection.

tination). Predictions without distance and heading information are barely useful, especially when performed several times in sequence. Therefore, step-detection and turn-detection are assumed to be available, provided by the accelerometer and gyroscope installed within every modern smartphone. The impact of including additional knowledge besides these two, is examined using another short walk, conducted with the LG G6. It consists of walking along a hallway, opening the door into a stairwell, moving up by one floor, leaving the stairwell through another door, walking along the hallway. This setup examines the impact of including a known origin, activity-detection, and navigation, as the walk represents a shortest path towards the destination. To improve 2D visualizations, floors and stairwell are separated by a blue glass facade.

Figure 6.38 shows the result when using only step-detection and turn-detection based on the navigation mesh, with the initial whereabouts and heading well known. As can be seen, the floorplan allows for an almost ideal result, entering the stairwell, walking along the stair's center, reaching the next floor, and the destination. The result for the navigation grid is similar.

Figure 6.39 depicts the same experiment, but with the initial whereabouts and heading unknown, modeled by a uniform distribution. As the shown floorplan is very limited, most of the initial particles eventually yield incorrect movements, removed by resampling, leaving only a few likely spots after 13 s. This effect increases when the pedestrian starts to walk along the stair, causing impossible turns for many of the particles. Having reached the end of the stair, the density converges, concentrating most of the particles near the actual whereabouts. The chance for converging strongly depends on the conducted walk, as well as the layout of the floorplan. Yet, it can be increased by adding other sensors besides step and turn-detection.
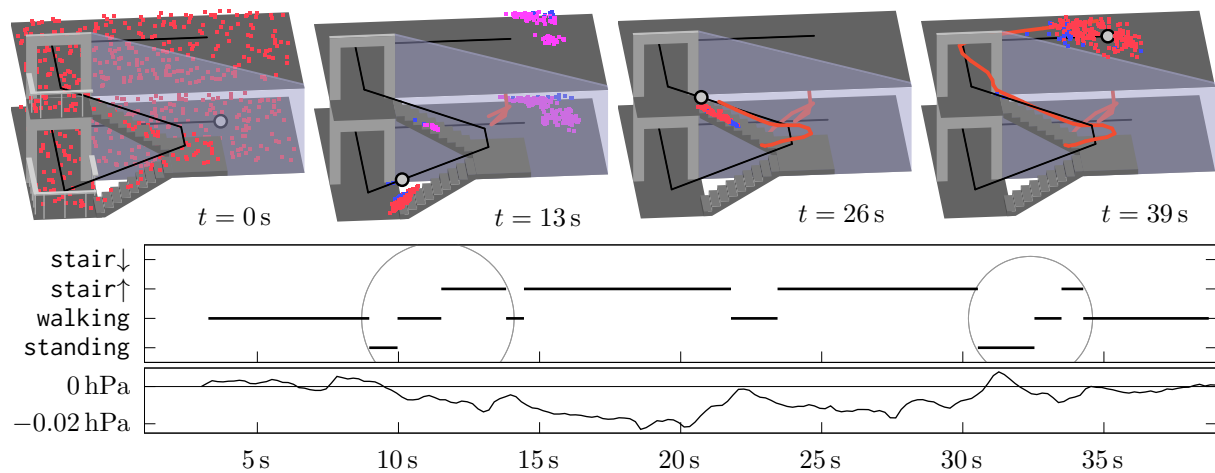
Figure 6.40: Probabilistic PDR based on the building's floorplan, with location and heading of the initial state q_0 unknown, modeled by a uniform distribution, hereafter using step, turn, and activity-detection by the Bayes classifier (see section 2.6 and 6.2.6), to distinguish between walking and taking stairs. The stair↑ at $12\,\mathrm{s}$ is a false positive, caused by opening the door into the stairwell. After some delay, the activity is correct at $15\,\mathrm{s}$, removing all non-stair particles until $26\,\mathrm{s}$. At $34\,\mathrm{s}$, another false positive occurs, again caused by opening the door. The change in atmospheric pressure within $500\,\mathrm{ms}$ is shown below.

Mentioned in section 2.6, activity-detection is one component, providing some sort of absolute location information, e.g. by focusing solely on stairs, when the corresponding activity was detected. Figure 6.40 shows the result for unknown initial whereabouts, when including activity-detection based on the Bayes classifier (cf. section 2.6 and 6.2.6), using the semantic information from the navigation mesh and (2.80) with $\kappa_{\mathrm{match}} = 0.6$, slightly favoring all transitions matching with the currently detected activity. At $9\,\mathrm{s}$, the pedestrian rests to open the door into the stairwell, detected as standing. Opening causes a change in atmospheric pressure, which yields an incorrect stair↑ at $12\,\mathrm{s}$, instead of walking. After some delay, at $15\,\mathrm{s}$, the activity is correct, and the density slowly starts to concentrate along the stair. When leaving the stairwell at $30\,\mathrm{s}$, there is another false detection, again caused by opening a door to leave the stairwell. For this example, including the activity causes the density to converge a few seconds after taking the stairs, representing a major benefit. As can be seen from the incorrectly detected activities, and the underlying change in atmospheric pressure within a $500\,\mathrm{ms}$ window (see section 6.2.6), the barometer itself is rather erroneous, affected by opening the door into or out of the stairwell. When Wi-Fi is used as well, providing an absolute z-estimation, it is advisable to omit the barometer as z-indicator, and refer to activity detection, to prevent negative side effects. While the barometer is useful for a coarse z-estimation in buildings with numerous floors, Wi-Fi provides a similar quality (cf. section 6.2.7), and is available within far more devices.

In contrast to Wi-Fi, activity, or the barometer, knowledge on the pedestrian's desired destination does not directly exclude locations within the building, but is able to favor some of the
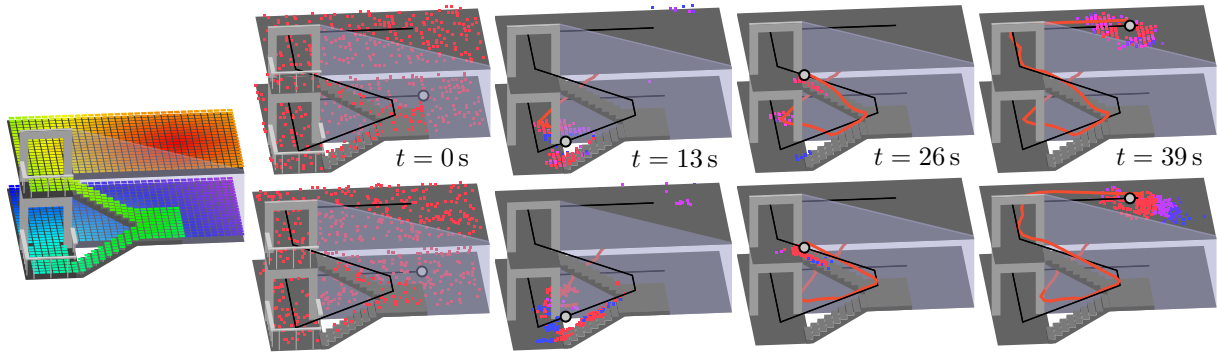
Figure 6.41: Probabilistic PDR based on the building's floorplan, with location and heading of the initial state q_0 unknown, modeled by a uniform distribution, hereafter using step-detection, turn-detection, and the pedestrian's desired destination, evaluating all potential moves using the navigation grid with (3.32) (top), and the navigation mesh (bottom), both with $\kappa_{\text{dest}} = 0.7$. The distance of each location towards the destination is shown in the left (far: blue, near: red). For both spatial models, at $t = 13\,\text{s}$ particles concentrate along or near the stairwell, quickly converging hereafter.

potential movements. For the example shown in figure 6.41, the destination is assumed on the second floor. The distance towards it is calculated once for every location within the building, as shown in the left. Starting from a uniform distribution, the same step and turn-detection as earlier is applied, but all movements approaching the destination are slightly favored by (3.32) with $\kappa_{\text{dest}} = 0.7$. While this still allows for movements departing from the destination, the density starts to concentrate, where several consecutive transitions approached the destination. For the examined walk, the result of the navigation grid (top) and navigation mesh (bottom) is similar. As can be seen, assuming the pedestrian to use the shortest path towards a desired destination can improve movement predictions, helping the density to converge.

Similar to navigation is the eCompass (see section 2.4.3), limiting potential walking directions, based on the absolute heading estimated from the magnetometer. Shown in section 6.2.4, architectural influences prevent using the compass directly as sole heading indicator. However, when working with relative heading adjustments based on the gyroscope, as earlier, and including the eCompass as a coarse heading evaluation, it can provide an additional benefit. In the following example, the eCompass is included by (2.66), with a width of $\pm 30°$. That is, all particles with a heading deviating less than $30°$ from the eCompass' indication receive the same weight, diminishing when the difference is above $30°$. In doing so, the eCompass limits only movements deviating significantly. While (2.21) does not sum to 1 for the range of $[0, 2\pi]$, the error is marginal and re-normalized by the particle filter. An example is depicted in figure 6.42, re-using the data from figure 6.16, walking four times around an obstacle, where the compass indicated only minor outliers. Starting from a uniform distribution with random heading, the evaluation removes all particles pointing in the opposite direction. Even though the final den-
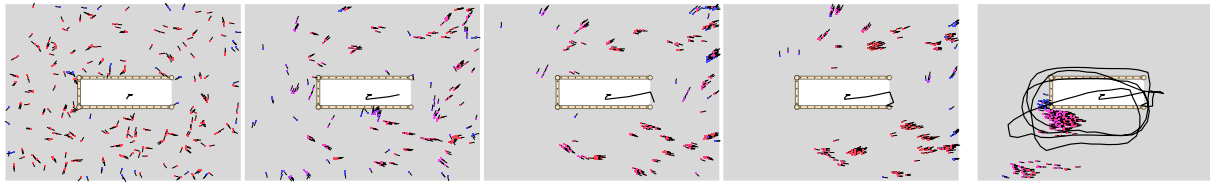
Figure 6.42: Probabilistic PDR based on the building's floorplan, with location and heading of the initial state q_0 unknown, modeled by a uniform distribution, hereafter using step-detection, turn-detection and absolute heading from the eCompass based on (2.64).

sity did not fully converge, showing a second mode (lower left particles), and thus an invalid weighted-average estimation, the ground truth is covered by the majority of the density.

Figure 6.43 depicts another result of probabilistic PDR, including the building's floorplan for the data from figure 6.18. Seen on the left, when the origin is known, the floorplan corrects the drift of the gyroscope and yields an almost ideal result. Additionally including the eCompass as evaluation ((2.64) or (2.66)), is shown by two additional estimations: One using a normal distribution (ND) with $\sigma_{\mathrm{comp}} = 45°$, and one using the distribution from (2.21) (RD) at a width of $\pm 30°$. The normal distribution keeps the mean of the eCompass, which was erroneous for this walk (see figure 6.18), forcing the density towards the right, corrected by the floorplan, preventing some impossible walks. When using (2.21) as distribution, there is no clear mean, as all particles with approximately matching heading receive the same weight, mitigating the drag onto the mean. The right half of the figure shows the result for an unknown origin, modeled by a uniform distribution. When the eCompass is unavailable, the density converged a few seconds before reaching the destination (lower left). With the eCompass enabled, convergence is improved, although it still mistakenly enters the room in the upper left. The effect of using a normal distribution for evaluating the eCompass is similar to the left half. Since the mean is preserved, the estimation is dragged to the right. This is partly compensated by the floorplan, which prevents resulting impossible walks.

While the previous results were viable, to prevent negative effects of architectural influences (cf. figure 6.17), further increasing σ or temporally disabling the eCompass can be required.

6.3.4   Limitations

One main limitation of the presented movement predictions is convergence. This not solely relates to the prediction model or the underlying floorplan, but also stems from the required simulation: As discussed earlier, both, the navigation grid and the navigation mesh, do not directly provide densities, but approximate one based on samples. Therefore, multiple samples are required, with the quality directly proportional to their number (cf. section 4.5). Indicated
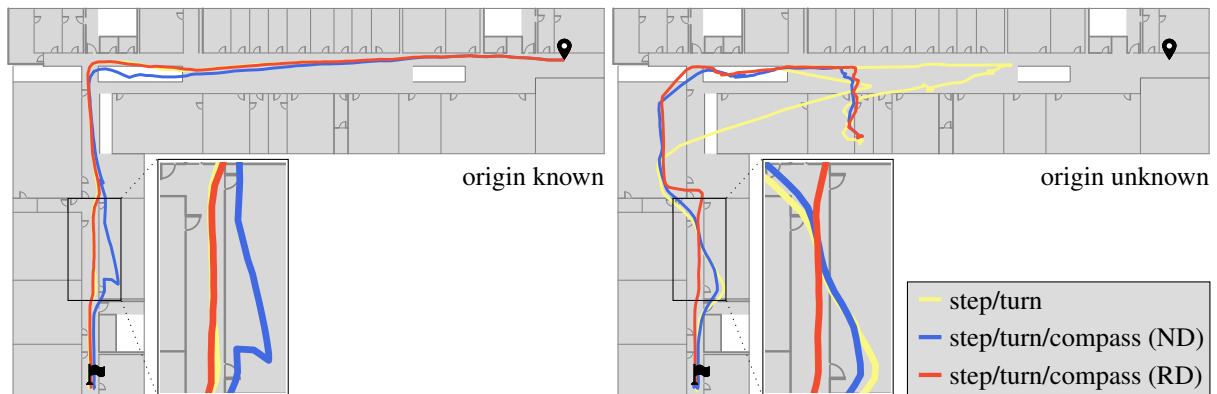
Figure 6.43: Probabilistic PDR based on the building's floorplan, combined with step-detection, turn-detection and absolute heading from the eCompass using (2.64) with a normal distribution (ND), and using (2.66) with (2.21) (RD). Once with a known origin (left), and with an unknown origin (right).

by previous experiments, the floorplan, and additionally available information also affect this quality. Especially when starting from a uniform prior, the chance for a limited sample-set to contain the correct origin and heading strongly depends on the size of the floorplan.

The impact of certain aspects is analyzed by executing a walk-simulation multiple times, using pseudo-random numbers based on a *seed*. For each of the repetitions, a different seed is used, ensuring a varying pseudo-random process, and thus another result. Likewise, consecutive tests may refer to the same seed, to produce the same random numbers, e.g. for testing solely the influence of the number of particles, keeping all other random decisions.

An example is visualized in the left of figure 6.44. When starting from a uniform prior with 1000 particles, the chance for convergence is poor, depicted by the results for two different seeds, shown in black and red. While the black one walked in the opposite direction, the red one was closer to the actual solution, but did still not converge. When increasing the number of particles, the results (blue and green) converged, reaching the actual destination.

A similar problem arises from invalid assumptions made during the movement prediction. As discussed in section 2.4.1, while detecting pedestrian steps is easy, estimating their size is not, especially when the phone is held upfront. Thus, apart from stairs, where the tread-size is known, a constant step length of $\mu_{\text{step}} \approx 70\,\text{cm}$ is assumed, including $\sigma_{\text{step}} = 10\,\text{cm}$ for uncertainty. The impact of adjusting this assumption, which is similar to the pedestrian changing the step length, is shown in the right of figure 6.44. While the larger step length (red) converges eventually, the shorter one (blue) does not, as all particles get trapped within a room.

A related problem occurs for many of the real-world walks, conducted in Museum 1 or Museum 2. Here, the pedestrian often rested in front of exhibits, slowly turning to watch multiple items. This *looking-around* typically involves minor foot movement, to turn the body, misclassified as steps, causing the movement prediction to update the estimated location. Depending on
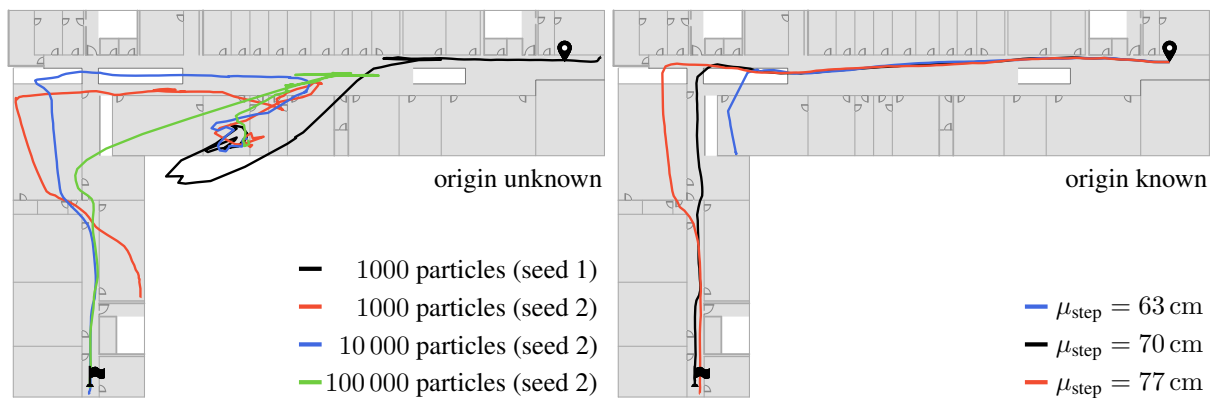
Figure 6.44: Limitations of probabilistic PDR based on the building's floorplan. When using only step and turn-detection, starting from a uniform distribution (left), the chance for convergence is poor. For 1000 particles, most results are like the black and the red path. For $\geq 10\,000$ particles (blue,green), the chance is increased. When starting from a known origin (right), invalid sensor readings, or false assumptions, can lead to similar problems, depicted by using three different step sizes for the same walk.

the surrounding architecture, and the pedestrian's behavior, this can lead to the whole density getting trapped, walking against obstacles. This e.g. happens within walk C2. After walking a few seconds, the pedestrian watches exhibits, slowly turning, with the detected steps causing the density to approach a corner, unable to move any further, indicated by the red path in figure 6.45e. This can often be mitigated by using the probabilistic step-detection (2.9) and (2.31), instead, as the magntitude of the accelerometer is significantly smaller when turning on the spot. While the binary yes/no step-detection got stuck after a few seconds of walking, the probabilistic variant (blue path) converged, with an average error of $\approx 1.5\,\mathrm{m}$.

A similar problem occurs for walk A4, after finishing most of the walk without issues. During climbing stairs, the step-detection indicates a false positive, yielding almost the whole density to leave the stairwell through an adjacent door, shown in figure 6.45c and 6.45d. While a few particles remained on the stair, taking the correct path, they collided with the stair's boundary, caused by a slight drift of the gyroscope, and were lost during resampling.

For walk A1, shown in figure 6.45a, the density does not get stuck, but does not converge either. When re-entering the building, some of the (red) particles use the correct path, while others refer to an adjacent stair. The density then splits into two modes, and remains like this for a few seconds (green particles). With the following right turn, many of the particles within the building collide with the floorplan, while the outdoor-ones can freely proceed without obstacles. Even though a few (cyan) reached the actual destination, the majority remained outside, causing the weighted average estimation to indicate an incorrect path.

For walk A2, shown in figure 6.45b, obstacles within the outdoor area caused the red density to split into two modes. Without other obstacles nearby, this yields a large and growing uncer-

(a) walk A1 - top

(b) walk A2 - top

(c) walk A4 - 3D

(d) walk A4 - top

(e) walk C2 - top
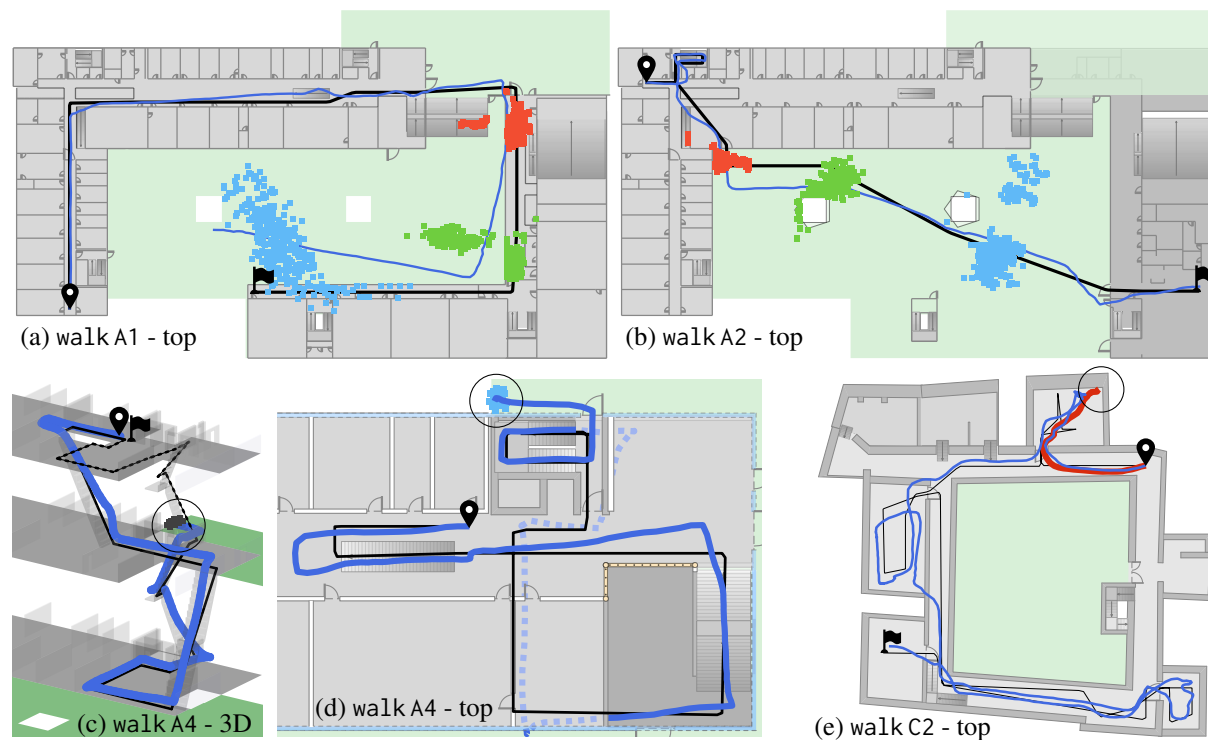
Figure 6.45: Issues of probabilistic PDR, for some of the real-world scenarios from section 6.1. Black and blue lines denote ground truth and estimation. Detailed descriptions are provided within the text.

tainty. The upper mode eventually vanishes, colliding with the building's exterior. The lower one enters the building through two doors, one of them connected to the destination.

Thus, matching with prior discussions (cf. section 5.2), the floorplan yields both, advantages and disadvantages. Without, 3D estimations can not be performed, at least, not when using only 2D sensors. Similarly, when not limiting impossible walks, a uniform prior can not converge without additional absolute sensors, such as Wi-Fi. However, when assumptions or sensor data are invalid, these benefits become a drawback and the overall density might get stuck, unable to recover, requiring for a restart or similar techniques (see section 5.2).

To get an impression on actual numbers, two of the previously discussed PDR walks were simulated 100 times, for each of which two metrics were determined: The first denotes if the density got stuck, which is the case when all particles received a weight of 0, denoting they were unable to move. The second identifies whether the final location estimation was valid, matching with the ground truth, determined using a threshold of 1.5 m. Results for the walk, using a stairwell to change the floor, are shown in table 6.17. The longer walk along a floor's hallway is listed in table 6.18. By comparing both, various aspects can be recognized. As expected, the number of particles is important, affecting whether getting stuck, or yielding a valid estimation. In all cases, the resulting quality was directly proportional to the number

| | 100 particles | | 250 particles | | 500 particles | | 1000 particles | |
	valid	stuck	valid	stuck	valid	stuck	valid	stuck
known origin	100 %	0 %						
uniform	13 %	66 %	28 %	31 %	41 %	8 %	67 %	1 %
uniform, activity	15 %	69 %	34 %	28 %	47 %	6 %	70 %	1 %
uniform, navigation	22 %	65 %	47 %	30 %	80 %	5 %	91 %	1 %

Table 6.17: Convergence results for the data from figures 6.38, 6.39, 6.40, 6.41 using 100 simulations. The metrics `stuck` and `valid` are explained within the text.

| | 1000 particles | | 2500 particles | | 5000 particles | | 10000 particles | |
	valid	stuck	valid	stuck	valid	stuck	valid	stuck
known origin	100 %	0 %						
known origin, compass	100 %	0 %						
uniform	5 %	95 %	10 %	90 %	33 %	67 %	50 %	50 %
uniform, compass	41 %	59 %	73 %	27 %	96 %	4 %	100 %	0 %
uniform, navigation	61 %	39 %	89 %	11 %	98 %	2 %	100 %	0 %

Table 6.18: Convergence results for the data from figure 6.43, using 100 simulations.

of particles used. Also as expected, the larger floorplan requires more particles, at least, when starting from an uniform prior. The listed values also denote the impact of additional information, like a known origin, activity, eCompass or known destination. In all cases, this information increased the correctness of the final result, and reduced the risk of getting stuck.

Between both walks, there is a noteworthy difference. For the longer one (table 6.18), all simulations that did not get stuck, actually returned the correct location estimation for the final position, which is not the case for the shorter walk (table 6.17). This behavior is induced by the floorplan: For the shorter walk, the destination on the upper floor is surrounded by free space, for the longer walk, it represents a dead end. Thus, the floorplan concentrates the density around the desired destination, increasing the chance for matching with the ground truth.

As denoted by the presented results, probabilistic floorplan-based PDR is powerful. When the pedestrian's origin and initial heading are known, assumed step lengths are correct, and sensors do not suffer from major errors, the result converges, yielding an average accuracy of $\approx 1.5\,\mathrm{m}$ for most parts of a walk. However, in case of sensor errors or invalid assumptions, the density easily gets stuck, unable to recover. Furthermore, origin and initial heading are unknown for most use cases, and, if entered manually by the pedestrian, too inaccurate to converge.

Both can be addressed by additionally including sensors that provide absolute location estimations, like Wi-Fi or Bluetooth beacons, examined within the following, final experiments.

6.4 Evaluation of the Overall System

As indicated by previous experiments, Wi-Fi can provide a coarse absolute location estimation indoors, with the quality dependent on the available infrastructure and the chosen signal strength prediction model. When initial heading and whereabouts are known, and a spatial floorplan model is available, PDR can provide accurate results, at the risk of getting stuck, increasing with time, and within large open spaces. When both components are combined, knowledge on initial whereabouts is not required, as they are provided by Wi-Fi, whereas accurate probabilistic movement predictions are included by the floorplan-based PDR, yielding the final indoor localization and navigation system.

The performance of this final system is analyzed using the walks from section 6.1. Each transition is performed using the probabilistic PDR based on the navigation mesh, as it is more suited for embedded use, requiring less memory than the navigation grid. Potential movements are predicted whenever a pedestrian step is detected, moving into the direction given by each particle's heading, adjusted by the cumulated turn rate and some uncertainty (see (3.35) and (3.37)). If a destination is not reachable, the particle remains in place, and is assigned a weight of zero. As some of the walks also include *standing and turning in front of exhibits*, periodic updates are forced every $1000\,\text{ms}$, when no steps were detected.

For the final system, all individual uncertainties are slightly increased, to prevent either component from being too strict, reducing the risk of getting stuck. The chosen walking distances were $\mu_{\text{step}} = 70\,\text{cm}$ with $\sigma_{\text{step}} = 15\,\text{cm}$ when a step was detected, and $\mu_{\text{step}} = 0\,\text{cm}$ with $\sigma_{\text{step}} = 5\,\text{cm}$ for forced periodic updates every $1000\,\text{ms}$, when no step occurred. Along stairs, their known tread-size is used with $\sigma_{\text{step}} = 10\,\text{cm}$. The heading is updated with an uncertainty of $\sigma_{\text{turn}} = 3°$, applied when a step is detected, or a forced update is issued, thus equal to either $3°/\text{s}$ or $6°/\text{s}$. The Wi-Fi component uses `oLDCPFi` as optimization strategy and signal strength prediction model (see section 6.2.7), which is evaluated against the smartphone's RSSI readings using (2.111) and $\sigma_{\text{rssi}} = 10\,\text{dB}$ as uncertainty for every single transmitter. As earlier, the activity-detection (2.80) is included using $\kappa_{\text{match}} = 0.6$, slightly favoring movements matching the detected activity. That is, the barometer is not included individually, but only as part of the activity-detection, serving the same purpose, with a reduced risk for temporal environmental influences. Based on previous findings, the eCompass is included using (2.21) with a large width of $60°$, but only if the last $3\,\text{s}$ provided an almost stable reading, hopefully catching outliers heuristically. The only walk covering an outdoor area, where GPS could be used, is walk A2. However, the actual time outdoors was too short for most smartphone GPS sensors to reach a stable fix, thus not providing a benefit. Including navigational knowledge is only possible for

(a)   (b)
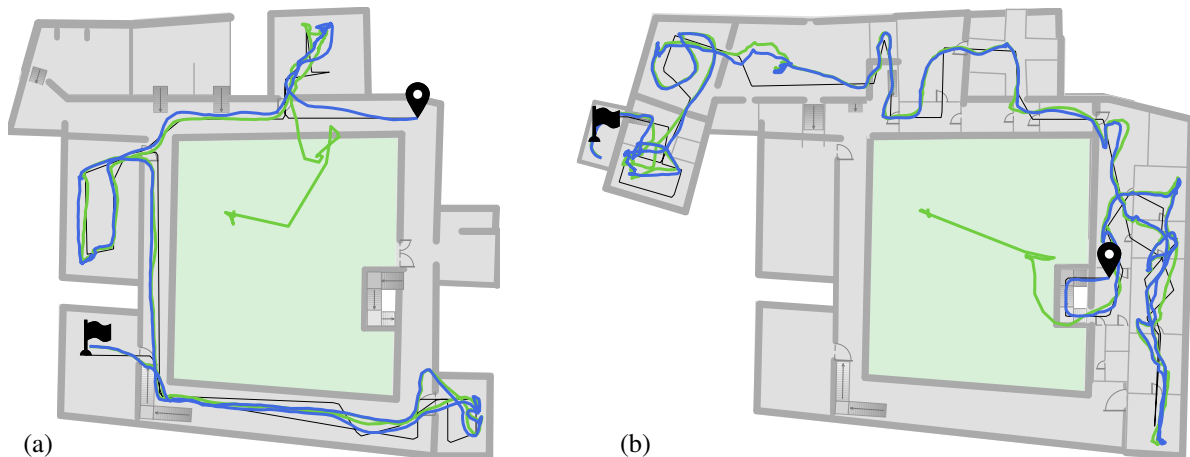
Figure 6.46: Results for walk C2 (a) and walk C3 (b), estimated once using only PDR from a known origin (blue), and once with an unknown origin, using the final system (green). Due to the weighted average estimation, the latter starts in the building's center of mass, taking some seconds to determine likely whereabouts based on nearby access points. Hereafter, both variants yield similar results. Only in some regions Wi-Fi affects the estimation, slightly dragging the density.

walk A3 and walk C1, as all others do not follow a shortest path. While walk A1 and walk A2 are close to using the shortest path, the remaining ones contain loops, and are thus unsuited.

If not mentioned otherwise, experiments were based on the particle filter (see section 4.5) using 5000 particles. This value was chosen empirically, and is suited for most smartphones that are currently in use. Corresponding location estimations are determined by the weighted average of all particles (cf. section 4.5.3).

Figure 6.46 shows the results for two walks, comparing a known-origin PDR, against the final system with an unknown origin. The latter's initial uniform distribution yields an estimation starting in the building's center of mass. After a few seconds, and some RSSI readings from nearby access points, the distribution starts to focus. Hereafter, both variants produce very similar results, as the main focus is on the PDR, and the constraints imposed by the floorplan.

However, in some regions and cases, the Wi-Fi estimation can cause dragging, forcing the density onto a location more likely in terms of Wi-Fi signals. This is similar to earlier findings for the eCompass, when evaluated based on a normal distribution (cf. figure 6.43). This was addressed by using the distribution from (2.21) instead, canceling the observation's mean. While (2.21) can be applied to Wi-Fi as well, it only affects the probability for single transmitters. Similar to the law of total probability, the product of several such distributions – required for the combined Wi-Fi likelihood (2.111) – will contain peaks, again causing dragging towards some location. The impact can be seen when comparing the blue path of figure 6.47a – where Wi-Fi is enabled – against figure 6.45b, displaying the same walk, but using only PDR. The poor radio coverage of the outdoor area causes the Wi-Fi component to provide invalid estima-
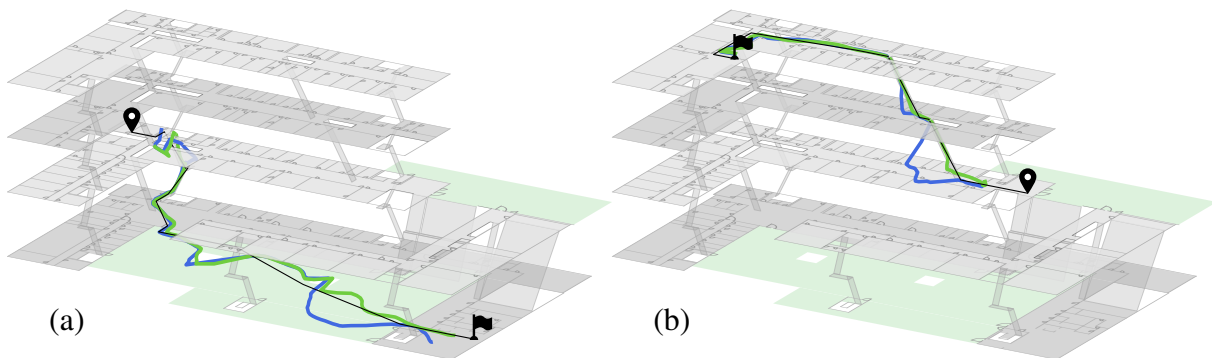
Figure 6.47: Results for walk A2 and walk A3, estimated by the final system. In contrast to the blue result, green also included navigational knowledge in the evaluation, based on $\kappa_{\text{dest}} = 0.7$. For (a), this enhances the last segment of the walk, straightening the outdoor area. Within (b), the initial uncertainty of using the stair is addressed, yielding a faster convergence.

tions, dragging the density. When additionally including navigational knowledge for this walk (green path of figure 6.47a), the effect is mitigated, as walking straight towards the destination in the lower right is more likely. For figure 6.47b the effect of including navigational knowledge (green path) is similar, correctly using stairs instead of staying on the starting floor (blue).

The impact of using other Wi-Fi optimization strategies is shown in figure 6.48. oELD was not further examined, as it requires the locations of the transmitters to be known and added manually to the floorplan. This information often is unavailable, and conducting a few reference measurements takes similar amounts of time, but offers improved accuracy (cf. section 6.2.7). Likewise, oLDCW is omitted, as it provided only minor improvements compared to oLDC, but comes with a computational overhead, exceeding the capabilities of commodity smartphones. While the results for oLDCPFi, oLDCPF and oLDC were very similar, oLD suffers from previously indicated z-estimation problems: By not considering ceilings, adjacent floors receive a similar likelihood from the Wi-Fi component. With the initial density thus splitting onto multiple floors, a decent number of particles is required to converge and prevent the estimation from settling on the wrong floor. This is visualized in figure 6.48b and 6.48c, where the latter used only 1000 particles, causing the recursive density estimation to settle for the wrong floor (yellow particles), unable to converge, eventually getting stuck. When using oLDCPFi, oLDCPF or oLDC, Wi-Fi provides an accurate floor-preference, shown in figure 6.48a. While the estimations in x and y are similar between the depicted models, the error in z is significantly different.

The stability of the final system is also analyzed based on the pedestrian walks from section 6.1. Each of the 12 paths was conducted by various pedestrians and smartphones. All resulting recordings are simulated by the final system 50 times, each of which using a different *seed*, to adjust underlying pseudo random processes. For all results it is determined, whether the estimation converged, and reached the path's destination. The resulting chances for not
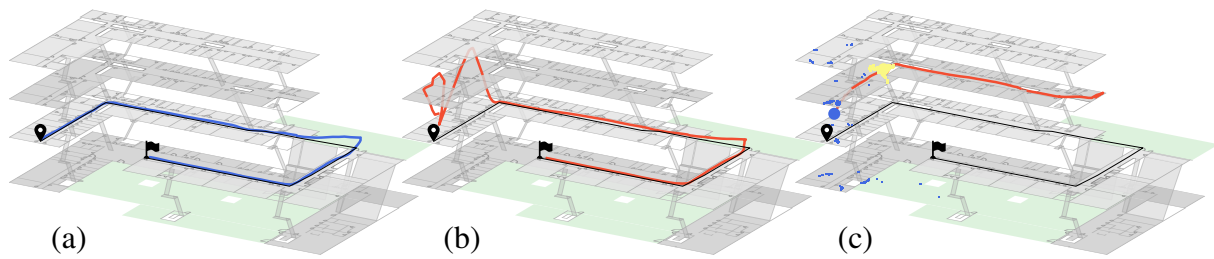
Figure 6.48: Results for walk A1, estimated by the final system. When using oLDCPFi, oLDCPF or oLDC as Wi-Fi model (a), results are almost optimal. For oLD, however, the z-estimation is not ideal: For 5000 particles (b), the estimation eventually converges. For 1000 particles (c), it also starts on all four floors (blue), but settles on the wrong one after a few seconds (yellow), finally getting stuck.

reaching the destination are listed in table 6.19. For most walks, this risk ranged well below 10%. Only walk A2 suffered from a much higher chance, stemming from the walk itself: It starts within a region with viable Wi-Fi location estimation, but after a few steps the pedestrian enters an adjacent stairwell, with poor Wi-Fi-coverage, due to massive concrete walls. Even though a few particles follow into the stairwell, take the stair, matching with the detected activity, the majority remains outside, within areas much more likely in terms of Wi-Fi. That is, the initial uniform distribution did not have enough time to settle on a stable mode, thus failing to converge after a few seconds. This can be mitigated by disabling the Wi-Fi sensor as soon as the stairwell is entered, e.g. by using the Wi-Fi quality metric (5.2). As all measurable RSSIs are rather low throughout the stairwell, the quality metric approaches zero, indicating that the observations are unlikely to be valid. When including this technique within the final system, the number of failed results was reduced from 25% to 15%. Still being higher than for the other examined walks, this is due to the outdoor area, directly before this walk's destination. Previous experiments have shown that the Wi-Fi-coverage outdoors is poor for this building, and signal strength predictions are rather erroneous in this area. Thus, many of the seeded simulations entered through a wrong door, or got stuck shortly before the destination. This could not be mitigated by the GPS either, as the time outdoors was too short for the GPS to estimate a viable fix on most smartphones. Both issues could be addressed by installing additional transmitters throughout such regions. Especially within stairwells, one or two Bluetooth beacons are sufficient to significantly increase localization accuracy, and prevent potential issues.

The localization accuracy of the final system is calculated based on all simulations that converged, reaching the correct destination. For error visualization, the 3D Euclidean distance between estimation and ground truth was determined every $500\,\mathrm{ms}$. Individual results for every walk are shown in figure A.7 to A.15. The corresponding overall results are provided as cumulative distribution functions, grouped by building, shown in figure 6.49. When compared against the Wi-Fi-only location estimation from figure 6.31, the benefit of including additional

	SHL				Museum 1			Museum 2			
A1	A2	A3	A4	A5	B1	B2	B3	C1	C2	C3	C4
4 %	25 %	0 %	8 %	9 %	8 %	9 %	6 %	6 %	6 %	8 %	7 %

Table 6.19: Number of results that got stuck or did not reach the correct destination, determined by applying the final system – 5000 particles, activity-detection, Wi-Fi – 50 times with 50 different seeds, to the conducted pedestrian walks from section 6.1.
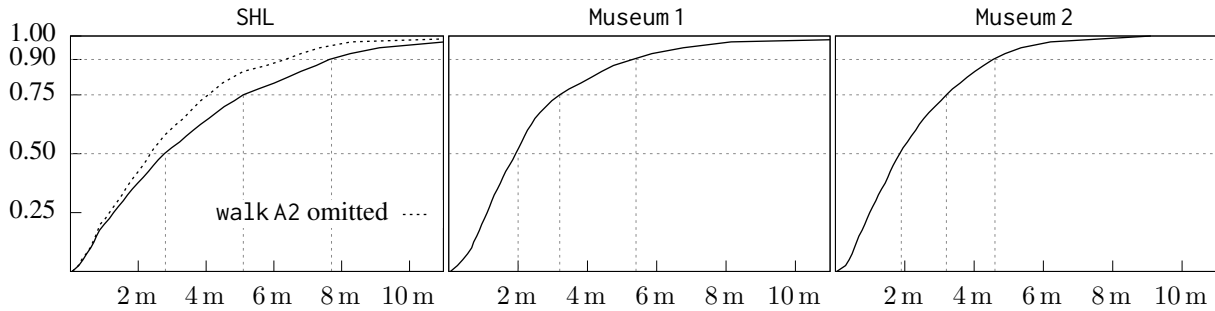
Figure 6.49: Cumulative distribution function after simulating all walks from section 6.1 using the final system, showing the 3D distance error between estimations and ground truth, grouped by building.

sensors and the building's floorplan becomes evident. As earlier, the two museums yield more accurate results, which is mainly due to an increased number of available Wi-Fi transmitters, with respect to each building's size. However, when omitting the results for walk A2 (dashed line), suffering from the mentioned outdoor errors, the results for SHL become similar.

As can be seen from the final experiments, for most cases, the presented final system provides viable results with a median error somewhere around $2\,\mathrm{m}$ to $3\,\mathrm{m}$. While the floorplan allows for 3D location estimation and navigation, preventing impossible movements is not always a benefit. Based on the values listed in table 6.19, there still is a notable risk of the recursive density estimation getting stuck, like for PDR-only approaches. While Wi-Fi was intended to address this risk, it can also be its root cause, when predicted signal strengths do not match with real-world behavior, dragging the density to some location. The presented quality metric (5.2) can mitigate the problem, but does not provide a general solution, as it estimates potential issues solely on low signal strengths. To address these remaining cases, more complex prevention and reset strategies will be required (cf. section 5.2.4), to further enhance the stability in the future.

6.5   Summary

This chapter provided experimental results for all previously introduced techniques. The testbed was based on synthetic tests using a turntable and confined scenarios, as well as real-world walks of several pedestrians within multiple buildings. First, the probabilistic sensor models from chapter 2 were examined, using both, synthetic and real-world setups. Second, the probabilistic movement models from chapter 3 were tested, using synthetic and real-world floorplans, with and without recursive density estimation. Finally, the overall indoor localization and navigation system was evaluated, applying it to multiple walks within actual buildings.

Section 6.1 briefly described the testbed used for all experiments. To allow focused examinations of sensors and effects, synthetic tests were included alongside real-world scenarios. The synthetic variants were based on smartphones placed on a turntable, as well as walks conducted under constrained conditions. Additionally, three different buildings were examined, varying in architecture, size and age. All tests were conducted using Android smartphones of different price classes and ages, carried by several pedestrians in case of real-world walks.

Section 6.2 gave a brief overview on sensors available within various smartphone models. Most of today's devices contain an IMU with accelerometer, gyroscope and magnetometer. While Wi-Fi was available within all examined phones, the barometer is rather rare.

Then, section 6.2.2 examined the accelerometer-based step-detection from section 2.4.1. All conducted tests indicated the benefits of filtering, and the differences between the two discussed implementations, FIR and IIR. The latter is computationally efficient, and, when chained, sufficient for filtering noise, without introducing large delays. A simple peak detection within the filtered accelerometer magnitude was sufficient to provide a robust step-detection. Solely for fast paced walks, the detection rate was reduced, which is due to increasing amounts of noise.

In section 6.2.3, relative turn-detection and absolute heading (section 2.4.2 and 2.4.3) estimation were examined. First, each smartphone was mounted onto a turntable, providing the ground truth for angular changes. This allowed determining the accuracy of the gyroscope, the magnetometer, and the required tilt compensation (see section 2.4.2). For all examined cases and devices, results were promising. Both, turn-detection and eCompass, were off by only a few degrees, indicating that both, sensors and algorithms, are viable. However, real-world experiments conducted in section 6.2.4 denoted several issues. The gyroscope's drift cumulates significantly over time, and the magnetometer is often notably influenced by the buildings architecture.

Shown in section 6.2.5, readings from the barometer (see section 2.5) are also strongly dependent on the current ambient conditions. Absolute altitude estimations thus require some sort of reference. Using changes in altitude since starting to walk thus significantly improved

the probabilistic evaluations. Yet, after walking several minutes, the initial reference is expected to be invalid, due to constantly changing ambient conditions.

An alternative was examined in section 6.2.6. The barometer's change in pressure and the accelerometer's standard deviation within a short timeframe are used for pedestrian activity-detection (cf. section 2.6). While not providing actual altitude estimations, this allows distinguishing between standing, walking and taking stairs, and remains stable even for longer walks. Yet, the detection suffered from notable delays, mainly caused by the barometer sensor itself.

In section 6.2.7, absolute location estimation was examined, provided by the smartphone's signal strength indications for nearby access points (section 2.7). First, signal strength prediction models were trained, based on a few reference measurements conducted within a building. As shown, the quality of the resulting predictions is strongly dependent on the trained model, and the building's architecture. While simple prediction models can be sufficient for some buildings, others require more complex variants, also considering architecture details. Experiments indicated that at least floors and ceilings should be considered to provide viable predictions. Then, based on the trained models, location estimations were performed. While more accurate prediction models tendentially provided better location estimations, the difference between all examined models was less significant. Yet again, the model should at least consider floors and ceilings, as this notably improved floor-level estimations. Independent of the chosen model, often more than one location is likely based on the current readings from the smartphone. Furthermore, Wi-Fi location estimations are noisy, often encountering large jumps. Due to both, filtering is strongly recommended. Results when applying a simple particle filter with an unconstrained prediction yielded notably improved results, by suppressing major outliers. However, even with filtering Wi-Fi on its own is insufficient to solve the problem of indoor localization.

Section 6.3 examined the benefits of including the building's floorplan by using the two spatial models introduced in chapter 3, with and without additional recursive density estimation and sensor fusion. First, the navigation grid and the navigation mesh were utilized to model various real-world buildings, which pointed out major differences. The navigation grid is rather discrete, more suited for axis-aligned architecture, and can require extensive amounts of memory. The navigation mesh adapts to the local architecture, and requires significantly less memory, due to the irregular triangle (primitive) placement. Concerning movement prediction, random walks (see section 3.5.2) are more versatile than the approaches supported by the navigation mesh, as they allow for efficient random sampling based on arbitrary metrics. Yet, for real scenarios and short simulation timeframes, this aspect was less critical than its limitations, producing rather discrete results, and requiring large amounts of memory.

Nevertheless, both spatial models allowed for realistic 3D movement predictions, examined by a pedestrian dead reckoning setup. Here, step-detection and turn-detection were combined

with a known starting position, which was hereafter adjusted based on sensor observations and the floorplan. As shown, this enabled 3D location estimations, even though sensors provide only 2D data. Yet, the impact of cumulating drifts and sensor errors became clear. Probabilistic setups based on the particle filter, including additional knowledge, are thus advisable.

Section 6.3.2 examined the impact of including navigational knowledge, supported by both the navigation grid and navigation mesh. First, both variants required adjustments to prevent the estimated shortest paths from adhering unnaturally close to obstacles. Hereafter, both spatial models provided routes that match with typical pedestrian walking behavior. When included within the probabilistic process, the desired destination can be used to rule out unlikely movements. This often helped the density to converge, especially when starting from a uniform prior.

Likewise, when currently taking stairs, activity-detection reduces the likely whereabouts to a few regions. Shown in section 6.3.3, this often presented major improvements concerning convergence. While the magnetometer's compass can be included in a similar way, to limit potential headings, results strongly depend on ambient conditions. Nearby metal objects often affected observations negatively. Therefore, uncertainty ranges and influences were examined as well, e.g. comparing a normal distribution against the modified version from section 2.3. By not using a clear mean value, the adjusted distribution can prevent negative impacts from biased sensors, such as the magnetometer. The section concluded by examining the contribution of individual components, by calculating the amount of random simulations that got stuck or did not fully converge towards the destination. When the origin is well-known, floorplan-based probabilistic PDR without Wi-Fi provides viable results, as long as no major sensor faults occur. When including Wi-Fi, initial whereabouts need not be known, and faults from other sensors can be compensated. However, poor signal strength readings can also cause the density estimation process to get stuck, sometimes unable to recover.

Finally, section 6.4 examined the overall system behavior for multiple long real-world walks within three buildings, conducted by several pedestrians. The results notably varied between the three buildings, but also among walks within the same building. This is mainly due to certain regions, where Wi-Fi reception is poor, causing significant temporal errors. These errors can also cause the estimation to get stuck, unable to recover. Except for one walk, this risk was similar between all examined walks and buildings. The building with the poorest results among the three was equipped with significantly less wireless transmitters than the other two. While Wi-Fi can compensate gyroscope drift, and drops the requirement for knowing the initial origin, its uncertainty is significant, and often reduces the overall localization result, which could be better when referring to the remaining sensors. Thus indicating the necessity for further improvements.

Chapter 7

Summary

In contrast to navigation outdoors, which has been well established during the last two decades, indoor localization and navigation still remains a niche existence. However, due to the ever expanding need and wish to travel, it becomes of increasing importance. Locating the terminal at an unknown airport, finding the correct ward within a large hospital, or receiving additional information on nearby exhibits within a museum, denote just a few of the numerous use cases. For this new technique to be successful, it should be cheap, easy to set up, and be based on readily available components. These requirements are not limited solely to the system's user, but also apply to its vendors, and the owners of each specific building where the system is installed. Leaving industrial use cases and special hardware components aside, this work focused on pedestrian indoor localization and navigation only. With smartphones being almost always at hand, and containing a variety of sensors, these devices represent a desirable target platform.

The first chapter thus presented various smartphone sensors as well as their contribution towards pedestrian indoor localization and navigation. As the observations from every sensor are noisy or in some way uncertain, they are examined on a probabilistic basis, where the expected amount of noise is included as uncertainty. Throughout the chapter, a relation between sensor readings and potential pedestrian whereabouts or movements within the building was established. After introducing required preprocessing steps, probabilistic evaluations were developed, assigning a quantified likelihood to all potential whereabouts and movements, dependent on expected sensor errors. Based on the well-known global positioning system, a brief example was provided. However, with this localization component unavailable indoors, other smartphone sensors are required. An alternative to the GPS that can be used indoors, and is already available within many public buildings, is Wi-Fi. The smartphone's signal strength readings from nearby transmitters can be used to infer the distance towards them, e.g. allowing multilateration. However, as radio signals are notably affected by walls and other obstacles,

multilateration is too coarse within most buildings. Thus, Wi-Fi is often combined with fingerprinting, performing hundreds of signal strength measurements throughout the building to determine the behavior of the radio signals. Each smartphone observation for nearby transmitters can then be compared with the recorded fingerprints, determining the one that matches best. While offering the best accuracy, the setup process is time consuming. This work thus presented a compromise based on signal strength prediction models that are able to include obstacles within their calculations. These models were trained based on a few reference measurements, hereafter providing signal strength estimations for every transmitter, throughout the whole building. The trained models enable probabilistic location estimations, determining the likelihood for certain whereabouts by comparing their corresponding signal strength predictions with the smartphone's observations. Yet, sometimes being inaccurate, this component is unable to solve the localization problem on its own. If the smartphone contains a barometer sensor, it can be used in a similar way, determining the most likely floor, based on the current ambient pressure readings. Though, with these observations dependent on the weather and changing over time, they require some sort of calibrated reference. Thus, the concept of relative pressure readings was introduced. However, dependent on the duration of the walk and the age of the reference, it eventually becomes invalid, requiring for another approach. This was given by activity-detection, combining the accelerometer and barometer readings within a short timeframe to determine the likelihood of the pedestrian to be currently resting, walking or taking stairs. While it does not provide absolute floor estimations, this approach is unaffected by slow changes in ambient pressure and limits potential whereabouts when taking stairs is detected. Also based on the accelerometer, step-detection was used to infer individual steps made by the pedestrian. Combined with an assumption on the average step size, and including an uncertainty, this provides information on potential movements within a certain timeframe. Besides this velocity, the current walking direction is of similar importance. The magnetometer installed within smartphones was therefore used as a compass, to estimate the heading with respect to geomagnetic north. Yet, indoor environments significantly affect its readings, causing the sensor to often be unstable. More stable and unaffected by surroundings is the gyroscope, which was used to infer changes in heading, referred to as turn-detection. However, as this sensor requires mathematical integration, it suffers from cumulating drifts. The uncertainty thus cumulates over time, and eventually reaches a level where the sensor can not provide a contribution.

Similar to navigation outdoors, mapping information can not only be used to estimate the shortest path to a desired destination. It can also be considered to limit impossible movements, and thus to reduce cumulating uncertainties. For this, two spatial models were introduced, modeling the walkable surface: A graph with vertices and edges, referred to as navigation grid, and the navigation mesh, using adjacent and irregularly sized triangles. Both spatial models allow

for movement predictions based on the building's floorplan. The presented random walks along the graph, for example, estimate all pedestrian movements that are possible within a certain timeframe. The intention is similar to the approaches used within car navigation systems. A car's last known velocity, heading, and road can be used to provide predictions, even when the GPS is temporarily unavailable, like within a tunnel. Furthermore, this approach can also be used to suppress outliers, when the GPS indicates a faulty reading that does not conform with the last known heading and velocity. Yet, in contrast to car navigation, pedestrian movement behavior is significantly less restricted, and usually not limited to certain paths or directions. Unlike cars, a pedestrian can stop, accelerate, or turn at any given instant. This aspect was examined based on movement prediction models with and without knowledge of the building's floorplan. As pointed out, the quality of movement predictions depends on the available prior knowledge. When assumptions on velocity, heading, and surrounding obstacles can be made, results are significantly more realistic than without. Including the information from step-detection, turn-detection is thus strongly advisable, notably limiting potential movements. This also concerns navigational knowledge. When the pedestrian's desired destination is known, the shortest route for reaching it can be estimated. For this, common routing algorithms were used, with a modification to create realistic walking paths through the building, while avoiding nearby obstacles. The system then assumes that the user follows the presented routing. While deviating from it is still possible, it is typically less likely.

The combination of both aspects, sensor observations and floorplan-based movement prediction, was provided by recursive density estimation, using the Bayes filter. It enables a probabilistic fusion of multiple individual aspects, including their expected uncertainties, hereafter deriving the most likely result. Applied to localization indoors, the Bayes filter estimates the pedestrian's most likely whereabouts based on the history of all sensor observations, and constraints given by the floorplan. This also increases the usefulness of relative sensors, like step-detection and turn-detection, as their whole series of observations is considered, and matched against the floorplan. While the Bayes filter can be implemented analytically using the (extended) Kalman filter, this variant is too restricted for indoor localization, as it does not support the discontinuous constraints from the floorplan. This was addressed by using the particle filter instead, which approximates all required densities based on several discrete samples, referred to as particles. Each of the particles represents one potential location the pedestrian might currently reside at, including an estimated walking direction. The particles can directly be used to simulate potential movements, e.g. by relocation based on an assumed walking speed, hereafter determining whether the resulting movement is possible by the floorplan.

For an efficient, yet flexible, implementation of the overall system, floorplan design choices were briefly discussed. A manually modeled floorplan is hereafter used to automatically derive

a navigation grid or navigation mesh for movement predictions. When using the particle filter as implementation, particles can denote impossible movements, if they collide with a wall or another obstacle. These impossible movements must be addressed, to prevent negative influences, and to ensure that all particles remain likely. For this, several strategies were discussed, such as assigning corresponding particles an infinitesimally small weight, which causes them to be replaced during the particle filter's resampling step.

Shown by experiments, all examined smartphone sensors provide a viable contribution towards indoor localization. Their general suitability was examined by synthetic tests, suppressing unwanted influences, and focusing on individual sensors only. A turntable provided an accurate ground truth to inspect turn-detection and absolute heading estimations, indicating viable results among all examined smartphones. Likewise, step-detection applied to constrained walks was accurate, as long as the pedestrians used typical pace. While both sensors combined already enable indoor localization via pedestrian dead reckoning, this setup requires initial whereabouts to be well known, and is likely to fail eventually, due to cumulating sensor errors. While Wi-Fi provides absolute location estimations, its estimated whereabouts can notably deviate from the ground truth. Corresponding estimations were performed within three buildings, based on signal strength prediction models, trained by a few reference measurements. Comparing the results, the limitations of the examined prediction models became clear. Simple line of sight models are too inaccurate to provide viable estimations, and at least floors and ceilings should be considered. Using multiple models, each focusing on a fraction of the building, further increased the prediction quality. Matching with these findings, the probabilistic setup, based on the particle filter, has proven to be notably more robust. Triggering updates whenever a step is detected, each particle's heading estimation is updated based on turn-detection and the associated uncertainty. This relocates all particles to new whereabouts, if the floorplan conforms with a potential movement. If not, the movement is unlikely and thus suppressed. Similarly, movements not matching with the currently detected activity are suppressed as well. Additionally, Wi-Fi weights all particles based on whether the signal strength prediction for their location matches with the smartphone's current observations. Over time, this combination of multiple sensors, movement prediction, and the building floorplan yields a robust probabilistic location estimation. When assumptions on sensor noise and errors are correct, every additional component increases the quality of the particle filter's estimation. Yet, sensor faults, like poor Wi-Fi observations, or magnetometer readings affected by nearby metal objects, can cause the density approximation to destabilize. When encountered for longer timeframes, the particles can e.g. get stuck within a room, unable to leave. While several techniques for mitigation exist, which strategy and combination serves best, is one of several topics to be examined in the future.

Chapter 8

Future Work

Discussed within previous chapters and confirmed by experiments, the presented indoor localization and navigation system provided an accuracy that is sufficient for localization and routing within most buildings. However, indicated by the examined real-world walks, some caveats remain and should be addressed in the future to further improve accuracy and stability.

The used step-detection e.g. relies on an empiric choice for the pedestrian's step length and its uncertainty. While this was sufficient for most of the examined cases, it is not a generic solution. To improve this part of the system, a dynamic step length estimation should be considered, such as the ones provided by [Goy+11] or [Yu+19]. In doing so, varying walking patterns, like hectically pacing towards a gate in an airport, can be supported as well. While the presented turn-detection itself does not rely on such constants, its uncertainty assumed within evaluations also is a rather empiric heuristic. Here, additional research should examine methods to determine the sensor's current uncertainty on a profound basis [Cla+17]. This allows detecting sensor errors, and situations where the turn-detection is unstable. As shown within experiments, while the magnetometer can generally provide a stable heading, the influences within indoor environments significantly limit its applicability. Here, further research should be conducted to determine whether environmental influences can be detected from the observations themselves. If so, the magnetometer should only be considered when it is expected stable. Besides, due to these influences being region-dependent, this opens room for absolute location estimations based on magnetic fingerprinting [Shu+15; KS18]. Just like Wi-Fi fingerprinting, this compares the current sensor observations against a database of several localized measurements. Somewhat similar, critical areas of a building could also be equipped with devices that transmit unique magnetic patterns, to provide additional location estimations by detecting these patterns within observations [EBS16]. To further improve accuracy, these techniques should be examined and integrated if they match with the requirement of being cheap and easy to set

up. While the barometer also indicated room for further improvements, its limited availability within recent smartphones does not suggest that further research is advisable. To the contrary, this indicates the need for modifying the activity-detection to work without this sensor. More complex estimators, additionally including the gyroscope and magnetometer, could be used to distinguish between the presented activities, without requiring a barometer [Cil+14; Li+18]. However, indicated by sensor data gathered during experiments, when the smartphone is held upfront, the difference between walking and taking stairs is minimal, and thus potentially hard to classify. This indicates the need for additional research concerning smartphone-only pedestrian activity-detection. Wi-Fi signal strength prediction and location estimation also yield room for further improvements. The used reference measurements already provided setup times that are notably faster than the ones required for fingerprinting. Yet, these times could further be reduced by modifying their acquisition. Instead of resting and measuring at predefined locations, the pedestrian could walk consistently through the building, following a defined path. Based on the recorded data, the exact location belonging to each received measurement can be interpolated along the path, similar to the ground truth estimations discussed in section 6.1. Just like step-detection and turn-detection, the uncertainty assumed for measured Wi-Fi signal strengths also relies on an empiric choice, and research on potential dynamic estimations for this value should be conducted. Furthermore, shown within experiments, sometimes Wi-Fi estimations are completely invalid, which indicates a non zero mean error. To prevent corresponding observations from causing localization errors, circumstances should be further examined, to derive a more robust metric for sensor fault detection.

Like the sensors, indoor maps and corresponding pedestrian movement predictions also hold room for further improvements. Discussed only briefly, escalators and elevators should be modeled, and considered within predictions. However, while supported by the navigation grid, elevators represent a problem for the navigation mesh. The mesh describes the walkable *surface*, but vertical surfaces are neither supported by the mesh, nor by the movement prediction algorithms presented for it, thus requiring future work on potential solutions. Besides, both presented spatial models, navigation grid and navigation mesh, are derived from a manually created indoor map. Depending on the size of the building, the time needed for creating this floorplan can be significant. Therefore, further research should be conducted, examining the suitability of other means for creating them, such as using scanned blueprints, cameras, laser-scanners, SLAM, robots, or similar [Çel+09; Hes+16; SCI13; Zha+15; Liu+17; CF14].

When examining the results from the overall system, a few caveats can be detected as well. Even though the floorplan prevents walking through walls, many of the displayed paths, estimated from the particle filter, did directly cross obstacles. While this is not an issue in general, as the pedestrian is only interested in the current whereabouts and not their history, it leads to

related future work. The reason for the issue lies, among others, within the used weighted average estimation, which, for multimodal densities, often resides in between all modes, and thus anywhere on the map, independent of obstacles. This can e.g. be addressed by using the maximum of a KDE instead, at the cost of computational complexity [Bul+18]. On the other hand, the discussed behavior also results from the path being constructed by simply connecting consecutive estimations, not considering obstacles in between. Briefly mentioned earlier, this can be improved by introducing a slight delay on the results presented to the user, thereby making future observations available to the estimation process. Doing so allows for retrospective result smoothing, which yields better paths, but, more importantly, also stabilizes the estimations presented to the pedestrian, at the cost of a slight delay [Fet+16]. Another floorplan-related issue is the risk of the recursive density estimation process to get stuck. Poor Wi-Fi estimations, or other sensor faults, can cause the density to get dragged into some room, unable to leave. Described only briefly, these situations should be detected and handled, e.g. by running multiple particle filters (IMMPF), one with floorplan and one without, mixing both based on error estimations [Fet+17]. Besides causing the density to get stuck, Wi-Fi can also yield more subtle issues. Shown in experiments, when the pedestrian's origin is known, particle filter-based PDR provides viable results. When including Wi-Fi, a known origin is not required, and its absolute indications can correct gyroscope-drift. However, Wi-Fi often showed a negative effect, increasing the localization error unnecessarily. Here, further research should be conducted, e.g. using the IMMPF to mix between PDR and Wi-Fi only when necessary, using the best of both estimation results.

Concerning an indoor navigation product, additional aspects not discussed within this work should be considered. One of which are viable visualization strategies. Potential options are to present the pedestrian a 2D top view of the current floor, a 3D view in first person perspective, or an augmented reality, where routing information is displayed on top of the smartphone's camera image [HB08]. Here, further research should be conducted, examining which variant provides the best usability. Also, due to the large variety of different smartphones and corresponding sensors, localization accuracy will vary between different brands and models. Especially when a sensor component is unavailable, or provides erroneous observations. These and similar aspects should be taken into account as well, examining the impact on an even broader range of devices.

List of Figures

List of Tables

List of Symbols

Sign	Description
α, β	general purpose angles
d	general purpose scalar distance value
κ	general purpose scalar value used for mixing κ and $(1 - \kappa)$
η	general prupose normalization constant
τ	general purpose threshold value
$\boldsymbol{u}, \boldsymbol{v}$	general purpose vectors
v	general purpose velocity value
ε	general purpose error value
E	general purpose cumulative error value
$\boldsymbol{\rho}$	general purpose 3D point
$\rho^{(x)}, \rho^{(y)}, \rho^{(z)}$	components of a 3D point
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	general purpose random variables
$\boldsymbol{\rho}_{\text{dest}}$	the pedestrian's desired destination
Q	quaternion
bw	KDE bandwidth
$\boldsymbol{q}$	unknown state vector
$q_t^{(x)}, q_t^{(y)}, \ldots$	attributes of the unknown state at time t
$q_t^{(\Theta)}$	unknown state heading at time t
$q_{t-1}^{(\varepsilon_{\text{head}})}$	unknown state cumulated heading error at time $t - 1$
$q_{t-1}^{(\varepsilon_{\text{dist}})}$	unknown state distance error at time $t - 1$
$\boldsymbol{o}$	observation vector
$o_t^{(\Theta)}, o_t^{(\boldsymbol{s})}, \ldots$	attributes of the observation at time t
w	probabilistic weight
W	cumulated probabilistic weight
$G = (V, E)$	graph with vertices and edges
v_i	vertex within a graph G
$v_i^{(x)}, v_i^{(y)}, \ldots$	attributes of vertex i

Sign	Description
$e_{i,j}$	edge between vertex v_i and v_j
v_{dest}	vertex for the pedestrian's desired destination
gs	distance between adjacent graph vertices in x and y
gs_z	distance between adjacent graph vertices in z
$p(e_{i,j} \mid \boldsymbol{q}_{t-1})$	probability to walk the given edge
$\boldsymbol{g}$	GPS observation $(\text{lon}, \text{lat}, \text{alt}, \text{err})$
$\boldsymbol{a}$	accelerometer observation (x, y, z)
$\boldsymbol{\omega}'$	gyroscope observation, turn rate (x, y, z)
$\boldsymbol{\theta}'$	cumulative gyroscope turn rates
$Q^{\boldsymbol{\omega}'}, Q^{\boldsymbol{\theta}'}, Q^{\boldsymbol{a}}, \hat{Q}$	quaternions for: gyro turn rates, cumulative gyro turn rates, pose from accelerometer, filtered pose
$\boldsymbol{R}$	smartphone pose rotation matrix
$\boldsymbol{R}^{-1}$	tilt compensation matrix
ω	the pedestrian's turn rate
θ	relative heading indication (cumulated within a timeframe)
Θ	absolute heading indication (cumulated since start)
$\boldsymbol{B}'$	magnetometer observation, magnetic flux density (x, y, z)
$\boldsymbol{B}$	magnetic flux density vector, after tilt compensation
$\Theta_{\text{N}_{\text{mag}}}$	angle towards geomagnetic north
$\Theta_{\text{N}_{\text{geo}}}$	angle towards geographic north
Θ_{dec}	local declination correction
Θ_{bldg}	local declination correction, including building orientation
ϱ	barometer observation, atmospheric pressure
h	altitude
h_{bldg}	building's first floor's altitude, above mean sea level
Ω	activity class
$\boldsymbol{s}$	observed Wi-Fi signal strengths
s_{ap}	observed Wi-Fi signal strength for one transmitter

Sign	Description
ς	offline RSSI database (fingerprints/reference measurements)
$\varsigma_{\text{fp,ap},n}$	n-th measurement for transmitter ap at location fp
λ	wavelength
$\boldsymbol{\rho}$	3D position of a model prediction
$\boldsymbol{\rho}, \boldsymbol{\rho}$	3D position of a transmitter/receiver
P, P	signal power at a transmitter/receiver
G, G	Wi-Fi antenna gain for a transmitter/receiver
γ	Wi-Fi path-loss exponent
ϕ	attenuation factor for a material
$\boldsymbol{\psi}$	signal strength prediction model parameters
$\boldsymbol{\psi}_{\text{ap}}$	signal strength prediction model parameters for one AP
κ_{dest}	empiric value for favoring the shortest path
κ_{act}	empiric value for favoring the same activity
$\kappa_{\text{w/s}}$	empiric value to mix standing and walking

Bibliography

[AP99]   E. Abbott and D. Powell. "Land-Vehicle Navigation Using GPS". In: *Proceedings of the IEEE* 87.1, 1999.

[ARC12]   Imad Afyouni, Cyril Ray, and Christophe Claramunt. "Spatial Models for Indoor Context-Aware Navigation Systems: A Survey". In: *Journal of Spatial Information Science* 4, 2012.

[Aig15]   Martin Aigner. *Graphentheorie : Eine Einführung aus dem 4-Farben Problem.* Wiesbaden: Springer Spektrum, 2015.

[AY12]   Moustafa Alzantot and Moustafa Youssef. "CrowdInside: Automatic Construction of Indoor Floorplans". In: *International Conference on Advances in Geographic Information Systems*. 2012.

[ARY95]   Jørgen Bach Andersen, Theodore S. Rappaport, and Susumu Yoshida. "Propagation Measurements and Models for Wireless Communications Channels". In: *IEEE Communications Magazine*, 1995.

[ABA11]   J. Andreu, R. D. Baruah, and P. Angelov. "Real Time Recognition of Human Activities from Wearable Sensors by Evolving Classifiers". In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2011.

[App]   Apple. *Documentation - Core Motion - Getting Raw Accelerometer Events.* https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events. Accessed: 2020-01-04.

[AAO11]   A. A. B. Ariffin, N. H. A. Aziz, and K. A. Othman. "Implementation of GPS for Location Tacking". In: *IEEE Control and System Graduate Research Colloquium*. 2011.

[Ark87]   Ronald Craig Arkin. "Towards Cosmopolitan Robots : Intelligent Navigation in Extended Man-made Environments". PhD thesis. University of Massachusetts, 1987.

[Aru+01]   Sanjeev Arulampalam et al. "A Tutorial on Particle Filters for On-line Non-linear / Non-Gaussian Bayesian Tracking". In: *IEEE Transactions on Signal Processing* 50, 2001.

[Ary+98]   Sunil Arya et al. "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions". In: *Journal of the ACM (JACM)* 45.6, 1998.

[Aur91]   Franz Aurenhammer. "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure". In: *ACM Computing Surveys (CSUR)* 23.3, 1991.

[AC16]   J. Azevedo and S. Crisóstomo. "Weather Stations-Assisted Barometric Altimeter for Android: Interpolation Techniques for Improved Accuracy". In: *IEEE Sensors Applications Symposium (SAS)*. 2016.

[BP00]   Paramvir Bahl and Venkata N. Padmanabhan. "RADAR: An In-Building RF-based User Location and Tracking System". In: *Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 2. 2000.

[BSA16]   Leor Banin, Uri Schatzberg, and Yuval Amizur. "WiFi FTM and Map Information Fusion for Accurate Positioning". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[BP66]   Leonard E. Baum and Ted Petrie. In: *The Annals of Mathematical Statistics* 37, 1966.

[Baw+15]   B. Bawazeer et al. "Reducing Scanning Delay for WiFi-to-WhiteFi Handovers". In: *Fifth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*. 2015.

[BJK05]   P. Beeson, N. K. Jong, and B. Kuipers. "Towards Autonomous Topological Place Detection Using the Extended Voronoi Graph". In: *IEEE International Conference on Robotics and Automation*. 2005.

[Ben75]   Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Communications ACM* 18.9, 1975.

[Ber17]   Florian Berchtold. *Geometrie : von Euklid bis zur hyperbolischen Geometrie mit Ausblick auf angrenzende Gebiete*. Springer Spektrum, 2017.

[Ber+08]   Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[BD13]   Gérard Blanchet and Bertrand Dupouy. *Computer Architecture*. Iste Wiley, 2013.

[Blo83]   Jacques-François Blondel. *Cours d'architecture*. 1683.

[BDH04]   Miodrag Bolić, Petar M. Djurić, and Sangjin Hong. "Resampling Algorithms for Particle Filters: A Computational Complexity Perspective". In: *Journal on Advances in Signal Processing (EURASIP)* 15, 2004.

[BC12]   William Bolstad and James Curran. *Introduction to Bayesian Statistics*. Wiley, 2012.

[Bon+01]   P. Bonnifait et al. "Data Fusion of Four ABS Sensors and GPS for an Enhanced Localization of Car-like Vehicles". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 2001.

[BN09]   A. G. Bors and N. Nasios. "Kernel Bandwidth Estimation for Nonparametric Modeling". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6, 2009.

[Bos14]   Bosch Sensortec. *BMC156 - 6-axis eCompass*. BST-BMC156-DS000-01. 2014.

[Bos15]   Bosch Sensortec. *BMP180 - Digital pressure sensor*. BST-BMP180-DS000-12. 2015.

[Bos18]   Bosch Sensortec. *BMP280 - Digital pressure sensor*. BST-BMP280-DS001-19. 2018.

[Bou13]   M. Bouhedda. "Neuro-Fuzzy Sensor's Linearization Based FPGA". In: *IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*. Vol. 1. 2013.

[BM58]   George Edward Pelham Box and Mervin Edgar Muller. "A Note on the Generation of Random Normal Deviates". In: *The Annals of Mathematical Statistics* 29, 1958.

[Bra+05]   D. Braganza et al. "Tracking Control for Robot Manipulators with Kinematic and Dynamic Uncertainty". In: *IEEE Conference on Decision and Control*. 2005.

[Bre65]   J. E. Bresenham. "Algorithm for computer control of a digital plotter". In: *IBM Systems Journal* 4.1, 1965.

[Bro+06]   Raymond C. Browning et al. "Effects of obesity and sex on the energetic cost and preferred speed of walking". In: *Journal of Applied Physiology* 100.2, 2006.

[Bul+18]   Markus Bullmann et al. "Fast Kernel Density Estimation using Gaussian Filter Approximation". In: *International Conference on Information Fusion (FUSION)*. 2018.

[BS12]   M. Butta and I. Sasada. "Sources of Noise in a Magnetometer Based on Orthogonal Fluxgate Operated in Fundamental Mode". In: *IEEE Transactions on Magnetics* 48.4, 2012.

[CF14]   R. Cabral and Y. Furukawa. "Piecewise Planar and Compact Floorplan Reconstruction from Images". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[CGM07]   O. Cappe, S. J. Godsill, and E. Moulines. "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo". In: *Proceedings of the IEEE* 95.5, 2007.

[Cap+17]   N. Capurso et al. "An Android-Based Mechanism for Energy Efficient Localization Depending on Indoor/Outdoor Context". In: *IEEE Internet of Things Journal* 4.2, 2017.

[CB07]   Tom Carpenter and Joel Barrett. *CWNA Certified Wireless Network Administrator Official Study Guide (Exam PW0-100)*. 4th ed. McGraw-Hill Osborne Media, 2007.

[Cas+14]   D. Caspari et al. "Smartphone Sensor Based Algorithms for Dead Reckoning Using Magnetic Field Sensor and Accelerometer for Localization Purposes". In: *International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems*. 2014.

[Çel+09]   Koray Çelik et al. "Monocular Vision SLAM for Indoor Aerial Vehicles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2009.

[CHP79]   Y. T. Chan, A. G. C. Hu, and J. B. Plant. "A Kalman Filter Based Tracking Scheme with Input Estimation". In: *IEEE Transactions on Aerospace and Electronic Systems* 2, 1979.

[Cha82]   B. Chazelle. "A Theorem on Polygon Cutting with Applications". In: *Annual Symposium on Foundations of Computer Science (SFCS)*. 1982.

[CYJ15]   Y. Chen, J. Yang, and S. Jiang. "Data Validation and Dynamic Uncertainty Estimation of Self-validating Sensor". In: *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2015.

[Che+05]   Yu-Chung Cheng et al. "Accuracy Characterization for Metropolitan-scale Wi-Fi Localization". In: *International Conference on Mobile Systems, Applications, and Services*. 2005.

[Che+12]   Yan-Ming Cheng et al. "Analysis on Interference Impact of WiFi on DTV". In: *World Automation Congress*. 2012.

[CHP16]   C. Chesneau, M. Hillion, and C. Prieur. "Motion estimation of a Rigid Body with an EKF using Magneto-Inertial Measurements". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[CPP10]   Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. "Indoor Localization Without the Pain". In: *Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2010.

[Cho+12]   I. N. Cholakova et al. "Temperature Influence on Hall Effect Sensors Characteristics". In: *Telecommunications Forum (TELFOR)*. 2012.

[CB95]   H. Choset and J. Burdick. "Sensor Based Planning, Part I. The Generalized Voronoi Graph". In: *IEEE International Conference on Robotics and Automation*. Vol. 2. 1995.

[CZK98]   O. Chutatape, Liu Zheng, and S. M. Krishnan. "Retinal blood vessel detection and tracking by matched Gaussian and Kalman filters". In: *International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 6. 1998.

[Cil+14]   F. De Cillis et al. "Indoor Positioning System using Walking Pattern Classification". In: *Mediterranean Conference on Control and Automation*. 2014.

[Cla83]   K. L. Clarkson. "Fast Algorithms for the All Nearest Neighbors Problem". In: *Annual Symposium on Foundations of Computer Science (SFCS)*. 1983.

[Cla+17]   P. Clausen et al. "An Overview of a New Sensor Calibration Platform". In: *IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2017.

[Cla16]   Christoph Clauser. *Einführung in die Geophysik : globale physikalische Felder und Prozesse in der Erde*. Springer Spektrum, 2016.

[Cor09]   Thomas Cormen. *Introduction to Algorithms*. MIT Press, 2009.

[CK08]   Erhard Cramer and Udo Kamps. *Grundlagen der Wahrscheinlichkeitsrechnung und Statistik*. Springer, 2008.

[CS11]   Xiao Cui and Hao Shi. "A*-based Pathfinding in Modern Computer Games". In: *International Journal of Computer Science and Network Security (IJCSNS)* 11.1, 2011.

[Cul56]   E. G. Cullwick. "Electromagnetic Momentum and Electron Inertia in a Current Circuit". In: *Proceedings of the IEE - Part C: Monographs* 103.3, 1956.

[DH03]   Winnie Daamen and Serge Hoogendoorn. "Experimental Research of Pedestrian Walking Behavior". In: *Transportation Research Record Journal of the Transportation Research Board* 1828, 2003.

[DT05]   N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2005.

[Del98]   Pierre Del Moral. "Measure Valued Processes and Interacting Particle Systems. Application to Non Linear Filtering Problems". In: *The Annals of Applied Probability* 8, 1998.

[Del96]   Pierre Del Moral. "Nonlinear Filtering: Interacting Particle Solution". In: *Markov Processes and Related Fields* 2, 1996.

[Del34]   B. N. Delaunay. "Sur la sphère vide". In: *Bulletin de l'Académie des Sciences de l'URSS* 1934.6, 1934.

[DLK07]   Nikos Deligiannis, Spiros Louvros, and Stavros Kotsopoulos. "Optimizing Location Positioning Using Hybrid TOA-AOA Techniques in Mobile Cellular Networks". In: *International Conference on Mobile Multimedia Communications*. 2007.

[DW11]   Henry Deng and Hadley Wickham. *Density estimation in R*. Tech. rep. 2011.

[Deu10]   Deutsche Akkreditierungsstelle. *DAkkS-DKD-5 - Anleitung zum Erstellen eines Kalibrierscheines*. 1st ed. Version 2. 2010.

[Die06]   James Diebel. "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors". In: *Matrix* 58, 2006.

[Dij59]   E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1, 1959.

[Do+09]   D. V. Do et al. "Impedance Observer for a Li-Ion Battery Using Kalman Filter". In: *IEEE Transactions on Vehicular Technology* 58.8, 2009.

[DH10]   Hans Dodel and Dieter Häupler. *Satellitennavigation*. 2nd ed. Springer, 2010.

[DC05]   Randal Douc and O Cappe. In: 2005.

[DGK01]   A. Doucet, N. J. Gordon, and V. Krishnamurthy. "Particle Filters for State Estimation of Jump Markov Linear Systems". In: *IEEE Transactions on Signal Processing* 49.3, 2001.

[DS84]   Peter Doyle and James Snell. *Random Walks and Electric Networks*. American Mathematical Society, 1984.

[DB05]   H. Driessen and Y. Boers. "Efficient particle filter for jump Markov nonlinear systems". In: *IEE Proceedings - Radar, Sonar and Navigation* 152.5, 2005.

[DMX10]   I. Dryanovski, W. Morris, and J. Xiao. "Multi-Volume Occupancy Grids: An Efficient Probabilistic 3D Mapping Model for Micro Aerial Vehicles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010.

[Dvo+19]   Nir Dvorecki et al. "A Machine Learning Approach for Wi-Fi RTT Ranging". In: *International Technical Meeting of The Institute of Navigation*. 2019.

[DZM07]   Ramsay Dyer, Hao Zhang, and Torsten Möller. "Voronoi-Delaunay Duality and Delaunay Meshes". In: *ACM Symposium on Solid and Physical Modeling (SPM)*. 2007.

[Ebn+15]   Frank Ebner et al. "Multi Sensor 3D Indoor Localisation". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2015.

[Ebn+16]   Frank Ebner et al. "On Prior Navigation Knowledge in Multi Sensor Indoor Localisation". In: *International Conference on Information Fusion (FUSION)*. 2016.

[Ebn+17]   Frank Ebner et al. "On Wi-Fi Model Optimizations for Smartphone-Based Indoor Localization". In: *International Journal of Geo-Information (ISPRS)* 6.8, 2017.

[Ebn+14]   Frank Ebner et al. "Robust Self-Localization using Wi-Fi, Step/Turn-Detection and Recursive Density Estimation". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2014.

[Eck87]   Roger Eckhardt. "Stan Ulam, John Von Neumann, Monte Carlo Method". In: *Los Alamos Science*, 1987.

[EBS16]   C. R. Ehrlich, J. Blankenbach, and A. Sieprath. "Towards a Robust Smartphone-based 2.5D Pedestrian Localization". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[Elf89]   A. Elfes. "Using Occupancy Grids for Mobile Robot Perception and Navigation". In: *Computer* 22.6, 1989.

[Elh+14]   M. Elhoushi et al. "Using Portable Device Sensors to Recognize Height Changing Modes of Motion". In: *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2014.

[ETS12]   ETSI. *Broadband Radio Access Networks (BRAN); 5 GHz high performance RLAN; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive*. 2012.

[Eur04]   Eurocontrol. *Guidance Material for Transition Altitude Change*. 2004.

[EvA16]   EvAAL. *IPIN 2016 Indoor Localization Competition - Results*.
            `http://evaal.aaloa.org/2016/competition-results`. Accessed: 2020-01-04. 2016.

[Fan+11]   J. Fang et al. "A Novel Calibration Method of Magnetic Compass Based on Ellip-
            soid Fitting". In: *IEEE Transactions on Instrumentation and Measurement* 60.6,
            2011.

[Fed16]   Federal Aviation Administration (FAA). *Pilot's Handbook of Aeronautical Knowl-
            edge*. Aviation Supplies and Academics, 2016.

[Fet+16]   Toni Fetzer et al. "On Monte Carlo Smoothing in Multi Sensor Indoor Localisa-
            tion". In: *International Conference on Indoor Positioning and Indoor Navigation
            (IPIN)*. 2016.

[Fet+17]   Toni Fetzer et al. "Recovering from Sample Impoverishment in Context of In-
            door Localisation". In: *International Conference on Indoor Positioning and In-
            door Navigation (IPIN)*. 2017.

[Fet+18]   Toni Fetzer et al. "Smartphone-Based Indoor Localization within a 13th Century
            Historic Building". In: *Sensors* 18, 2018.

[FB81]   Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm
            for Model Fitting with Applications to Image Analysis and Automated Cartogra-
            phy". In: *Communications of the ACM* 24.6, 1981.

[Fit71]   R. Fitzgerald. "Divergence of the Kalman Filter". In: *IEEE Transactions on Au-
            tomatic Control* 16.6, 1971.

[Flu99]   Fluke Corporation. *Dual Display Multimeter - Service Manual*. PN 609203. 1999.

[FG02]   R. J. Fontana and S. J. Gunderson. "Ultra-wideband precision asset location sys-
            tem". In: *IEEE Conference on Ultra Wideband Systems and Technologies*. 2002.

[For+10]   Catherine Forbes et al. *Statistical Distributions*. 4th ed. Wiley, 2010.

[FM63]   S. M. Forman and M. J. Minneman. "A Magnetic Induction Gyroscope". In: *IEEE
            Transactions on Military Electronics* MIL-7.1, 1963.

[Fri46]   Harald T. Friis. "A Note on a Simple Transmission Formula". In: *Proceedings of
            the IRE* 34.5, 1946.

[Fri71]   Harald T. Friis. "Introduction to Radio and Radio Antennas". In: *Spectrum, IEEE*
            8.4, 1971.

[Fro+13]   M. Froehle et al. "Cooperative Multipath-Assisted Indoor Navigation and Track-
            ing (Co-MINT) Using UWB Signals". In: *IEEE International Conference on
            Communications Workshops (ICC)*. 2013.

[FT04]   Taku Fujiyama and Nick Tyler. "An Explicit Study on Walking Speeds of Pedestrians on Stairs". In: *International Conference on Mobility and Transport for Elderly and Disabled People*. 2004.

[Gar+16]   S. García et al. "Indoor SLAM for Micro Aerial Vehicles Control Using Monocular Camera and Sensor Fusion". In: *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2016.

[GCC12]   Shima Gerani, Mark Carman, and Fabio Crestani. "Aggregation Methods for Proximity-Based Opinion Retrieval". In: *ACM Transactions on Information Systems* 30.4, 2012.

[GY15]   M. Ghanbari and M. J. Yazdanpanah. "Delay Compensation of Tilt Sensors Based on MEMS Accelerometer Using Data Fusion Technique". In: *IEEE Sensors Journal* 15.3, 2015.

[GI10]   T. Glasmachers and C. Igel. "Maximum Likelihood Model Selection for 1-Norm Soft Margin SVMs with Multiple Parameters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.8, 2010.

[Gla90]   Andrew Glassner. *Graphics Gems*. Academic Press, 1990.

[GDW01]   S.J. Godsill, Arnaud Doucet, and Mike West. "Maximum a Posteriori Sequence Estimation Using Monte Carlo Particle Filters". In: *Annals of the Institute of Statistical Mathematics* 53, 2001.

[GV13]   Gene Golub and Charles Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

[Gooa]   Google. *Android Developers - android.net.wifi.rtt*.
            `https://developer.android.com/reference/android/net/wifi/rtt/package-summary`.
            Accessed: 2020-01-04.

[Goob]   Google. *Android Developers - Sensors Overview*.
            `https://developer.android.com/guide/topics/sensors/sensors_overview`. Accessed:
            2020-01-04.

[GSS93]   N. J. Gordon, D. J. Salmond, and A. F. M. Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE Proceedings F - Radar and Signal Processing* 140.2, 1993.

[Goy+11]   P. Goyal et al. "Strap-Down Pedestrian Dead-Reckoning System". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2011.

[Goz+11]   B. Gozick et al. "Magnetic Maps for Indoor Navigation". In: *IEEE Transactions on Instrumentation and Measurement* 60.12, 2011.

[Gra+11]   Ben Graham et al. "Analysis of the Effect of Human Presence on a Wireless Sensor Network". In: *International Journal of Ambient Computing and Intelligence (IJACI)* 3.1, 2011.

[GA01]   Mohinder Grewal and Angus Andrews. *Kalman filtering : theory and practice using MATLAB*. 2nd ed. Wiley, 2001.

[Gri18]   Geoffrey Grimmett. *Probability on Graphs : Random Processes on Graphs and Lattices*. Cambridge University Press, 2018.

[GS97]   Charles Grinstead and James Snell. *Introduction to Probability*. 2nd ed. American Mathematical Society, 1997.

[GSB05]   G. Grisettiyz, C. Stachniss, and W. Burgard. "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling". In: *IEEE International Conference on Robotics and Automation*. 2005.

[GGB12]   S. Grzonka, G. Grisetti, and W. Burgard. "A Fully Autonomous Indoor Quadrotor". In: *IEEE Transactions on Robotics* 28.1, 2012.

[GB15]   Frederico Gualberto and Heloisa Barbosa. "Factors That Influence Pedestrians Behaviour at Crossings". In: *International Conference on Mobility and Transport for Elderly and Disabled Persons*. 2015.

[Gui+16]   V. Guimarães et al. "A Motion Tracking Solution for Indoor Localization Using Smartphones". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[GH05]   André Günther and Christian Hoene. "Measuring Round Trip Times to Determine the Distance between WLAN Nodes". In: *International Conference on Research in Networking*. 2005.

[GF06]   W. Guo and N. P. Filer. "2.5D Indoor Mapping and Location-Sensing using an Impulse Radio Network". In: *IET Seminar on Ultra Wideband Systems, Technologies and Applications*. 2006.

[Gus10]   F. Gustafsson. "Particle Filter Theory and Practice with Positioning Applications". In: *IEEE Aerospace and Electronic Systems Magazine* 25.7, 2010.

[GD18]   Ralf Güting and Stefan Dieker. *Datenstrukturen und Algorithmen*. 4th ed. Springer, 2018.

[Har+03]   T. Harada et al. "Portable Orientation Estimation Device Based on Accelerometers, Magnetometers and Gyroscope Sensors for Sensor Network". In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. 2003.

[Har+13]   F. Harris et al. "An Extension of the Linkwitz-Riley Crossover Filters for Audio Systems and Their Sampled Data Implementation". In: *International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2013.

[HNR68]   P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, 1968.

[HTF09]   Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. Springer, 2009.

[Hee+11]   F. Heereman et al. "Development of path loss model for 802.11n in large conference rooms". In: *IEEE International Symposium on Antennas and Propagation (APSURSI)*. 2011.

[Hel98]   M. Held. "Efficient and Reliable Triangulation of Polygons". In: *Computer Graphics International*. 1998.

[Hel+13]   H. Hellmers et al. "An IMU/magnetometer-based Indoor positioning system using Kalman filtering". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2013.

[HS94]   John Hershberger and Jack Snoeyink. "Computing minimum length paths of a given homotopy class". In: *Computational Geometry* 4.2, 1994.

[Hes+16]   W. Hess et al. "Real-time Loop Closure in 2D LIDAR SLAM". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[Hig02]   Nicholas Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 2002.

[HB08]   H. Hile and G. Borriello. "Positioning and Orientation in Indoor Environments Using Camera Phones". In: *IEEE Computer Graphics and Applications* 28.4, 2008.

[Hil+14]   Sebastian Hilsenbeck et al. "Graph-based Data Fusion of Pedometer and WiFi Measurements for Mobile Indoor Positioning". In: *International Conference on Ubiquitous Computing (UbiComp)*. 2014.

[HA01]   Kai Hormann and Alexander Agathos. "The Point in Polygon Problem for Arbitrary Polygons". In: *Computational Geometry* 20, 2001.

[HNG94]   J. Horn, N. Nafpliotis, and D. E. Goldberg. "A Niched Pareto Genetic Algorithm for Multiobjective Optimization". In: *IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence.* 1994.

[HBS09]   Xu Huang, Mark Barralet, and Dharmendra Sharma. "Accuracy of Location Identification with Antenna Polarization on RSSI". In: *International MultiConference of Engineers and Computer Scientists.* Vol. 1. 2009.

[HS06]   Thomas Huckle and Stefan Schneider. *Numerische Methoden: Eine Einführung für Informatiker, Naturwissenschaftler, Ingenieure und Mathematiker.* 2nd ed. Springer, 2006.

[HD62]   T. Hull and A. Dobell. "Random Number Generators". In: *SIAM Review* 4.3, 1962.

[HNU99]   F. Hurtado, M. Noy, and J. Urrutia. "Flipping Edges in Triangulations". In: *Discrete & Computational Geometry* 22.3, 1999.

[Hut+16]   D. P. Hutabarat et al. "Human Tracking in Certain Indoor and Outdoor Area by Combining the use of RFID and GPS". In: *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob).* 2016.

[Ibr+18]   Mohamed Ibrahim et al. "Verification: Accuracy Evaluation of WiFi Fine Time Measurements on an Open Platform". In: *Annual International Conference on Mobile Computing and Networking (MobiCom).* 2018.

[IEE07]   IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* Institute of Electrical and Electronics Engineers, Inc. IEEE Computer Society, 2007.

[IEE12]   IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* Institute of Electrical and Electronics Engineers,f Inc. IEEE Computer Society, 2012.

[IEE16]   IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* Institute of Electrical and Electronics Engineers, Inc. IEEE Computer Society, 2016.

[IB98]   Michael Isard and Andrew Blake. "CONDENSATION - Conditional Density Propagation for Visual Tracking". In: *International Journal of Computer Vision* 29, 1998.

[Isl+16]   M. S. Islam et al. "A Low Cost MEMS and Complementary Filter Based Attitude Heading Reference System (AHRS) for Low Speed Aircraft". In: *International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*. 2016.

[JHS98]   Matthew J. Katz, Mark H. Overmars, and Micha Sharir. "Efficient Hidden Surface Removal for Objects with Small Union Size". In: *Computational Geometry* 2, 1998.

[JSZ04]   G. R. Jagadeesh, T. Srikanthan, and X. D. Zhang. "A Map Matching Method for GPS Based Real-Time Vehicle Location". In: *Journal of Navigation* 57.3, 2004.

[Jan+15]   Aleš Janota et al. "Improving the Precision and Speed of Euler Angles Computation from Low-Cost Rotation Sensor Data". In: *Sensors* 15.3, 2015.

[Jaz70]   A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. Mathematics in Science and Engineering. Elsevier Science, 1970.

[JGB14]   J. Jessup, S. N. Givigi, and A. Beaulieu. "Merging of Octree Based 3D Occupancy Grid Maps". In: *IEEE International Systems Conference Proceedings*. 2014.

[Jim+12]   A. R. Jimenez Ruiz et al. "Accurate Pedestrian Indoor Navigation by Tightly Coupling Foot-Mounted IMU and RFID Measurements". In: *IEEE Transactions on Instrumentation and Measurement* 61.1, 2012.

[Joe99]   Nigel P. Weatherill Joe F. Thompson Bharat K. Soni. *Handbook of Grid Generation*. CRC Press, 1999.

[JP04]   J. Jordana and R. Pallas-Areny. "Optimal Two-point Static Calibration of Measurement Systems with Quadratic Response". In: *IEEE Instrumentation and Measurement Technology Conference (I2MTC)*. Vol. 1. 2004.

[JPP18]   H. Ju, S. Y. Park, and C. G. Park. "A Smartphone-Based Pedestrian Dead Reckoning System With Multiple Virtual Tracking for Indoor Navigation". In: *IEEE Sensors Journal* 18.16, 2018.

[JLH11]   Sukhoon Jung, Choon-oh Lee, and Dongsoo Han. "Wi-Fi Fingerprint-based Approaches Following Log-Distance Path Loss Model for Indoor Positioning". In: *IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals*. 2011.

[Jun+12]   Wook Rak Jung et al. "Potential Risks of WiFi-based Indoor Positioning and Progress on Improving Localization Functionality". In: *ACM International Workshop on Indoor Spatial Awareness (SIGSPATIAL)*. 2012.

[ElK+10]   Kareem El-Kafrawy et al. "Propagation Modeling for Accurate Indoor WLAN RSS-Based Localization". In: *Vehicular Technology Conference (VTC)*. 2010.

[Kal10a]   Marcelo Kallmann. "Navigation Queries from Triangular Meshes". In: *Motion in Games*. 2010.

[Kal05]   Marcelo Kallmann. "Path Planning in Triangulations". In: *Workshop on Reasoning, Representation, and Learning in Computer Games, International Joint Conference on Artificial Intelligence (IJCAI)*. 2005.

[Kal10b]   Marcelo Kallmann. "Shortest Paths with Arbitrary Clearance from Navigation Meshes". In: *Eurographics / SIGGRAPH Symposium on Computer Animation (SCA)*. 2010.

[Kal60]   R. E. Kalman. "A New Approach To Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering (ASME)* 82D, 1960.

[KB61]   R. E. Kalman and R. S. Bucy. "New Results in Linear Filtering and Prediction Theory". In: *Journal of Basic Engineering* 83, 1961.

[Kar+17]   A. Karaagac et al. "Evaluation of Accurate Indoor Localization Systems in Industrial Environments". In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2017.

[Kar04]   Menelaos Karavelas. "A robust and efficient implementation for the segment Voronoi diagram". In: *International Symposium on Voronoi Diagrams in Science and Engineering*, 2004.

[KC15]   L. Kau and C. Chen. "A Smart Phone-Based Pocket Fall Accident Detection, Positioning, and Rescue System". In: *IEEE Journal of Biomedical and Health Informatics* 19.1, 2015.

[KWS12]   F. Keller, T. Willemsen, and H. Sternberg. "Calibration of Smartphones for the use in indoor navigation". In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2012.

[KSS17]   John Kessenich, Graham Sellers, and Dave Shreiner. *OpenGL programming guide : the official guide to learning OpenGL, version 4.5 with SPIR-V*. Addison-Wesley, 2017.

[KAO08]   T. J. S. Khanzada, A.R. Ali, and A.S. Omar. "Time Difference of Arrival Estimation using Super Resolution Algorithms to Minimize Distance Measurement Error for Indoor Positioning Systems". In: *IEEE International Multitopic Conference*. 2008.

[Kir+18]   M. P. R. S. Kiran et al. "A Novel System Architecture for Real-time, Robust and Accurate Step Detection for PDR based Indoor Localization". In: *World Forum on Internet of Things (WF-IoT)*. 2018.

[KGV83]   S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *SCIENCE* 220.4598, 1983.

[Kit06]   Thenu Kittappa. *Virtual Access Points*. Tech. rep. Aruba Networks, 2006.

[Kna17]   S. Knauth. "Smartphone PDR Positioning in Large Environments Employing WiFi, Particle Filter, and Backward Optimization". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2017.

[Kne18]   Ronald Kneusel. *Random Numbers and Computers*. Springer, 2018.

[KPN96]   Richard Knoblauch, Martin Pietrucha, and Marsha Nitzburg. "Field Studies of Pedestrian Walking Speed and Start-Up Time". In: *Transportation Research Record* 1538, 1996.

[KS18]   Manon Kok and Arno Solin. "Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps". In: *International Conference on Information Fusion (FUSION)*. 2018.

[KLW94]   Augustine Kong, Jun S. Liu, and Wing Hung Wong. "Sequential Imputations and Bayesian Missing Data Problems". In: *Journal of the American Statistical Association* 89.425, 1994.

[Kon09]   C Konvalin. *Compensating for Tilt, Hard-Iron, and Soft-Iron Effects*. Tech. rep. FierceElectronics, 2009.

[KGD14]   Lukas Köping, Marcin Grzegorzek, and Frank Deinzer. "Probabilistic Step and Turn Detection in Indoor Localization". In: *Conference on Data Fusion and Target Tracking: Algorithms and Applications (DFTT)*. 2014.

[KSG18]   Lukas Köping, Kimiaki Shirahama, and Marcin Grzegorzek. "A general framework for sensor-based human activity recognition". In: *Computers in Biology and Medicine* 95, 2018.

[Köp+14]   Lukas Köping et al. "Indoor Localization Using Step and Turn Detection Together with Floor Map Information". In: *FHWS Science Journal*, 2014.

[Köp+12]   Lukas Köping et al. "Indoor Navigation Using Particle Filter and Sensor Fusion". In: *Annual of Navigation* 19, 2012.

[KU94]   Arnold R. Krommer and Christoph W. Ueberhuber. *Numerical Integration on Advanced Computer Systems*. Springer, 1994.

[Kru56]   Joseph B. Kruskal. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". In: *American Mathematical Society* 7.1, 1956.

[KL51]   S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Annals of Mathematical Statistics* 22.1, 1951.

[Kus+15]   R. Kusber et al. "Direction Detection of Users Independent of Smartphone Orientations". In: *Vehicular Technology Conference (VTC2015-Fall)*. 2015.

[Led06]   Hugo Ledoux. "Modelling Three-dimensional Fields in Geoscience with the Voronoi Diagram and its Dual". PhD thesis. University of Glamorgan, 2006.

[Lee+14]   S. J. Lee et al. "Autonomous Tour Guide Robot by using Ultrasonic Range Sensors and QR code Recognition in Indoor Environment". In: *IEEE International Conference on Electro/Information Technology*. 2014.

[Lej50]   G. Lejeune Dirichlet. "Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen". In: *Journal für die reine und angewandte Mathematik* 40, 1850.

[Li+12]   Fan Li et al. "A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors". In: *International Conference on Ubiquitous Computing (UbiComp)*. 2012.

[Li+18]   Frédéric Li et al. "Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors". In: *Sensors* 18.2, 2018.

[LJ14]   J. Li and M. Johnson-Roberson. "Multi-altitude Multi-sensor Fusion Framework for AUV Exploration and Survey". In: *Oceans - St. John's*. 2014.

[Li+05]   Zang Li et al. "Robust Statistical Methods for Securing Wireless Localization in Sensor Networks". In: *International Symposium on Information Processing in Sensor Networks*. IEEE Press, 2005.

[Lin+11]   C. Lin et al. "Benchmark Dalvik and Native Code for Android System". In: *International Conference on Innovations in Bio-inspired Computing and Applications*. 2011.

[Lin15]   Joakim Lindh. *Bluetooth® low energy Beacons*. Texas Instruments. 2015.

[Liu+17]   Chen Liu et al. "Raster-To-Vector: Revisiting Floorplan Transformation". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Liu+12]   Hongbo Liu et al. "Push the Limit of WiFi based Localization for Smartphones". In: *International Conference on Mobile Computing and Networking*. 2012.

[LC98]   Jun Liu and Rong Chen. "Sequential Monte Carlo Methods for Dynamic Systems". In: *Journal of the American Statistical Association* 93, 1998.

[Liu+07]   Shih-Ping Liu et al. "Miniaturized WiFi System Module Using SiP/IPD for Handheld Device Applications". In: *International Microsystems, Packaging, Assembly and Circuits Technology*. 2007.

[LZQ15]   Y. Liu, P. Zhang, and M. Qiu. "Fast Numerical Evaluation for Symbolic Expressions in Java". In: *International Conference on High Performance Computing and Communications*. 2015.

[Loa99]   Clive Loader. *Local Regression and Likelihood*. Springer, 1999.

[LC13]   A. J. Lopez-Martin and A. Carlosena. "Sensor Signal Linearization Techniques: A Comparative Analysis". In: *IEEE Latin American Symposium on Circuits and Systems (LASCAS)*. 2013.

[Lu+94]   G. Lu et al. "Performance Analysis of A Shipborne Gyrocompass With A Multi-Antenna GPS system". In: *IEEE Position, Location and Navigation Symposium (PLANS)*. 1994.

[Lui+11]   Gough Lui et al. "Differences in RSSI Readings Made by Different Wi-Fi Chipsets: A Limitation of WLAN Localization". In: *International Conference on Localization and GNSS*. 2011.

[MK89]   Avraham Margalit and Gary D. Knott. "An algorithm for computing the union, intersection or difference of two polygons". In: *Computers & Graphics* 13.2, 1989.

[Mar51]   A. A. Markov. "The theory of algorithms". In: *Trudy Mat. Inst. Steklov.* Vol. 38. 1951.

[McC86]   Robert K. McConnell. "Method of and apparatus for pattern recognition". US 4567610A. 1986.

[McC+00]   Dennis D. McCrady et al. "Mobile Ranging Using Low-Accuracy Clocks". In: *IEEE Transactions on Microwave Theory and Techniques* 48.6, 2000.

[Mea80]   Donald Meagher. *Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. Tech. rep. Image Processing Laboratory, 1980.

[Meh70]   R. Mehra. "On the Identification of Variances and Adaptive Kalman Filtering". In: *IEEE Transactions on Automatic Control* 15.2, 1970.

[Men+11]   Wei Meng et al. "Secure and Robust Wi-Fi Fingerprinting Indoor Localization".
             In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*.
             2011.

[MMM96]   Ronald Merrill, Michael McElhinny, and Phillip McFadden. *The Magnetic Field
             of the Earth : Paleomagnetism, the Core, and the Deep Mantle*. Academic Press,
             1996.

[Met87]   Nicholas Metropolis. "The Beginning of the Monte Carlo Method". In: *Los Alamos
             Science* 15, 1987.

[MU49]   Nicholas Metropolis and Stanislaw Ulam. "The Monte Carlo Method". In: *Jour-
             nal of the American Statistical Association* 44.247, 1949.

[Mic18]   EM Microelectronic. *Bluetooth® Low-Energy Proximity Beacon With Accelerom-
             eter - EMBC22*. 2018.

[Mis18]   R von Mises. "Über die Ganzzahligkeit der Atomgewichte und verwandte Fra-
             gen". In: *Physikalische Zeitschrift* 19, 1918.

[Moi56]   Abraham de Moivre. *The Doctrine of Chances: Or, A Method of Calculating the
             Probability of Events in Play*. 3rd ed. A. Millar, 1756.

[ME85]   H. Moravec and A. Elfes. "High Resolution Maps from Wide Angle Sonar". In:
             IEEE International Conference on Robotics and Automation. Vol. 2. 1985.

[Mor88]   Hans P. Moravec. "Sensor fusion in certainty grids for mobile robots". In: *AI
             Magazine* 9, 1988.

[Mou+15]   Quentin Mourcou et al. "Performance Evaluation of Smartphone Inertial Sensors
             Measurement for Range of Motion". In: *Sensors* 15.9, 2015.

[Mur+14]   Kartik Muralidharan et al. "Barometric Phone Sensors – More Hype Than Hope!"
             In: *ACM HotMobile*. 2014.

[MS10]   C. Murphy and H. Singh. "Rectilinear Coordinate Frames for Deep Sea Naviga-
             tion". In: *IEEE/OES Autonomous Underwater Vehicles*. 2010.

[NC99]   K. Nagatani and H. Choset. "Toward Robust Sensor Based Exploration by Con-
             structing Reduced Generalized Voronoi Graph". In: *IEEE/RSJ International Con-
             ference on Intelligent Robots and Systems*. Vol. 3. 1999.

[Ndz+17]   S. G. Ndzukula et al. "A Bluetooth Low Energy based system for personnel track-
             ing". In: *Conference of the IEEE Industrial Electronics Society (IECON)*. 2017.

[ND97]   E. Nebot and H. Durrant-Whyte. "Initial calibration and alignment of an inertial navigation". In: *Conference on Mechatronics and Machine Vision in Practice*. 1997.

[NM65]   John A. Nelder and Roger Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4, 1965.

[NPM13]   P. Neto, J. N. Pires, and A. P. Moreira. "3-D Position Estimation from Inertial Sensing: Minimizing the Error from the Process of Double Integration of Accelerations". In: *Conference of the IEEE Industrial Electronics Society (IECON)*. 2013.

[Neu51]   John von Neumann. "Various Techniques Used in Connection With Random Digits". In: *National Bureau of Standards*, 1951. Summary by G. E. Forsythe.

[NS80]   Martin Newell and Carlo Sequin. "The Inside Story on Self-Intersecting Polygons". In: *Lambda* 1.2, 1980.

[New67]   Isaac Newton. "Methodus fluxionum et serierum infinitarum, 1671". In: *The method of fluxions and infinite series*, 1967.

[Ngu+16]   Phong Nguyen et al. "User-Friendly Heading Estimation for Arbitrary Smartphone Orientations". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016.

[Nie83]   Heinrich Niemann. *Klassifikation von Mustern*. Springer, 1983.

[Nis+09]   T. Nishida et al. "Dynamic State Estimation Using Particle Filter and Adaptive Vector Quantizer". In: *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 2009.

[NOA]   NOAA/NCEI/CIRES. *World Magnetic Model 2019*. `https://www.ngdc.noaa.gov/geomag/WMM/`. Accessed: 2020-01-04.

[Nuc+04]   A. Nuchter et al. "6D SLAM with an Application in Autonomous Mine Mapping". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 2004.

[NRP16]   Henri Nurminen, Matti Raitoharju, and Robert Piche. "An efficient indoor positioning particle filter using a floor-plan based proposal distribution". In: *International Conference on Information Fusion (FUSION)*. 2016.

[OAS76]   National Oceanic, National Aeronautics Atmospheric Administration, and United States Air Force Space Administration. *U.S. Standard Atmosphere*. 1976.

[Och+14]   M. Ochiai et al. "A study on indoor position estimation based on fingerprinting us-
            ing GPS signals". In: *International Conference on Indoor Positioning and Indoor
            Navigation (IPIN)*. 2014.

[Oga11]   Clement Ogaja. *Applied GPS for Engineers and Project Managers*. American So-
            ciety of Civil Engineers, 2011.

[OCV12]   U. Olgun, C. Chen, and J. L. Volakis. "Efficient Ambient WiFi Energy Harvesting
            Technology and its Applications". In: *IEEE International Symposium on Anten-
            nas and Propagation*. 2012.

[Ols+16]   F. Olsson et al. "Accelerometer calibration using sensor fusion with a gyroscope".
            In: *IEEE Statistical Signal Processing Workshop (SSP)*. 2016.

[Ope]   OpenStreetMap. *Indoor Mapping*.
            `https://wiki.openstreetmap.org/wiki/Indoor_Mapping`. Accessed: 2020-01-04.

[Pac+95]   Scott Pace et al. *The Global Positioning System : Assessing National Policies*.
            RAND, 1995.

[Pal+11]   Ravishankar Palaniappan et al. "Autonomous RF Surveying Robot for Indoor Lo-
            calization and Tracking". In: *International Conference on Indoor Positioning and
            Indoor Navigation (IPIN)*. 2011.

[PHU09]   Ambili T. Parameswaran, Mohammad I. Husain, and Shambhu Upadhyaya. "Is
            RSSI a Reliable Parameter in Sensor Localization Algorithms: An Experimental
            Study". In: *Field Failure Data Analysis Workshop (F2DA)*. 2009.

[PHP17]   S. Y. Park, S. J. Heo, and C. G. Park. "Accelerometer-based Smartphone Step
            Detection Using Machine Learning Technique". In: *International Electrical En-
            gineering Congress (iEECON)*. 2017.

[Par+06]   D. L. Partin et al. "Temperature Stable Hall Effect Sensors". In: *IEEE Sensors
            Journal* 6.1, 2006.

[Par62]   Emanuel Parzen. "On Estimation of a Probability Density Function and Mode".
            In: *The Annals of Mathematical Statistics* 33.3, 1962.

[PW09]   Anindya Paul and Eric Wan. "RSSI-Based Indoor Localization and Tracking Us-
            ing Sigma-Point Kalman Smoothers". In: *Selected Topics in Signal Processing* 3,
            2009.

[Pea95]   Karl Pearson. "Contributions to the Mathematical Theory of Evolution II. Skew
            Variation in Homogeneous Material". In: *Philosophical Transactions of the Royal
            Society of London* 186, 1895.

[Pen55]   R. Penrose. "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 51.3, 1955.

[PPG05]   J. M. Dias Pereira, O. Postolache, and P. Silva Girao. "Adaptive Self-Calibration Algorithm for Smart Sensors Linearization". In: *IEEE Instrumentationand Measurement Technology Conference Proceedings*. Vol. 1. 2005.

[Pit32]   Henri Pitot. "Description d'une Machine pour mesurer la vitesse des Eaux courantes, et le sillage des Vaisseaux". In: *Histoire de l'Académie royale des sciences*, 1732.

[PC94]   D. S. Polydorou and C. N. Capsalis. "A new path loss prediction statistical model for indoor wireless communications". In: *International Journal of Infrared and Millimeter Waves* 15.1, 1994.

[Pop+07]   D. R. Popovic et al. "Three-Axis Teslameter With Integrated Hall Probe". In: *IEEE Transactions on Instrumentation and Measurement* 56.4, 2007.

[Pow62]   Michael J. D. Powell. "An iterative method for finding stationary values of a function of several variables". In: *The Computer Journal* 5.2, 1962.

[Rac07]   Steve Rackley. *Wireless Networking Technology: From Principles to Successful Implementation*. Elsevier, Newnes, 2007.

[Raj+96]   A. Rajkumar et al. "Predicting RF coverage in large environments using ray-beam tracing and partitioning tree represented geometry". In: *Wireless Networks* 2.2, 1996.

[Rap02]   Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2002.

[Rec73]   Ingo Rechenberg. *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[RHG13]   Rüdiger Reinhardt, Armin Hoffmann, and Tobias Gerlach. *Nichtlineare Optimierung: Theorie, Numerik und Experimente*. Springer, 2013.

[Ric+00]   Elodie Richalot et al. "Electromagnetic Propagation into Reinforced-Concrete Walls". In: *Microwave Theory and Techniques* 48.3, 2000.

[RRF02]   Robert Riener, Marco Rabuffetti, and Carlo Frigo. "Stair ascent and descent at different inclinations". In: *Gait & Posture* 15.1, 2002.

[RM00]   Roerdink and Meijster. "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies". In: *Fundamenta Informatica (FUNDINF)* 41, 2000.

[Roo+02]   Teemu Roos et al. "A Probabilistic Approach to WLAN User Location Estima-
            tion". In: *International Journal of Wireless Information Networks* 9.3, 2002.

[Ror93]   C Rorabaugh. *Digital Filter Designer's Handbook : Featuring C Routines*. New
            York: McGraw-Hill, 1993.

[Ros56]   Murray Rosenblatt. "Remarks on Some Non-Parametric Estimates of a Density
            Function". In: *The Annals of Mathematical Statistics* 27, 1956.

[RC01]   Yong Rui and Yunqiang Chen. "Better Proposal Distributions: Object Tracking
            Using Unscented Particle Filter". In: *IEEE Computer Society Conference on Com-
            puter Vision and Pattern Recognition (CVPR)*. Vol. 2. 2001.

[Sär13]   Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press,
            2013.

[Sau+11]   Nicolas Saunier et al. "Estimation of Frequency and Length of Pedestrian Stride
            in Urban Environments with Video Sensors". In: *Transportation Research Record*
            2264.1, 2011.

[Sch17]   Harald Scheid. *Elemente der Geometrie*. Springer, 2017.

[Sch+12]   S. Schmitt et al. "A Reference System for Indoor Localization Testbeds". In:
            International Conference on Indoor Positioning and Indoor Navigation (IPIN).
            2012.

[SBW12]   Felix Scholkmann, Jens Boss, and Martin Wolf. "An Efficient Algorithm for Au-
            tomatic Peak Detection in Noisy Periodic and Quasi-Periodic Signals". In: *Algo-
            rithms* 5.4, 2012.

[Sch77]   Hans-Paul Schwefel. *Numerische Optimierung von Computor-Modellen mittels
            der Evolutionsstrategie*. Birkhäuser, 1977.

[SR92]   Scott Y. Seidel and Theodore S. Rappaport. "914 MHz Path Loss Prediction Mod-
            els for Indoor Wireless Communications in Multifloored Buildings". In: *IEEE
            Transactions on Antennas and Propagation* 40.2, 1992.

[Sen+12]   Souvik Sen et al. "You are facing the Mona Lisa: spot localization using PHY
            layer information". In: *International Conference on Mobile Systems, Applica-
            tions, and Services*. ACM, 2012.

[Ser28]   J. H. Service. "Radio Acoustic Position Finding in Hydrography". In: *Journal of
            the A.I.E.E.* 47.9, 1928.

[Sey05]   John S. Seybold. *Introduction to RF Propagation*. Wiley-Interscience, 2005.

[SJP13]   Jiten Shah, G.J. Joshi, and Purnima Parida. "Behavioral Characteristics of Pedestrian Flow on Stairway at Railway Station". In: *Procedia - Social and Behavioral Sciences* 104, 2013.

[SA11]   A. Shahzada and K. Askar. "Dynamic Vehicle Navigation: An A* Algorithm Based Approach Using Traffic and Road Information". In: *IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*. 2011.

[SCI13]   W Shane Grant, Randolph C. Voorhies, and Laurent Itti. "Finding Planes in LiDAR Point Clouds for Real-Time Registration". In: *International Conference on Intelligent Robots and Systems (IEEE/RSJ)*. 2013.

[She+09]   A. Sheinker et al. "Magnetic Anomaly Detection Using a Three-Axis Magnetometer". In: *IEEE Transactions on Magnetics* 45.1, 2009.

[SS09]   Hideaki Shimazaki and Shigeru Shinomoto. "Kernel bandwidth optimization in spike rate estimation". In: *Computational Neuroscience* 29, 2009.

[Shi03]   Peter Shirley. *Realistic Ray Tracing*. 2nd ed. AK Peters, 2003.

[Shu+15]   Y. Shu et al. "Magicol: Indoor Localization Using Pervasive Magnetic Field and Opportunistic WiFi Sensing". In: *IEEE Journal on Selected Areas in Communications* 33.7, 2015.

[Sie04]   Roland Siegwart. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[Sil86]   B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[SSM62]   Gerald L. Smith, Stanley F. Schmidt, and Leonard A. McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. Tech. rep. National Aeronautics and Space Administration, 1962.

[SGM14]   I. M. Smith, D.V. Griffiths, and L. Margetts. *Programming the Finite Element Method*. Wiley, 2014.

[Smi11]   Julius Smith. *Spectral Audio Signal Processing*. W3K, 2011.

[Smi99]   Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2nd ed. California Technical Publishing, 1999.

[SD16]   C. Soaz and K. Diepold. "Step Detection and Parameterization for Gait Assessment Using a Single Waist-Worn Accelerometer". In: *IEEE Transactions on Biomedical Engineering* 63.5, 2016.

[SB13]   D. Sonowal and M. Bhuyan. "Linearizing Thermistor Characteristics by Piece-
              wise Linear Interpolation in Real Time FPGA". In: *International Conference on
              Advances in Computing, Communications and Informatics (ICACCI)*. 2013.

[SMV15]   R. Soorat, K. Madhuri, and A. Vudayagiri. "Hardware Random number Generator
              for cryptography". In: *arXiv e-prints*, 2015.

[ST14]   Sara Stančin and Sašo Tomažič. "Time- and Computation-Efficient Calibration of
              MEMS 3D Accelerometers and Gyroscopes". In: *Sensors* 14.8, 2014.

[Stu15]   Roland Stull. *Practical Meteorology - An Algebra-based Survey of Atmospheric
              Science*. University of British Columbia, 2015.

[SND99]   S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte. "A High Integrity IMU/GPS
              Navigation Loop for Autonomous Land Vehicle Applications". In: *IEEE Trans-
              actions on Robotics and Automation* 15.3, 1999.

[TY09]   Tomoji Takasu and Akio Yasuda. "Development of the low-cost RTK-GPS re-
              ceiver with an open source program package RTKLIB". In: *International Sympo-
              sium on GPS/GNSS*, Jan. 2009.

[TG91]   Yordphol Tanaboriboon and Jocelyn Guyano. "Analysis of Pedestrian Movements
              in Bangkok". In: *Transportation Research Record* 1294, 1991.

[TT08]   Barry N. Taylor and Ambler Thompson, eds. *The International System of Units
              (SI)*. National Institute of Standards and Technology, 2008.

[Tay15]   Brook Taylor. *Methodus incrementorum directa et inversa*. Londini, 1715.

[Thr03]   Sebastian Thrun. "Learning Occupancy Grid Maps With Forward Sensor Mod-
              els". In: *Autonomous Robots* 15, 2003.

[TBF05]   Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The
              MIT Press, 2005.

[Tia+02]   Jing Tian et al. "Graph-Based Mobility Model for Mobile Ad Hoc Network Sim-
              ulation". In: *Annual Simulation Symposium*. 2002.

[Tia+15]   Qinglin Tian et al. "A Hybrid Indoor Localization and Navigation System with
              Map Matching for Pedestrians Using Smartphones". In: *Sensors* 15.12, 2015.

[Tim82]   John Timbs. *Wonderful Inventions: From the Mariner's Compass to the Electric
              Telegraph Cable*. G. Routledge and Sons, 1882.

[Tit11]   Peter Tittmann. *Graphentheorie : Eine anwendungsorientierte Einführung*.
              Hanser, 2011.

[Tor+17]   Joaquín Torres-Sospedra et al. "The Smartphone-Based Offline Indoor Location Competition at IPIN 2016: Analysis and Future Work". In: *Sensors* 17.3, 2017.

[Tor44]   Evangelista Torricelli. *Opera geometrica*. Typis Amatoris Massæ & Laurentij de Landis, 1644.

[TS12]   K. Tumkur and S. Subbiah. "Modeling Human Walking for Step Detection and Stride Determination by 3-Axis Accelerometer Readings in Pedometer". In: *International Conference on Computational Intelligence, Modelling and Simulation*. 2012.

[UN94]   Naonori Ueda and Ryohei Nakano. "A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers". In: *Neural Networks* 7.8, 1994.

[VF13]   A. A. Vasilakis and I. Fudos. "Depth-Fighting Aware Methods for Multifragment Rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 19.6, 2013.

[Vat92]   Bala R. Vatti. "A Generic Solution to Polygon Clipping". In: *Commun. ACM* 35.7, 1992.

[VDIa]   VDI/VDE/DGQ/DKD 2622. *Kalibrieren von Messmitteln für elektrische Größen*.

[VDIb]   VDI/VDE/DGQ/DKD 2622 Blatt 2. *Kalibrieren von Messmitteln für elektrische Größen - Methoden zur Ermittlung der Messunsicherheit*.

[Vin17]   John Vince. *Mathematics for Computer Graphics*. Springer, 2017.

[VS91]   Lee Vincent and Pierre Soille. "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations". In: *IEEE PAMI* 13.6, 1991.

[Vin75]   T. Vincenty. "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations". In: *Survey Review* 23.176, 1975.

[Vor08]   Georges Voronoi. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques". In: *Journal für die reine und angewandte Mathematik* 134, 1908.

[WGN15]   A. Wahdan, J. Georgy, and A. Noureldin. "Three-Dimensional Magnetometer Calibration With Small Space Coverage for Pedestrians". In: *IEEE Sensors Journal* 15.1, 2015.

[Wan11]   C. C. L. Wang. "Approximate Boolean Operations on Large Polyhedral Solids with Partial Mesh Reconstruction". In: *IEEE Transactions on Visualization and Computer Graphics* 17.6, 2011.

[Wan+11a]   Fasheng Wang et al. "Particle Filter with Hybrid Proposal Distribution for Non-linear State Estimation". In: *Journal of Computers* 6, 2011.

[Wan+11b]   Jie Wang et al. "Differential radio map-based robust indoor localization". In: *EURASIP Journal on Wireless Communications and Networking*, 2011.

[Wes50]   G. P. De Westfelt. "Motor Design for a High-Speed Electrically-Driven Gyroscope". In: *Electrical Engineering* 69.5, 1950.

[WT06]   Arthur Williams and Fred Taylor. *Electronic Filter Design Handbook*. McGraw-Hill, 2006.

[Wil02]   Robert Wilson. *Propagation Losses Through Common Building Materials 2.4 GHz vs 5 GHz*. 2002.

[Wit19]   Alexandra Witze. "Earth's magnetic field is acting up". In: *Nature* 565, 2019.

[WH08]   Oliver Woodman and Robert Harle. "Pedestrian Localisation for Indoor Environments". In: *International Conference on Ubiquitous Computing (UbiComp)*. 2008.

[Wu10]   H. Wu. "Survey on three-dimensional spatial data models in GIS". In: *Conference on Environmental Science and Information Application Technology*. Vol. 2. 2010.

[XYH09]   Wang Xiaoling, Zhao Yan, and Wang Hong. "Non-conduct Steering Sensor for Electric Power Steering". In: *International Conference on Information and Automation*. 2009.

[Xu+13]   C. Xu et al. "SCPL: Indoor Device-Free Multi-Subject Counting and Localization Using Radio Signal Strength". In: *International Conference on Information Processing in Sensor Networks (IPSN)*. 2013.

[Yan06]   W. Yanbing. "3D GIS Spatial Modeling for City Surface and Subsurface Integration". In: *IEEE International Symposium on Geoscience and Remote Sensing*. 2006.

[YWL12]   Zheng Yang, Chenshu Wu, and Yunhao Liu. "Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention". In: *International Conference on Mobile Computing and Networking*. 2012.

[Ye+14]   H. Ye et al. "B-Loc: Scalable Floor Localization Using Barometer on Smartphone". In: *International Conference on Mobile Ad Hoc and Sensor Systems*. 2014.

[YN01]   S. You and U. Neumann. "Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration". In: *IEEE Virtual Reality*. 2001.

[YAS03]   Moustafa A. Youssef, Ashok Agrawala, and A. Udaya Shankar. "WLAN Location Determination via Clustering and Probability Distributions". In: *IEEE International Conference on Pervasive Computing and Communications*. 2003.

[YAA05]   Moustafa Youssef, Mohamed Abdallah, and Ashok Agrawala. *Multivariate Analysis for Probabilistic WLAN Location Determination Systems*. 2005.

[YA05]   Moustafa Youssef and Ashok Agrawala. "The Horus WLAN Location Determination System". In: *International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2005.

[YMA07]   Moustafa Youssef, Matthew Mah, and Ashok Agrawala. "Challenges: Device-free Passive Localization for Wireless Environments". In: *Annual International Conference on Mobile Computing and Networking (MobiCom)*. 2007.

[Yu+19]   Yue Yu et al. "A Robust Dead Reckoning Algorithm Based on Wi-Fi FTM and Multiple Sensors". In: *Remote Sensing* 11.5, 2019.

[Zha+18a]   J. Zhang et al. "A Complex User Activity Recognition Algorithm based on Convolutional Neural Networks". In: *Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*. 2018.

[Zha+18b]   S. Zhang et al. "Indoor 2.5D Positioning of WiFi Based on SVM". In: *Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*. 2018.

[Zha+15]   Yizhong Zhang et al. "Online Structure Analysis for Real-Time Indoor Scene Reconstruction". In: *ACM Trans. Graph.* 34.5, 2015.

[ZY15]   Z. Zhang and G. Yang. "Micromagnetometer Calibration for Accurate Orientation Estimation". In: *IEEE Transactions on Biomedical Engineering* 62.2, 2015.

[Zha+11]   Zengbin Zhang et al. "I Am the Antenna: Accurate Outdoor AP Location using Smartphones". In: *International Conference on Mobile Computing and Networking*. 2011.

[Zho+15]   B. Zhou et al. "Activity Sequence-Based Indoor Pedestrian Localization Using Smartphones". In: *IEEE Transactions on Human-Machine Systems* 45.5, 2015.

[Zho+12]   Pengfei Zhou et al. "IODetector: A Generic Service for Indoor Outdoor Detection". In: *ACM Conference on Embedded Network Sensor Systems (SenSys)*. 2012.

Appendix

A.1 Tilt Compensation Example

The following example describes the tilt compensation process, based on the complementary filter, introduced in section 2.4.2. The smartphone is assumed to be held upfront in landscape mode, at an angle of $30°$ (see figure 2.5). Readings from the accelerometer are assumed as

$$\boldsymbol{a}_{[0]} = (0.10, 4.90, 8.40)^T \, .$$

According to (2.46), the corresponding quaternion $Q^a_{[0]}$ is

$$\alpha = \cos^{-1} \left(\frac{(0.10, 4.90, 8.40)^T}{\|(0.10, 4.90, 8.40)^T\|} \bullet (0, 0, 1)^T \right) \approx \cos^{-1}(0.8637) \approx 30.26° \, ,$$

$$\boldsymbol{u} = (0.10, 4.90, 8.40)^T \times (0, 0, 1)^T = (4.90, -0.10, 0.00)^T \, ,$$

$$Q^a_{[0]} \approx \left(30.26°, \frac{(4.90, -0.10, 0.00)^T}{\|(4.90, -0.10, 0.00)^T\|} \right)$$

$$\approx \left(30.26°, (1.00, -0.02, 0.00)^T \right)$$

$$\approx \left(\cos(15.13°) + \sin(15.13°)(1.00\mathbf{i} - 0.02\mathbf{j} + 0.00\mathbf{k}) \right)$$

$$\approx (0.965 + 0.261\mathbf{i} - 0.005\mathbf{j} + 0.000\mathbf{k}) \, .$$

For visualization, this quaternion is used to initialize the complementary filter (2.49), and thus $\hat{Q}_{[0]} = Q^a_{[0]}$. The pedestrian now increases the tilt angle, rotating with $\approx 10°/\text{s}$ around the device's x-axis. Assuming a sample rate of $2\,\text{Hz}$, new (noisy) accelerometer and (more robust) gyroscope readings are observed after $500\,\text{ms}$, denoting the change of the device's pose

$$\boldsymbol{a}_{[1]} = (0.98, 5.89, 7.85)^T \, , \quad \boldsymbol{\omega'}_{[1]} = (0.17, 0.01, 0.02)^T \, , \quad \Delta t = 500\,\text{ms} \, .$$

The new quaternion for the accelerometer is thus as follows

$$Q^a_{[1]} \approx \left(37.26°, (0.99, -0.16, 0.00)^T \right) \approx (0.948 + 0.315\mathbf{i} - 0.052\mathbf{j} + 0.000\mathbf{k}) \, .$$

According to (2.47), the quaternion for the gyroscope's turn-rate is given by

$$Q_{[1]}^{\omega'} \approx \left(4.91°, (0.99, 0.06, 0.12)^T\right)$$
$$\approx \left(\cos(2.46°) + \sin(2.46°)(0.99\mathbf{i} + 0.06\mathbf{j} + 0.12\mathbf{k})\right)$$
$$\approx (0.999 + 0.042\mathbf{i} + 0.002\mathbf{j} + 0.005\mathbf{k}) .$$

All quaternions can now be fused by the complementary filter (2.49). For the presented example, a mixing rate of $\kappa = 0.9$ is used

$$\hat{Q}_{[1]} \approx 0.9\left(Q_{[1]}^{\omega'}\hat{Q}_{[0]}\right) + 0.1Q_{[1]}^{a}$$
$$\approx 0.9\left((0.999 + 0.042\mathbf{i} + 0.002\mathbf{j} + 0.005\mathbf{k})(0.965 + 0.261\mathbf{i} - 0.005\mathbf{j} + 0.000\mathbf{k})\right) + 0.1Q_{[1]}^{a}$$
$$\approx 0.9\left((0.953 + 0.302\mathbf{i} - 0.004\mathbf{j} + 0.006\mathbf{k})\right) + 0.1\,(0.948 + 0.315\mathbf{i} - 0.052\mathbf{j} + 0.000\mathbf{k})$$
$$\approx (0.858 + 0.272\mathbf{i} - 0.004\mathbf{j} + 0.005\mathbf{k}) + (0.095 + 0.032\mathbf{i} - 0.005\mathbf{j} + 0.000\mathbf{k})$$
$$\approx (0.953 + 0.303\mathbf{i} - 0.009\mathbf{j} + 0.005\mathbf{k})$$
$$\approx \left(35.35°, (1.00, -0.03, 0.02)^T\right) .$$

As can be seen, the gyroscope corrected the noisy observation from the accelerometer, and the estimated angle is close to the actual $35°$ ($30° + 10\,°/\text{s} \cdot 500\,\text{ms}$). The complementary filter's quaternion $\hat{Q}_{[1]}$ is then converted into the tilt compensation matrix [Die06]

$$\mathbf{R}^{-1} \approx \begin{pmatrix} 1.00 & -0.02 & -0.01 \\ 0.00 & 0.82 & -0.58 \\ 0.02 & 0.58 & 0.82 \end{pmatrix} .$$

If the pedestrian now turns with $\approx 35\,°/\text{s}$ while holding the smartphone in the aforementioned pose, the gyroscope readings are approximately

$$\boldsymbol{\omega'} = (0.02, 0.36, 0.50)^T ,$$

where the $35\,°/\text{s}$ are divided into y and z. When transformed by $\mathbf{R}^{-1}$, this yields

$$\mathbf{R}^{-1}\boldsymbol{\omega'} \approx (0.01, 0.01, 0.62)^T \approx (0.43°, 0.29°, 35.32°)^T ,$$

where solely the z-axis contains the pedestrian's turn rate.

A.2   Step-Detection Filters

IIR parameters and FIR kernels used for the filters applied to the accelerometer's magnitude, before performing step-detection (see section 2.4.1).

	a_0	a_1	a_2	b_0	b_1	b_2
IIR low-pass	1.0000	-1.7347	0.7660	0.0078	0.0156	0.0078
IIR band-pass	1.0000	-1.9047	0.9198	0.0401	0.0000	-0.0401
IIR HLL-pass (low)	1.0000	-1.7347	0.7660	0.0078	0.0156	0.0078
IIR HLL-pass (high)	1.0000	-1.9112	0.9150	0.9565	-1.9131	0.9565

Table A.1: IIR filter parameters for step-detection (see (2.25) and table 6.3)

$$
\begin{aligned}
\boldsymbol{k} = (\,&0.0012,\ 0.0019,\ 0.0034,\ 0.0060,\ 0.0101,\ 0.0157,\ 0.0229,\ 0.0313,\\
&0.0406,\ 0.0501,\ 0.0592,\ 0.0673,\ 0.0735,\ 0.0775,\ 0.0789,\ 0.0775,\\
&0.0735,\ 0.0673,\ 0.0592,\ 0.0501,\ 0.0406,\ 0.0313,\ 0.0229,\ 0.0157,\\
&0.0101,\ 0.0060,\ 0.0034,\ 0.0019,\ 0.0012\,)
\end{aligned}
$$

Table A.2: FIR low-pass kernel for step-detection (see table 6.3)

$$
\begin{aligned}
\boldsymbol{k} = (\,&0.0000,\ \ \ 0.0000,\ \ \ 0.0000,\ \ \ 0.0001,\ \ \ 0.0002,\ \ \ 0.0003,\ \ \ 0.0005,\ \ \ 0.0007,\\
&0.0010,\ \ \ 0.0014,\ \ \ 0.0018,\ \ \ 0.0022,\ \ \ 0.0028,\ \ \ 0.0033,\ \ \ 0.0039,\ \ \ 0.0045,\\
&0.0050,\ \ \ 0.0054,\ \ \ 0.0057,\ \ \ 0.0057,\ \ \ 0.0056,\ \ \ 0.0052,\ \ \ 0.0044,\ \ \ 0.0033,\\
&0.0019,\ \ \ 0.0000,\ -0.0022,\ -0.0048,\ -0.0078,\ -0.0110,\ -0.0144,\ -0.0179,\\
&-0.0215,\ -0.0250,\ -0.0283,\ -0.0314,\ -0.0340,\ -0.0361,\ -0.0376,\ -0.0384,\\
&-0.0385,\ -0.0377,\ -0.0362,\ -0.0338,\ -0.0307,\ -0.0268,\ -0.0223,\ -0.0172,\\
&-0.0117,\ -0.0059,\ -0.0000,\ \ \ 0.0059,\ \ \ 0.0117,\ \ \ 0.0172,\ \ \ 0.0223,\ \ \ 0.0268,\\
&0.0307,\ \ \ 0.0338,\ \ \ 0.0362,\ \ \ 0.0377,\ \ \ 0.0385,\ \ \ 0.0384,\ \ \ 0.0376,\ \ \ 0.0361,\\
&0.0340,\ \ \ 0.0314,\ \ \ 0.0283,\ \ \ 0.0250,\ \ \ 0.0215,\ \ \ 0.0179,\ \ \ 0.0144,\ \ \ 0.0110,\\
&0.0078,\ \ \ 0.0048,\ \ \ 0.0022,\ \ \ 0.0000,\ -0.0019,\ -0.0033,\ -0.0044,\ -0.0052,\\
&-0.0056,\ -0.0057,\ -0.0057,\ -0.0054,\ -0.0050,\ -0.0045,\ -0.0039,\ -0.0033,\\
&-0.0028,\ -0.0022,\ -0.0018,\ -0.0014,\ -0.0010,\ -0.0007,\ -0.0005,\ -0.0003,\\
&-0.0002,\ -0.0001,\ -0.0000,\ -0.0000,\ -0.0000\,)
\end{aligned}
$$

Table A.3: FIR band-pass kernel for step-detection (see table 6.3)

A.3   Additionally Used Maps

Overview on maps that were used additionally (see table 6.15 and 6.16), besides the three
buildings introduced in section 6.1. The two maps from figure A.1 and A.2 were modeled
for attending the *IPIN 2016 Indoor Localization Competition*, and belong to the *University
of Alcalá de Henares*. The one from figure A.3 was modeled for an upcoming experimental
deployment within the *Hutmuseum* in Lindenberg.
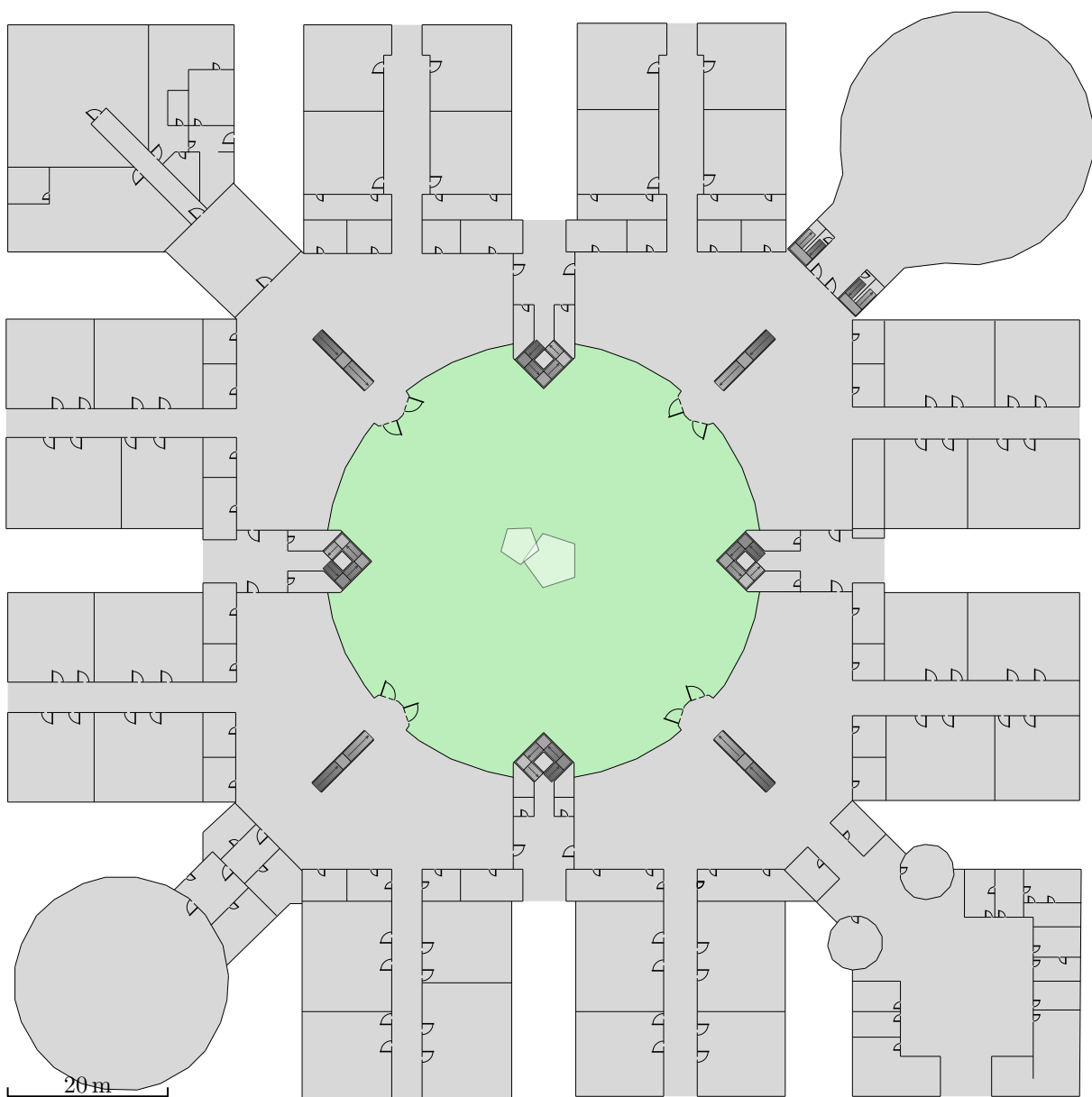
Figure A.1: UAH – Building of the *University of Alcalá de Henares*, first floor.
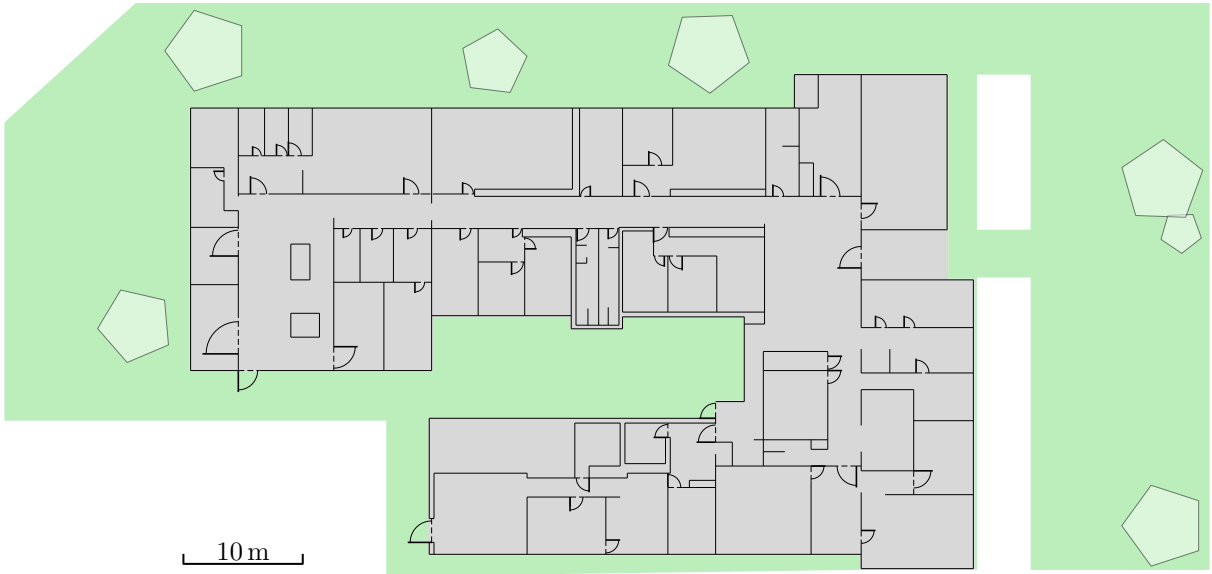
Figure A.2: CAR – Building of the *University of Alcalá de Henares*.
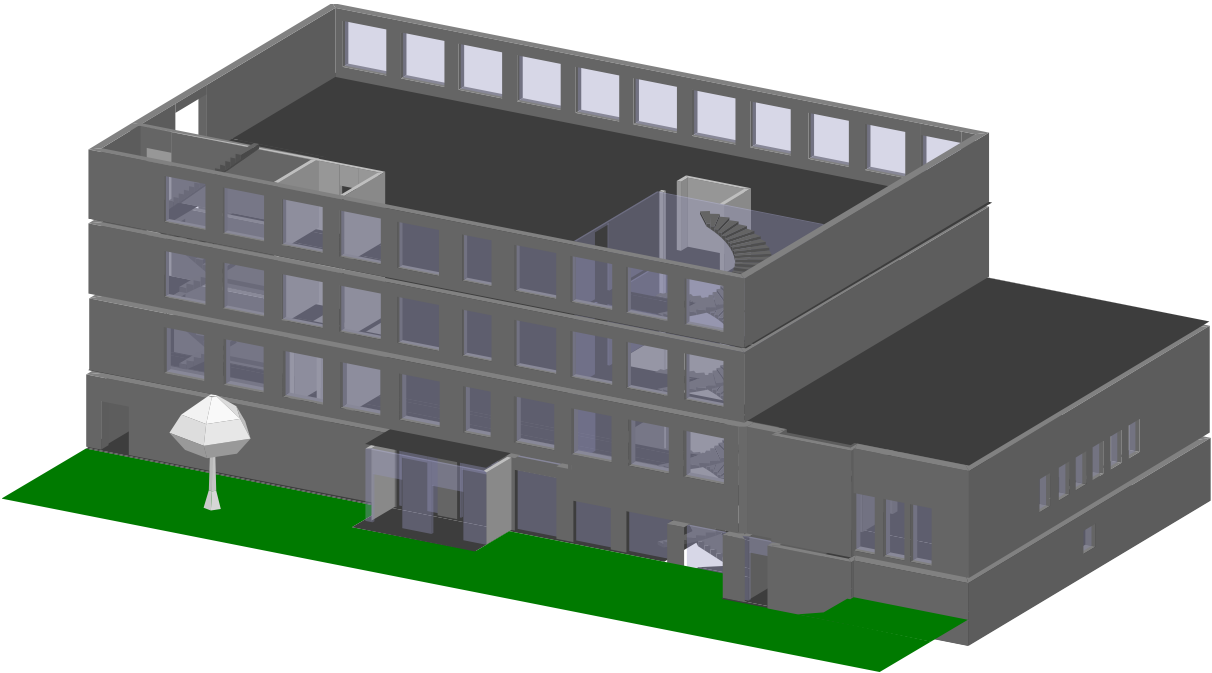
Figure A.3: Museum 3 – *Hutmuseum* in Lindenberg.

A.4   Final System Results

Excerpt of all results stemming from the final system, listing one instance for every conducted walk (see section 6.1). The images on the left depict the buildings, with their floors stretched for increased visibility, the estimated paths shown in blue, and the corresponding ground truth in black. The plots on the right denote the 3D error between estimation and ground truth, and its behavior over time. The additionally added horizontal gray line denotes the average error throughout the walk.

Museum 1
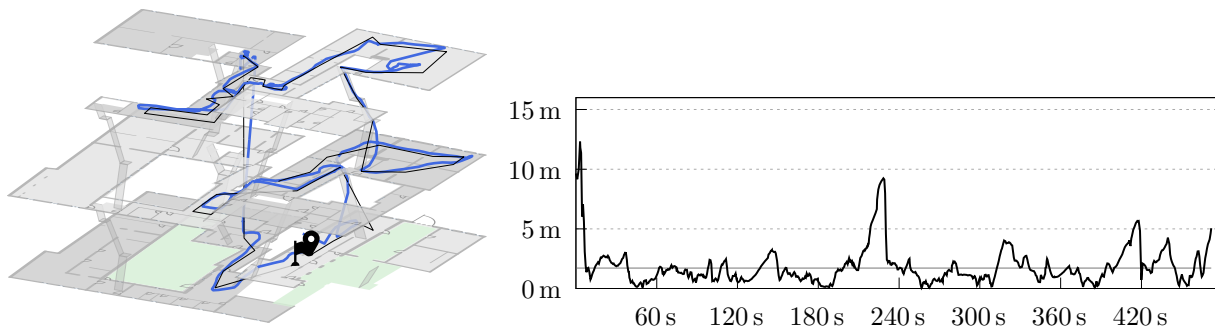
Figure A.4: Estimation result for walk B1, when using the final system.
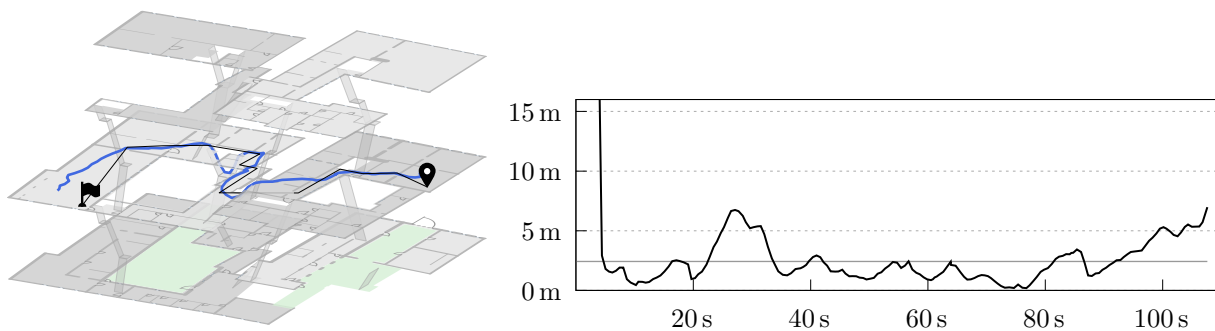
Figure A.5: Estimation result for walk B2, when using the final system.
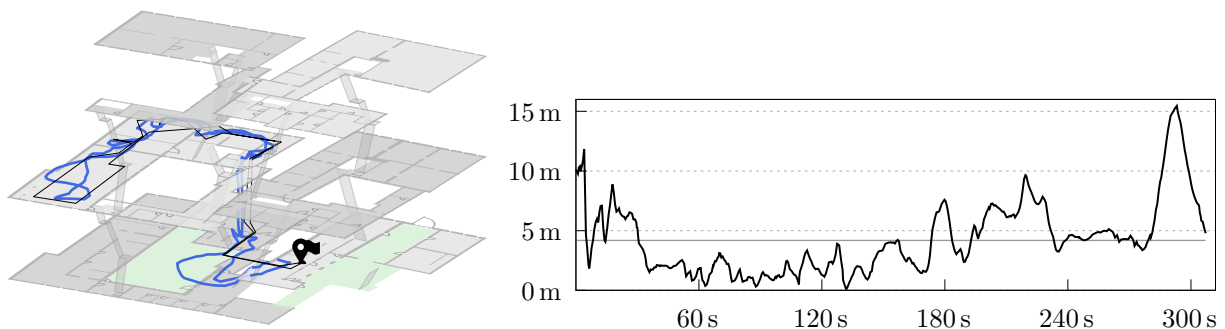
Figure A.6: Estimation result for walk B3, when using the final system.

SHL
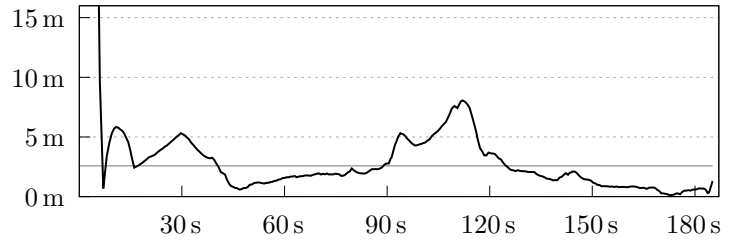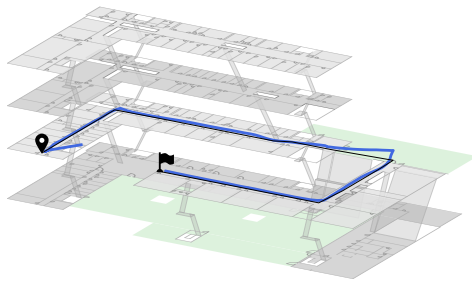
Figure A.7: Estimation result for walk A1, when using the final system.
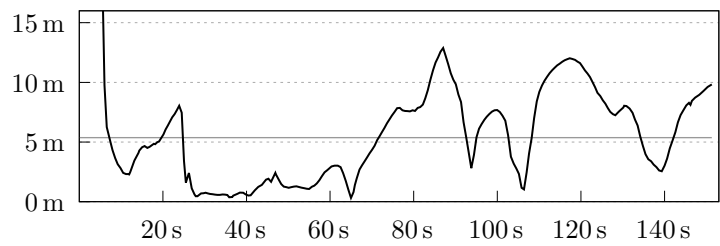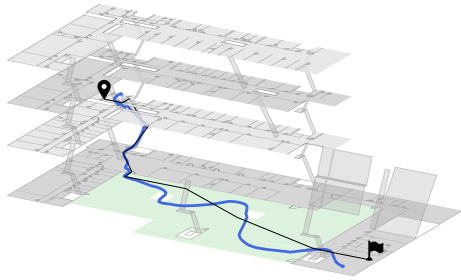
Figure A.8: Estimation result for walk A2, when using the final system.
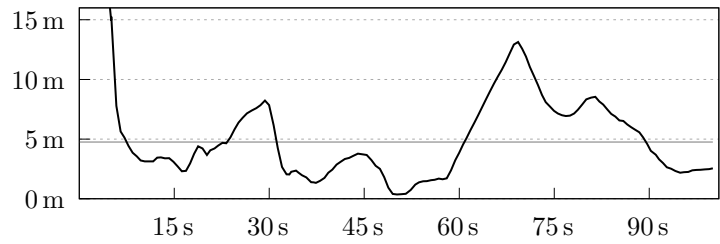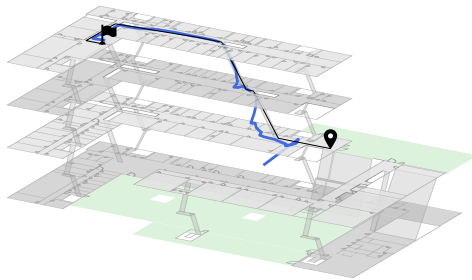
Figure A.9: Estimation result for walk A3, when using the final system.
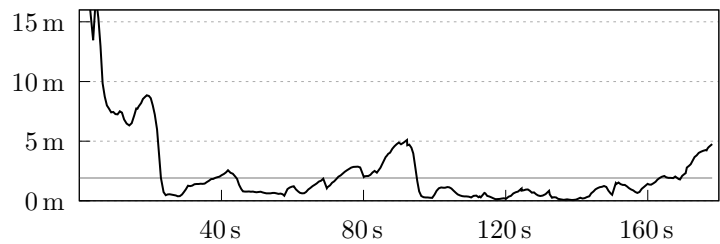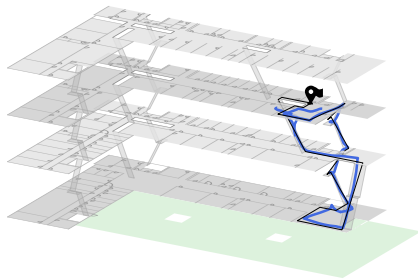
Figure A.10: Estimation result for walk A4, when using the final system.
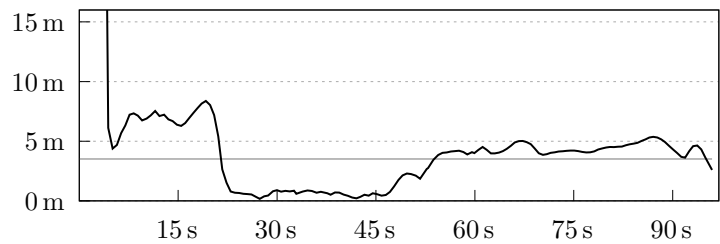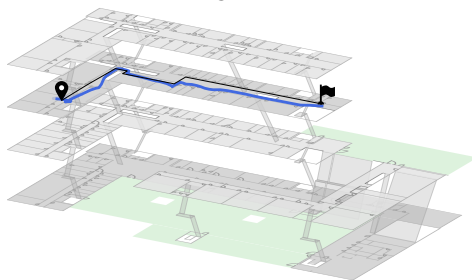
Figure A.11: Estimation result for walk A5, when using the final system.

Museum 2
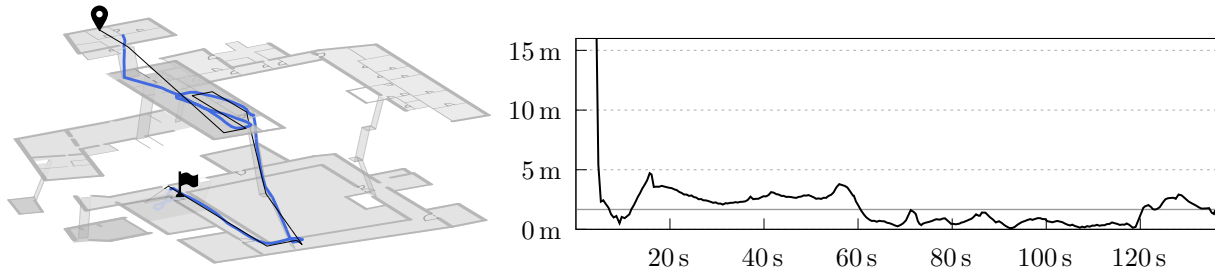
Figure A.12: Estimation result for walk C1, when using the final system.
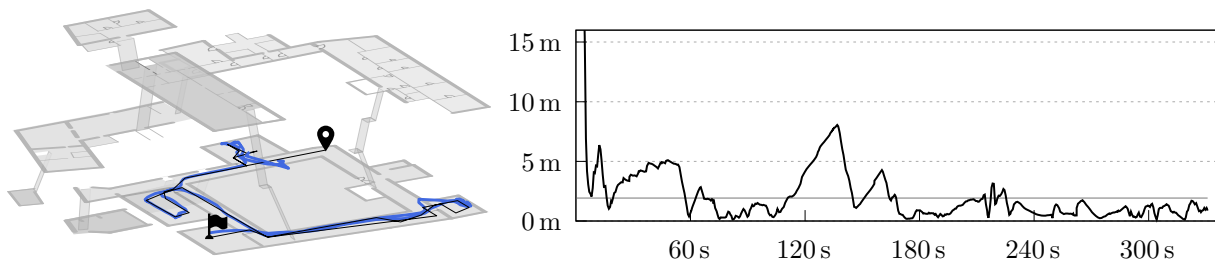
Figure A.13: Estimation result for walk C2, when using the final system.
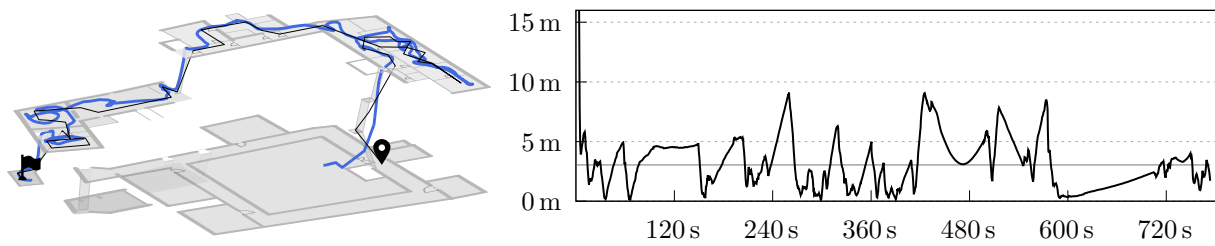
Figure A.14: Estimation result for walk C3, when using the final system.
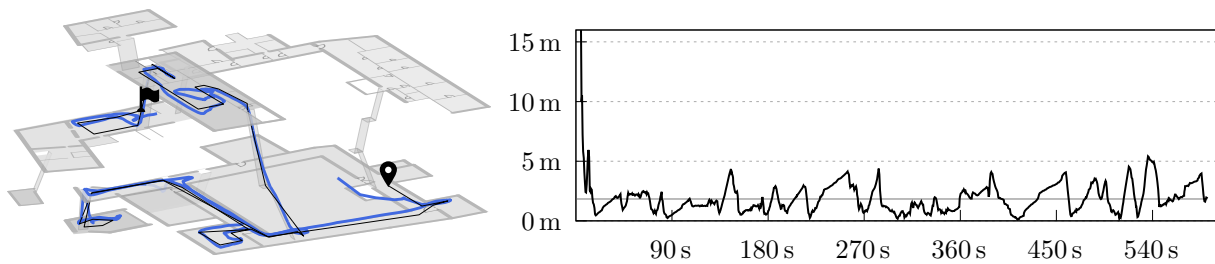
Figure A.15: Estimation result for walk C4, when using the final system.

Index

Lebenslauf

03/1986	Geboren in Werneck
09/1992 – 07/1996	Volksschule Am Wehrbusch, Grafenrheinfeld
09/1996 – 12/2004	Alexander-von-Humboldt-Gymnasium, Schweinfurt Abschluss: Mittlerer Schulabschluss
09/2005 – 07/2007	Friedrich-Fischer-Schule, Schweinfurt Abschluss: Fachabitur
09/2007 – 05/2008	Zivildienst im Jugendhilfezentrum Maria Schutz, Grafenrheinfeld
10/2008 – 03/2012	Bachelor Studium *Informatik* an der Fachhochschule Würzburg-Schweinfurt Abschluss: Bachelor of Engineering
03/2012 – 02/2014	Master Studium *Informationssysteme* an der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt Abschluss: Master of Science
02/2014 – 03/2020	Wissenschaftlicher Mitarbeiter an der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt Aufgabenschwerpunkt: Softwareentwicklung, Forschung und Lehre

Technical innovation results in an enormous amount of data and information that can be used for the description and analysis of individual humans and whole populations. In this context, methods combining different sensor modalities using suitable models are investigated to advance the understanding of the human physical and mental state. This book series is dedicated to this broad field of research.

During the last century, navigation systems have become ubiquitous and guide drivers, cyclists, and pedestrians towards their desired destinations. While operating worldwide, they rely on line-of-sight conditions towards satellites and are thus limited to outdoor areas. However, finding a gate within an airport, a ward within a hospital, or a university's auditorium also represent navigation problems. To provide navigation within such indoor environments, new approaches are required.

This thesis examines pedestrian 3D indoor localization and navigation using commodity smartphones: A desirable target platform, always at hand and equipped with a multitude of sensors. The IMU (accelerometer, gyroscope, magnetometer) and barometer allow for pedestrian dead reckoning, that is, estimating relative location changes. Absolute whereabouts can be determined via Wi-Fi, an infrastructure present within most public buildings, or by using Bluetooth Low Energy Beacons as inexpensive supplement. The building's 3D floorplan not only enables navigation, but also increases accuracy by preventing impossible movements, and serves as a visual reference for the pedestrian. All aforementioned information is fused by recursive density estimation based on a particle filter.

The conducted experiments cover both, theoretical backgrounds and real-world use-cases. All discussed approaches utilize the infrastructure found within most public buildings, are easy to set up, and maintain. Overall, this thesis results in an indoor localization and navigation system that can be easily deployed, without requiring any special hardware components.

Logos Verlag Berlin