Indoor Navigation Based on a Multimodal Positioning System

Master's Thesis

submitted in fulfillment of the requirements for the degree of
Master of Science (M. Sc.)

to the
Department of Economic Sciences II
Degree Program Applied Computer Science
University of Applied Sciences Berlin (HTW)

First Advisor:	Prof. Dr. Jürgen Sieck
Second Advisor:	Prof. Dr. Volodymyr Brovkov
Submitted by:	Björn Bittins
Matriculation Number:	519448
Date:	August 17, 2012

Contents

Acronyms

AIDL	Android Interface Definition Language. 66, 67, V
AoA	Angle of Arrival. 9
AR	augmented reality. 89
BIM	Building Information Models. 33
CAD	computer-aided design. 33
CoO	Cell of Origin. 6, 15, 16
EAN	European article number. 19
ERM	Entity-relationship model. 52
GIS	geospatial information systems. 33, 51
GPS	Global Positioning System. 11–15, 21, 39, 40, 43, 45, 48, 51, 66, 68, 77
HTW Berlin	*Hochschule für Wirtschaft und Technik Berlin*. 2, 41, 77, 87
IDE	integrated development environment. 65
KF	Kálmán filter. 24, 27
LBS	location-based service. 2, 21, 51
MA	map aiding. 37, 51
MEMS	Micro-Electro-Mechanical System. 17, 20, 43, 53
MM	map matching. 37, 51

MVC	model-view-controller. 67
NFC	near field communication. 17, 18, 45, 48, 66, 77
OS	operating system. 65
PDA	personal digital assistant. 2
PDF	probabilty density function. 22–24, 28, 31
PF	Particle filter. 24, 27
POI	point of interest. 38, 60, 70
QR	quick response code. 19, 45, 77
RFID	radio frequency indentification. 17, 18, 20, 43, 56
RMSE	Root Mean Square Error. 79, 81–85, 88
RSSI	received signal strength indicator. 8, 10, 16, 39, 40, 48
SDK	software development kit. 65, 66
TDoA	Time Difference of Arrival. 7
TGS	*Technologie und Gründerzentrum Spreeknie Berlin*. 56, 70, 77, 78, 85, 86, 88, XIII
ToA	Time of Arrival. 7, 8
TSP	travelling salesman problem. 34
UWB	ultra-wide band. 15
VM	virtual machine. 65
WGS 84	World Geodetic System 84. 68

1. Introduction

1.1. Motivation

Within living memory the domain of navigation is of great interest and was steadily researched and further developed. Nowadays, navigation and the possibilities provided by it have only very little in common with the orientation at landmarks and simple maps back then. Prerequisite for the development of navigation systems are sophisticated positioning methods which are able to provide the current location of a user or device with an adequate accuracy for a given context. Various technologies are available for different fields of application. The accuracy of these technologies range from several meters, up to a few centimeters, depending on the specific context.

With the *Global Positioning System* (GPS), the *Galileo* system and other satellite navigation systems, several globally operating positioning technologies are available nowadays. These systems already proved its suitability for daily use in various products, such as car navigation systems and smartphones, or will do so soon. In most environments the globally operating positioning systems work well. However, in specific areas, such as urban neighborhoods (so called *urban canyons*) and indoor environments, these systems operate unreliably or, in the worst case, not at all.

Various technologies are available to determine the location of a user or device in a local manner. These technologies are often based on optical, acoustic, or radio methods. Depending on the area of application and the specific environment these systems have various advantages and disadvantages.

As Lorenz and Ohlbach [LO06, p. 102] state, "car navigation systems are becoming a more or less standard commodity nowadays [... and ...] the problem of navigating cars through large road networks has been well investigated and the solutions are mature". Less investigated is the domain of navigating pedestrians through indoor environments. An actively assisted indoor navigation system would be beneficial, especially in large buildings, such as airports, hospitals, supermarkets, and office buildings. Thus, a user, equipped with a mobile device such as a smartphone, could be directed from the entrance hall of an office building to a clerk with whom one has an appointment. Furthermore, the path could lead the user past various facilities according to the users needs and interests, e.g. the information desk, rest rooms, or exhibition areas (so called *points of interest*, POI). Another scenario could be *efficient shopping* in large shopping malls. Users could save a lot of time if an indoor navigation system would not only show the way to the shops that provide the required goods but also calculate the shortest path along all products on their shopping list in a particular shop. In addition, prices could be compared between shops in the vicinity that are providing the same or similar products and thereby influencing the path of travel and shopping time. However, according to Lorenz and Ohlbach [LO06, p. 102] the navigation in indoor environments is likely to be more challenging in comparison to outdoor navigation. The reasons primarily is the more complex topology of buildings compared to a network of streets.

Over the past few years *smartphones*, a combination of a mobile phone and a *personal digital assistant (PDA)*, have spread rapidly. Today, smartphones enable the users a location independent access to a variety of services, e.g. the Internet and the playback of multimedia content. Furthermore, smartphones integrate various sensors which are able to determine the context of the user, including the location. Location awareness and the associated *location-based services (LBSs)* are, according to Gartner Inc., key technologies in the mobile sector for the next couple of years [Gar].

Due to the wide distribution of smartphones and the numerous possibilities that those provide, this device class represent a suitable platform to implement an indoor navigation system. Multiple sensors and external services can be utilized to determine a location as accurately as possible. The location information can be used within the navigation system.

1.2. Research Objectives

The aim of this work is the development of strategies to navigate a person reliably through an indoor environment. A simple office environment with bureaus and other *points of interests (POI)* will serve as example scenario.

The research of suitable positioning technologies is essential to actually offer navigation services. This work is based on a *multimodal positioning service* whose general architecture was developed in the context of a research project during the masters program at the *Hochschule für Wirtschaft und Technik Berlin* (HTW Berlin). The project is described in Bittins [Bit11], [Bit12b], and [Bit12a]. It will be briefly introduced and discussed for general understanding. The positioning service is able to integrate various sensors and external services to provide information about the current location. A combination of several positioning methods is possible. To provide a suitable foundation of positioning capabilities for indoor environments, different sensors, services, and combinations of those will have to be evaluated. The focus will be primarily set on optical, acoustic, and radio systems as well as sensor data fusion methods.

The indoor navigation application relies on the information provided by the *multimodal positioning service*, but both components are independent of each other. The results of the research project indicates that the position information provided by the *multimodal positioning service* might not be as accurate as needed to provide reliable navigation services in indoor environments. Basically there are two options to improve the accuracy and reliability of the position information: integrate additional positioning components directly into the *multimodal positioning service* and using the information of third-party applications, such as the indoor navigation application, to correct the inaccurate position information. Feasible options and mechanisms of how the indoor navigation application with its additional information from navigational databases can contribute to improve the accuracy of the positioning service will also be researched.

As mentioned before, outdoor navigation, such as car navigation, is very different from the navigation in indoor environments. Suitable representations of the topology of buildings and additional information will have to be evaluated. The differences between outdoor and indoor navigation can also result in adjustments to common path finding algorithms that are used in navigation systems. The specific needs and interests of a user in these kinds of environments should be taken into account by those algorithms.

1.3. Thesis Outline

The underlying technologies and systems and the state of the art technology in the relevant areas will be introduced and discussed in the section **Fundamentals and State of the Art**. Positioning and navigation systems as well as the theory of sensor data fusion will be covered in this section. Related work will be reviewed considering the respective advantages and disadvantages.

The requirements and use cases for the underlying positioning service and the navigation component will be defined in the section **Specification Analysis**. Furthermore, it is also equally important to specify the requirements of the underlying positioning service and the user interface to be able to describe the features of the prototype in detail.

In the following section, named **Conceptual Design**, a strategy that transforms the requirements into a system design will be composed. The components and algorithms that will be employed are described in detail. The conception will be based on the example scenario which was described before and covers the underlying positioning service as well as the navigation component.

The strategy, which was developed before, will be reviewed in terms of the example scenario in the chapter **Prototypic Implementation**. The implementation of the system design will be discussed by means of the scenario and potential problems will be revealed.

The implementation will be **evaluated** afterwards with regards to the realization of the requirements which were defined during the analysis. The implemented system will be critically evaluated in an actual office environment on the basis of predefined criteria with the help of a test scenario.

Finally, the results of the work will be **summarized** and a short *outlook* about future improvements and developments will be given.

2. Fundamentals and State of the Art

In the following the technologies that are necessary for positioning and navigation will be introduced. Positioning methods and technologies play a key role for navigation systems. Several positioning technologies can be combined using sensor data fusion methods. The chapter concludes with the fundamentals of navigation.

2.1. Positioning

Navigation is not possible without having position information. Actual smartphones integrate a variety of sensors that can be used to determine the position of the device. Specific positioning methods of how a position can be derived independent from the underlying technology will be described. Later, key technologies in the area of RF-based, optical, and further positioning technologies will be introduced.

2.1.1. Positioning Methods

Positioning methods are independent from the used positioning technology. Different positioning methods can be potentially employed by a specific positioning technology. Several positioning methods will be introduced and reviewed according to their advantages and disadvantages. Particular attention is paid to the capability of the implementation of these methods with sensors that commonly integrated into actual smartphones.

Two different ways exists for implementing a positioning system: self-positioning and remote-positioning. "In self-positioning, the physical location is self-determined by the user's device using transmitted signals from terrestrial or satellite beacons. The location is known by the user and can be used by applications and services operating on the user's mobile device. In remote positioning, the location is determined at the server side using signals emitted from the user device. The location is then either used by the server in a tracking software system, or transmitted back to the device through a data transfer method" [LSW09]. Which way is used to implement a positioning system is of particular importance when choosing the methods and technologies that should be used within the system. Sensory hardware that is integrated in actual smartphones might be too inaccurate and using better and more accurate hardware might be too expensive or not feasible due to other limitations of mobile devices, such as processing power, energy consumption, and size. On the other hand, the same methods can be beneficial when implementing a remote-positioning system as the expensive hardware can be integrated in sparsely distributed base stations which do not have the limitations of mobile devices.

Four different categories of positioning methods are typically used for positioning systems. These are based on the cell of origin, distances (lateration), angles (angulation), or pattern recognition [Cis08, para. 2]. Methods of all categories will be introduced.

Cell of Origin

The *Cell of Origin (CoO)* method uses location information of the base station the mobile device is currently connected to. It can be used in cellular networks such as the GSM network (Global System for Mobile Communication). The simplest variant of this method is, that the location of the base station is considered as location of the mobile device. This method is not specified by the 3GPP (3rd Generation Partnership Project) which specified the GSM and other standards of the mobile communication sector [Küp05, p. 192]. This variant can be used for self-positioning as well as remote-positioning. The method is easy to implement and no further enhancements are required on the side of the base station and the mobile station, but the accuracy is very low. Several enhancements can be applied to this method in GSM networks to improve the accuracy. GSM base stations usually use three or more antennas to cover the area around the base station. These so called sector antennas can be used to improve the accuracy of the location of the mobile station. A further enhancement of the method uses a *Timing Advance* value that "allows to identify a ring of potential positions of the target terminal with the serving base station in its center" [Küp05, p. 192] (see Figure 2.1). A characteristic of GSM networks is, that the mobile stations periodically connect to the base stations nearby to determine the base station with the strongest signal. The location information of multiple base station is used to narrow down the location of the mobile station [Küp05, pp. 194-197].

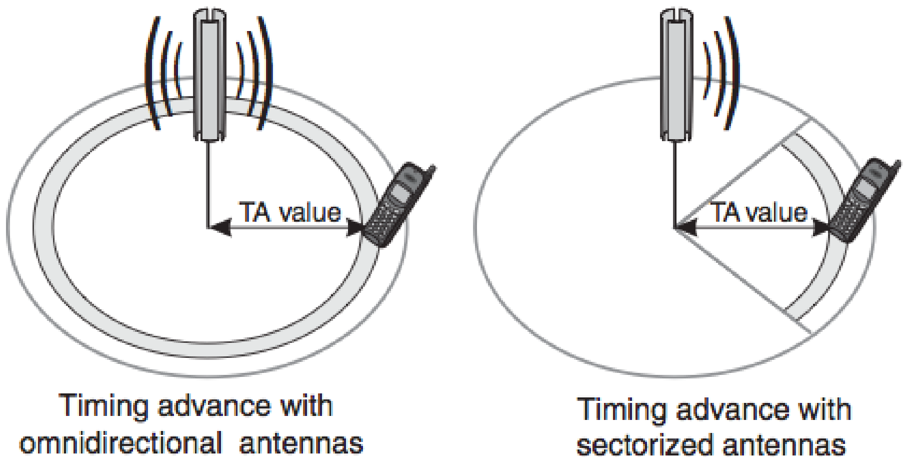

TA value
TA value
Timing advance with omnidirectional antennas
Timing advance with sectorized antennas

Figure 2.1.: Timing advance with omnidirectional and sectorized antennas. [Küp05, p. 192]

Time of Arrival / Time Difference of Arrival

Transit time methods are amongst the most important techniques for positioning. Generally two different methods exist: the measurement of the transit time between a base station and the mobile station and the time difference between the arrival of a signal at different base stations [Str+08, p. 41].

"*Time of Arrival (ToA)* systems are based on the precise measurement of the arrival time of a signal transmitted from a mobile device to several receiving sensors" [Cis08, para. 2]. Because the velocity of the signal is known (speed of light with optical and electromagnetic signals, sonic speed with sound waves), the distance between two stations can be determined from the time the signal travels from one station to the other. Figure 2.2 a) illustrates the concept of ToA with three base stations and one mobile station (tri-lateration). The disadvantage of ToA approaches is the need of very precise and synchronized clocks of all stations. This can be challenging especially for the mobile stations when the signal velocity is almost the speed of light [Cis08, para. 2].

The *Time Difference of Arrival (TDoA)* approach depends on relative time measurements in comparison to the absolute time measurements used in ToA systems and therefore does not require synchronized clocks for mobile stations. In fact time synchronization is only required amongst the receiving base stations. To calculate the position of a mobile device TDoA uses the concept of *hyperbolic lateration*. A message has to be sent from the mobile station and received by at least three base stations to be able to determine its position [Cis08, par. 2]. The case of a mobile station M, that sends a

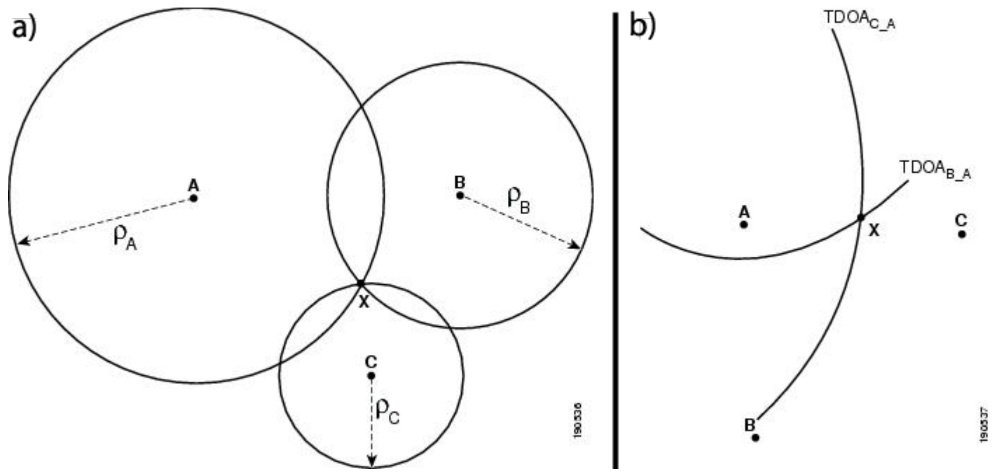


Figure 2.2.: a) Time of Arrival. b) Time Difference of Arrival. [Cis08, para. 2]

message which is received by two base stations B1 and B2, will illustrate the concept: if the message is received by B1 earlier than by B2, the mobile device is closer to B1 than to B2. The time difference between the arrival of the message at B1 and the arrival at B2 can be mapped to a distance difference according to the velocity of the signal. This relation can be represented by a hyperbola of possible locations of the mobile device. As mentioned before, a third base station is required to form a second hyperbola. The intersection of both hyperbola is the position of the mobile device [Bil12, p. 5] (see Figure 2.2 b).

Received Signal Strength Indicator

The basic idea behind the *received signal strength indicator (RSSI)* approach is the measurement of the received power and thus the attenuation of the signal, which depends on the distance between the transmitter and the receiver [Str+08, p. 42]. Knowledge about the physical relation between distance and attenuation (the propagation model) is essential to calculate the distance between transmitter and receiver. With RSSI values of at least three base stations, the position of a mobile device can be determined via lateration. The illustration of the concept is similar to the ToA approach (cp. Figure 2.2 a). Positioning systems based on RSSI in WiFi networks are widespread because the required infrastructure often exists and therefore is cheap and easy to install. The accuracy of the RSSI method depends on the accuracy of the propagation model. Especially in wireless technologies that operate in the 2.4 GHz band, such as Bluetooth and Wi-Fi, the propagation of the signal is heavily influenced by interference factors, such as reflections due to multipath or obstacles, such as walls, steel constructions, and water. Generally the signal strength information can be obtained from two sources: the infrastructure (base stations) and the mobile device. When using a client-based positioning approach, the results of the RSSI measurements can vary due to different hardware of the mobile devices [Cis08, para. 2].

Angle of Arrival

"The Angle of Arrival (AoA) technique [...] locates the mobile station by determining the angle of incidence at which signals arrive at the receiving sensor" [Cis08, para. 2]. Figure 2.3 illustrates the case of locating a mobile device with two base station (receiving sensors) in a two-dimensional plane: a radial line can be drawn from each base station in the direction of the highest signal strength. The mobile device is located at the section of all radial lines. The accuracy of this approach depends on the estimation of the incidence angle. Mechanically-agile directional antennas can be aimed at the bearing with the highest signal strength. "In practical commercial and military implementations of AoA, multiple element antenna arrays are used to sample the receiving signal, thereby eliminating the need for more complex and maintenance-intensive mechanical antenna systems " [Cis08, para. 2]. Downside of this approach is the requirement of expensive and complex antenna systems. Furthermore, a clear *line of sight* between the mobile station and the base stations is assumed. Otherwise multipath propagation will occur due to reflections and affect the determined angles at the base stations, thus lowering the accuracy [Str+08, p. 45].

Fingerprinting

Fingerprinting techniques are of particular interest in environments where base stations are sparsely distributed and thus, methods based on angles or transit time would not produce reliable results [Str+08, p. 45]. Sources of interferences, such as multipath propagation, are utilized to create a specific pattern of the observed parameters for each location. These patterns are stored in a fingerprint data base. To locate a device, the parameters are recorded by the mobile device and compared to the fingerprints in the data base. The fingerprint with the highest score according to a predefined similarity function is the location of the device. Two phases are required to implement a fingerprint system [Cis08, para. 2]:

Calibration phase During the *calibration* or *off-line* phase, suitable location-dependent parameters are identified for the given area, for example RSSI values of nearby base stations. The area is divided into a more or less fine-meshed net, according to the desired accuracy [Str+08, p. 45] (see Figure 2.4). The positioning is sector-based

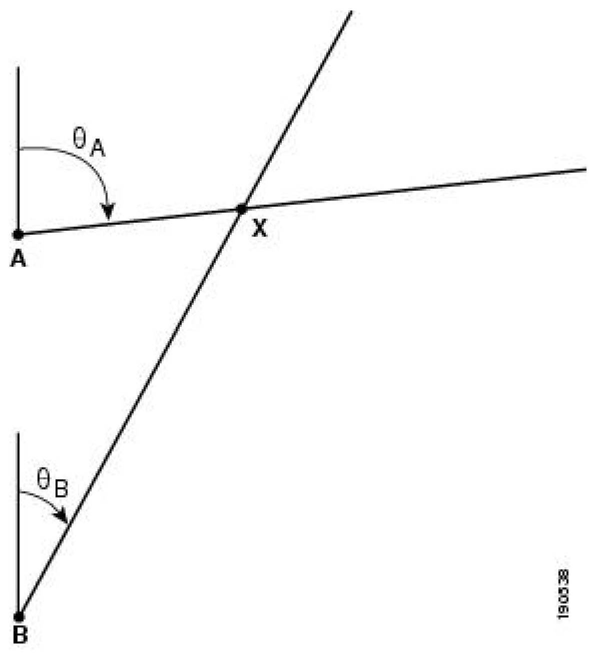


Figure 2.3.: Angle of Arrival. [Cis08, para. 2]

and the accuracy depends on the edge length of each sector. If the net consists of more, and thus smaller sectors, the accuracy might be better, but more fingerprints have to be recorded and it is more likely that multiple fingerprints show a high degree of similarity, resulting in false positioning or ambiguity.

Operational phase During the *operational* or *on-line* phase, the specified parameters are recorded by the mobile station. The resulting fingerprint is compared to the fingerprints that were recorded during the *calibration* phase. The coordinates of the sector, in which the fingerprint(s) with the highest similarity score was originally recorded, represent the position of the mobile station.

The identification of parameters that are used for the fingerprint is of particular importance: the more characteristic (location-depended) parameters are available the better the fingerprints will be distinguishable [Str+08, p. 45]. Location patterning positioning algorithms are used to compare the fingerprints recorded during the operational phase with the recorded fingerprints of the calibration phase. These algorithms can be classified into three groups [Cis08, para. 2]:

Deterministic algorithms "attempt to find minimum statistical signal distance between a detected [...] location vector and the location vectors of the various calibration sample points" [Cis08, para. 2].

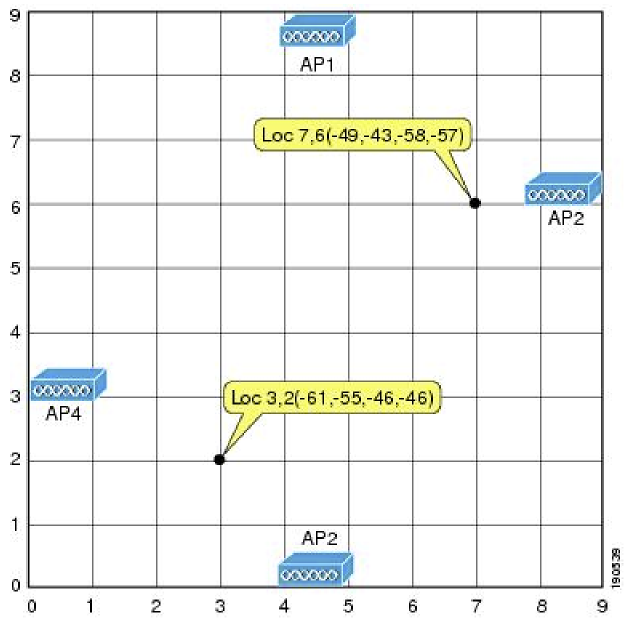


Figure 2.4.: Pattern Recognition / Fingerprinting. [Cis08, para. 2]

Probabilistic algorithms "use probability inferences to determine the likelihood of a particular location given that a particular location vector array has already been detected. The calibration data base itself is considered as an a priori conditional probability distribution by the algorithm to determine the likelihood of a particular location occurrence" [Cis08, para. 2]. Different deterministic and probabilistic algorithms algorithms are described in [Bil12, pp. 8-10].

Other techniques go beyond deterministic or probabilistic approaches. Examples for this group are neural networks and support vector modeling (SVM) [Cis08, para. 2].

According to [LSW09, p. 14] fingerprint methods are very accurate. On the other hand these techniques have some drawbacks. The characteristic parameters are subject to change over time. Minor changes in the arrangement of the building or simply a varying number of people in the sphere of influence can have a huge impact on the detected fingerprint and thus producing poor positioning results. The fingerprint data base must be kept up-to-date which would require, depending on the utilized RF-technology, a regular recalibration of the fingerprints. Especially in large areas this can lead to an uneconomical workload.

2.1.2. Radio-based Positioning Systems

Radio-based positioning systems are well researched and several systems exists for different radio technologies, such as Global Positioning System (GPS) , Wi-Fi, Bluetooth, RFID, and others. These systems have advantages and disadvantages in specific environments, which will be discussed in the following sections. The systems can be classified into four different groups:

- Global systems
- Wide-area systems
- Local-area systems
- Near-field systems

Global Systems

Typical representative of globally operating systems is the group of the so called *Global Navigation Satellite Systems (GNSS).* With *GPS* and *GLONAS*, the Russian counterpart, two mature technologies of this category are available and widely used within positioning and navigation systems, such as maritime and aviation positioning systems, car navigation, and many more. With *Galileo*, an European GNSS system is currently deployed and will be available around mid-decade[1], granting better accuracy and further services to the customers. All these systems have in common that they are based on precise transit time measurements from satellites, that orbit the earth, to receivers on the earth's surface. The systems were "conceived as a ranging system from known positions of satellites in space to unknown positions on land, at sea, and in space" [HLW03, p. 172]. Because GPS is the most widely used system and conceptually similar to the other systems, it will be described in detail in the following. The GPS system consists of three different segments [Küp05, pp. 162-163]:

- Space segment
- Control segment
- User segment

[1]http://www.esa.int/esaNA/galileo.html

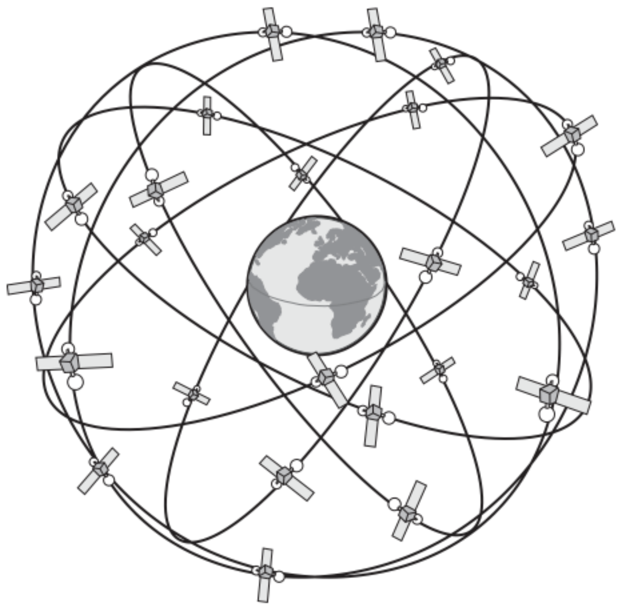


Figure 2.5.: GPS satellite constellation. [Küp05, p. 164]

The *space segment* consists of a number of satellites in nearly circular orbits in an altitude of around 20 200 km. The operational constellation is composed of 24 satellites which are deployed "in six evenly spaced planes with an inclination of 55° against the equator and with four satellites per plane" [HLW03, p. 173] (see Figure 2.5). With this constellation, the earth's surface is completely covered with four to eight simultaneously observable satellites (elevation angle >15°) [HLW03, p. 173]. "The *control segment* is responsible for monitoring and controlling the satellites of the space segment"[Küp05, p. 163]. The health, orbits, and accuracy of the internal clocks of each satellite are observed and corrections are calculated and transmitted to the satellites. The user segment is often referred to as *GPS receiver*. Nevertheless, there are many different types of receivers available, depending on the application the receiver will be used for. For navigation application a receiver with moderate accuracy but fast obtaining of the first position (*Time to First Fix, TTFF*) will be sufficient, whereas a high accuracy is required for mapping applications and military purposes [Küp05, p. 163].

A direct line-of-sight of at least four satellites is required to calculate the unknown position of a mobile station in the space between the satellites and the earth's surface, utilizing the transit time of the signals from the satellites to the mobile device and the location of the satellites. This information (timestamp and the satellites coordinates) is encoded in the messages that are periodically sent by the satellites. With tri-lateration, only three satellites should be sufficient to calculate the position of the mobile station but that would imply that the clocks of the satellites and the mobile station are synchronized. Because atomic clocks are not suitable for most mobile stations, the signal of a fourth satellite is used to compensate the clock error of the inaccurate clocks of the mobile station [HLW03, p. 179]. GPS basically provides two types of services: *standard positioning service (SPS)* and *precise positioning service (PPS)*. SPS focuses on civilian applications whereas PPS provides better accuracy and is meant for military purposes. Receivers conforming to the SPS category only have access to one of the two carriers (L1 and L2 at 1575.42MHz respectively 1227.60MHz) on which the GPS signals are sent. Due to the modernization of the satellites an accuracy of around 8m is provided for SPS [Dep08].

Differential GPS (DGPS) is an enhancement that uses at least two receivers. The location of one of the receivers, the so called reference or base receiver, is known, the location of the other, mostly mobile, receiver is to be determined. The base receiver processes the messages of all observable satellites and computes the ranges to these. With the precisely known coordinates of the base receiver and the coordinates of the satellites derived from the GPS messages, the actual ranging errors of the satellites can be determined (DGPS corrections). These corrections are sent to the mobile station which subsequently uses the information to correct the ranges computed from its own GPS measurements. Using this method the accuracy can be improved, achieving a sub-meter level [ElR02, pp. 78-80].

The GPS technology is an excellent choice for positioning systems that are designed for outdoor environments. The disadvantage of this technologies is the poor availability in urban areas and indoor environments.

Wide-area Systems

Mobile communication systems like GSM and UMTS (Universal Mobile Telecommunication System) cover large parts of the earth's landmass. But in contrast to GNSS they do not form a contiguous network and some rural parts as well as most of the oceans and other water expanses are not covered at all. Nevertheless, these networks provide services that can be used to locate a mobile device. Most mobile communication networks are organized in a cellular manner. A base station serves the mobile stations in the surrounding area. The mobile devices connect to the base station with the best reception. The radius of a cell in GSM networks can vary between several 100m in urban environments up to 35 km in rural areas [Küp05, p. 91]. The advantage of this design is, that the base stations can be deployed more dense in areas where many users should be served simultaneously. In areas where less users are expected, the base stations are distributed more sparsely, covering a greater area (see Figure 2.6).

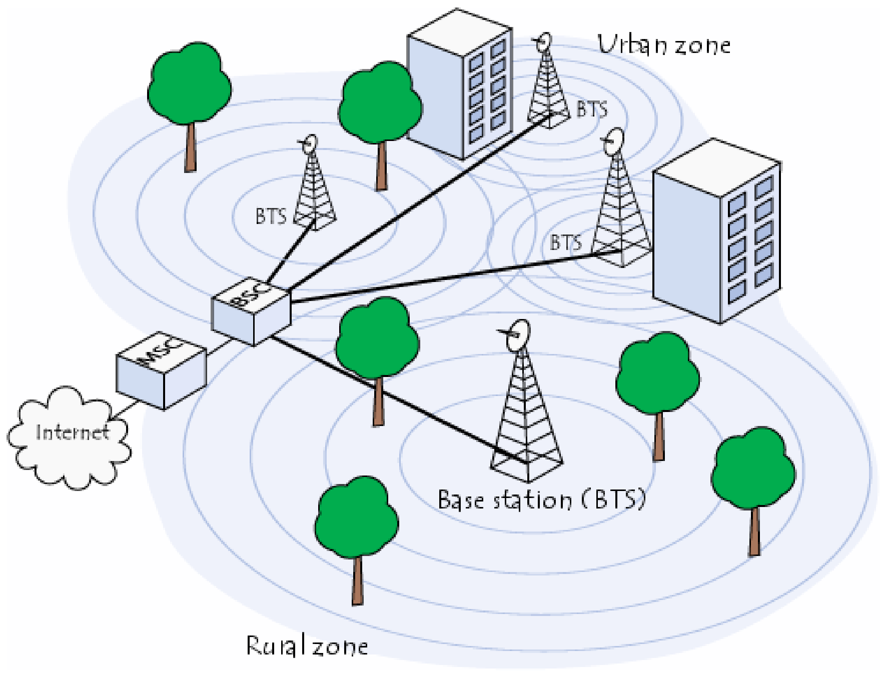


Figure 2.6.: How GSM networks work. [Isl12]

Services to locate mobile devices in mobile communication networks are available. The *CoO* method is one of the basic localization methods and described in section 2.1.1. Advanced positioning methods, such as multi-lateration, where the position information of multiple base stations is used, are also available in most of these networks [Küp05, p. 92]. In rural areas, the reception of signals from multiple base stations might not be possible, due to the sparse distribution of base stations.

Mobile communication networks can be used to assist other positioning methods. In *Assisted GPS (AGPS)*, the rough position estimate of the cellular network is used to narrow down the search area for the signals of the satellites and to obtain information regarding the satellites, such as the orbital parameters and correction data. This data is used to reduce the period until the first position is obtained (TTFF) from between 12 minutes and 30 second with GPS to only a few seconds with AGPS . Furthermore, the communication link required for AGPS can be realized with mobile communication networks [Str+08, p. 30].

High availability, even in many indoor and outdoor environments, is the advantage of positioning in mobile communication networks. In rural areas and within buildings, the reception might fail. Also the accuracy of these systems is very poor. Mobile communication networks can be used to assist other technologies by providing a rough position very fast as well as providing additional information.

Local-area systems

In local-area positioning systems the wireless technologies for *local area networks (LAN)* are used. Typical representatives in this section are Wi-Fi, ultra-wide band (UWB), and ZigBee. Most effort was conducted in the field of Wi-Fi and Bluetooth positioning systems because these technologies are widespread and integrated in many mobile devices.

The Wi-Fi technology is based on standards described in IEEE 802.11 and is meant for *Wireless Local Area Networks (WLAN)* operating in the 2.4 GHz and 5 GHz band. The 802.11 standard family defines several protocols that implement enhancements, such as security features and higher data rates. The 802.11g standard is very common and supported by most mobile devices. The data rate is up to $54\,\mathrm{Mbit\,s^{-1}}$ with approximate ranges of 38m in indoor environments and up to 140 m under optimal conditions. The successor, the 802.11n protocol is widely supported by actual smartphones. Systems conforming to this protocol achieve data rates of up to $150\,\mathrm{Mbit\,s^{-1}}$. Wi-Fi networks are composed of cells, forming a cellular network. Each cell is served by a base station, the so-called *access point (AP)*. If the client nodes connect to the network via an AP, the operational mode is called *infrastructure mode*. Wi-Fi also enables the clients to communicate directly without any further infrastructure. This operational mode is called *ad hoc mode* [Küp05, pp. 233-234].

Positioning services of Wi-Fi networks are commonly based on RSSI, fingerprinting, or CoO approaches (see subsection 2.1.1). Examples for research in this area are presented in Biswas and Veloso [BV10] and Prasithsangaree, Krishnamurthy, and Chrysanthis [PKC01]. A RSSI-based fingerprinting approach was used in both works. An accuracy of 1.8m with a precision of 80% was achieved in Biswas and Veloso [BV10].

Bluetooth is a wireless communication technology originally developed for *Wireless Personal Area Networks (WPAN)*, aimed to connect personal devices, such as mobile phones, PCs, laptops and peripherals. However, different classes of Bluetooth devices exists, that differ in maximum range and power consumption. Devices of class 1 or 2 are usually integrated in actual smartphones. Class 2 devices consume up to 2.5 mW with a range of up to 10 m whereas class 1 devices can consume up to 10 mW providing a range of up to 100 m. Like Wi-Fi, Bluetooth operates in the 2.4 GHz band. Bluetooth was developed to exchange data over short distances and in a direct manner. Networks are created *ad hoc*, so no dedicated hardware, such as access points in Wi-Fi networks, are required. Up to eight devices can form a so-called *piconet* where one of the devices is promoted as *master* and thus managing the communication in this piconet. Several pico-nets can form a *scatter-net* by establishing connections between the master devices of each piconet.

Positioning systems based on the Bluetooth technology are researched for instance in Kotanen et al. [Kot+03], Rodas, Escudero, and Iglesia [REI08], and Pei et al. [Pei+10]. All works are using the RSSI approach to infer the distance to distributed Bluetooth beacons. Pei et al. [Pei+10] uses the RSSI values to create fingerprints. In Rodas, Escudero, and Iglesia [REI08] an accuracy of around 1m with a precision of 50% respectively 2 m with 95% precision is achieved. However, the other works are not able to achieve such high accuracies and account the unreliability of the RSSI values ([Kot+03]) or the only sparsely distributed Bluetooth beacons ([Pei+10]) for the inadequate results. One major problem of RSSI based approaches with Bluetooth is, that the RSSI value can only be obtained during the inquiry process, when the environment is scanned for other Bluetooth devices. This cannot be done in a very high frequency because not all devices will be found if the inquiry period is too short.

Near-field Systems

Near-field systems utilize technologies with ranges of only up to a few meters. Depending on the density of the distribution of beacons, these methods rely on proximity sensing and lateration approaches. Typical technologies in this area are *radio frequency indentification (RFID)* and *near field communication (NFC)*. RFID is a contactless, wireless system, primarily intended to identify objects automatically. It uses magnetic and electromagnetic fields to transfer data between a transponder and a reader [Fin06, pp. 6-7]. Several specifications define the parameters, such as the frequency band, the transmission methods, and the range, of different RFID systems, such as ISO/IEC 14443 and ISO/IEC 15693. In RFID systems, the transponders can be passive or active. Passive transponders do not have an internal power supply and use the energy obtained from the field of the reader as energy source. Depending on the technology, antenna size, and the tag type, RFID systems provide transmission ranges from a few centimeters (passive tags) up to several meters (active tags).

RFID systems are occasionally used in positioning systems. Bouet and Santos [BS08] gives an overview on how RFID can be used for positioning. Most common approaches are RSSI-based or proximity-based. Due to better transmission ranges, active RFID tags are more often used than the passive counterparts. Renaudin et al. are proposing a RFID and Micro-Electro-Mechanical System (MEMS)[2]-based approach for emergency

[2]MEMS “that result from the integration of mechanical elements on a common substrate” [Ren+07, para. 2]. Examples: accelerometer, gyroscope, magnetometer, barometer.

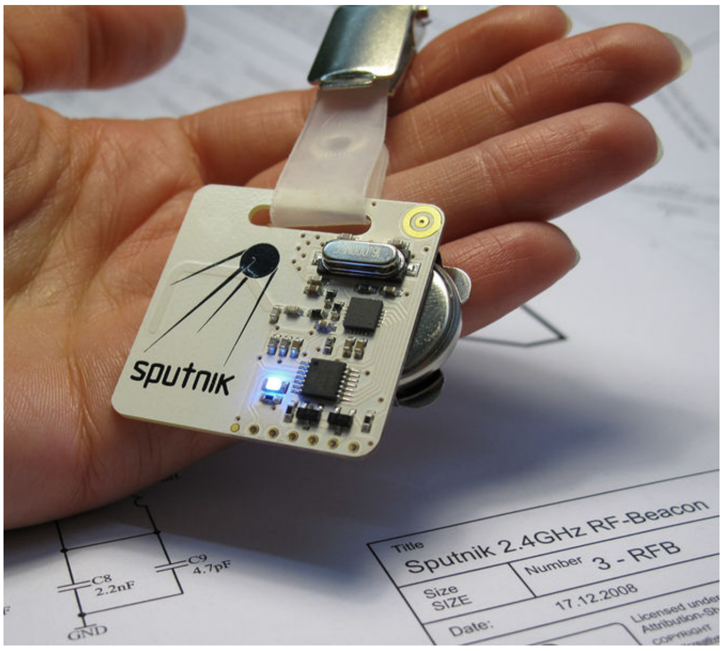


Figure 2.7.: Active RFID OpenBeacon Proximity Tag (Sputnik). [Mer10]

scenarios. RFID tags are distributed ad hoc in the area for the purpose of compensating the accumulated errors of the MEMS system. In this way, an “almost self-deployable” [Ren+07] was developed. Bergemann presented a RFID-based, proximity-sensing system for social events in [BSH10]. This system employes active RFID tags of the *OpenBeacon*[3] platform. A RFID tag is capable of scanning the immediate neighborhood for other tags by alternating transmission and receiving cycles. Data packets with the tag ID and the power level indicator are sent at different power levels. The tag subsequently switches into the receive mode on the same radio channel and listens for packets from other tags. In this way, the position of a tag can be derived in relation to the position of other tags that are nearby.

The NFC standard is based on the RFID technology. It provides a limited communication range of up to 20cm and transmits data in the 13.56MHz band [FM10, p. 57]. The maximum data rate is 424kBit/s [Ozd+11] NFC distinguishes three operating modes [FM10, pp. 57-58]: active mode, passive mode and card emulation mode.

Only little research has been conducted to evaluate the capabilities of NFC for positioning systems. In [Ozd+11] a navigation system based on NFC reference tags, which encode location information, was developed. The user is only located when a reference

[3]The OpenBeacon platform was introduced by the Berlin-based company *Bitmanufaktur* (`http://www.openbeacon.org/`).

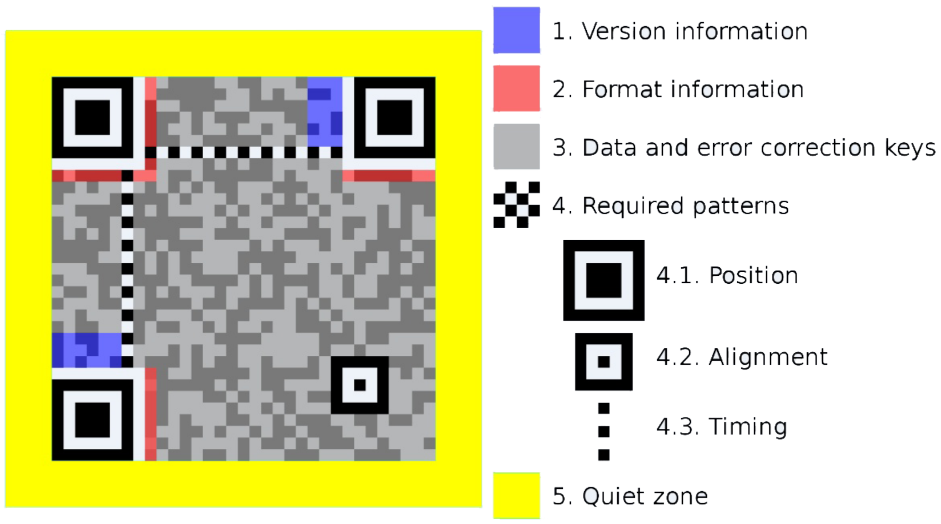


Figure 2.8.: Structure of a QR code. [Egl]

tag is read by the mobile device. Neither positioning nor guidance can be provided in between the reading of two reference tags. Nevertheless, NFC tags have the advantage of low costs, so they can be distributed densely in the environment. Due to the limited transmission range, the accuracy of location information, that is encoded with these tags, is very high. Furthermore, the reading procedure can be triggered automatically, not requiring the user to take any action except placing the mobile device near the NFC tag.

2.1.3. Optical Positioning Systems

In optical positioning systems visual markers are commonly used to obtain relative (to the marker) or absolute position information. The markers have to be designed in a way, that the pattern of each marker can be clearly identified. To achieve this requirements the markers can conform to an established standard, such as *bar codes* or *QR codes*[4], or the design of the marker must be unique for the specific optical system.

The software library *ARToolKit* simplifies the creation of *augmented reality (AR)* applications. According to the Oxford Dictionary augmented reality is “a technology that superimposes a computer-generated image on a user’s view of the real world, thus providing a composite view” [Oxf]. With this library arbitrary images can be defined as markers and be used in an AR application later on, triggering actions. *ARToolKit*, including the source code, is licensed under the *GNU General Public License* for non-commercial usage [Lam].

[4]Quick Response code

Several standards have been specified to encode information in images and markers. Amongst these are bar codes and quick response codes (QRs). Both technologies specify the design of a marker. A bar code is one-dimensional and based on a varying sequence of bars and gaps with differing thickness. Bar codes are commonly used to identify products in markets automatically. Representatives of this category are the European article number (EAN)-11 and EAN-13 specification. QR code is an extension of the bar code. The information is presented in two dimensions, allowing to encode more data on the same space (see Figure 2.8). Several specifications, differing in size, amount of data, and error correction data, exist for QR codes as well. A detailed description of the QR code norm ISO/IEC 18004 is available at [ISO00].

2.1.4. Further Positioning Systems

Acoustic Systems

By recording acoustic signals of the environment, information about the current location can be obtained. Basically two different approaches are possible: on the one hand, an infrastructure that emits acoustic signals actively can be deployed and on the other hand, the natural ambient sound and noise can be analyzed to acquire relevant information. Kushwaha et al. consider the first possibility in [Kus+05]. By sending well-defined acoustic signals from a (mobile) beacon to the nodes of a *wireless sensor network (WSN)*, the nodes are able to determine their own position relatively to the position of the beacon. The second approach was researched in [Str10]. Ambient sounds that occur during a metro ride were analyzed (in combination with further sensor systems) to locate a mobile device.

Dead Reckoning

Dead reckoning (DR) approaches rely on MEMS systems to obtain relative position information. Accelerometers, compasses, gyroscopes, magnetometers, and other sensors, that can be integrated or connected to a mobile device, are used to determine the heading and movements of users or objects. DR "is the process of estimating the current position of a user based upon a previously known position, and advancing that position based upon measured or estimated speeds over elapsed time and course" [SCM10, para. 3]. An initial position must be obtained through another positioning method or by eliminating unreachable positions according to the prior movements. The advantage of DR is, that these systems are fully independent in regard to external components, such as beacons or reference stations. DR systems are commonly used in combination with other positioning methods, because it is more convenient to obtain an initial position from other sources. Errors, which are induced by inaccurate measurements or bad estimates, accumulate over time and will have a significant effect on the positioning results [SCM10, para. 3]. Step detection algorithms are very often implemented with MEMS systems. As mentioned before, an indoor navigation system for emergency scenarios based on RFID and MEMS was introduced in [Ren+07]. In [SCM10], the initial position is obtained by scanning a QR code that encodes its own position.

Summary Positioning

Different positioning technologies have been reviewed in the last section. Each technology has its strengths and weaknesses in specific environments. None of the technologies is suitable to cover the majority of scenarios, such as indoor and outdoor environments. The main driver of smartphones and telecommunications are LBSs and within this area, seamless navigation and particularly indoor navigation is upcoming. To be able to provide these features, several positioning technologies have to be combined in a way that position information is available in a vast majority of vicinities, for example that the technologies are chosen to complement each other, and without the user noticing that different technologies are used. *Sensor data fusion* methods can be used to fuse data from different sources in such a way. Many positioning systems use sensor data fusion approaches to integrate different positioning technologies.

2.2. Sensor Data Fusion

As stated in the previous section, the different positioning systems have specific advantages and disadvantages in certain environments. Therefore, the combination of different positioning systems that complement each other would be beneficial in regard to the quality of the positioning results in varying environments. For example, in most outdoor environments, the GPS sensor works reliable and provides location information with an accuracy of up to 5 meters. This accuracy is sufficient for most location-based outdoor applications, such as pedestrian or car navigation. However, in indoor environments, the result of GPS is unreliable or no position can be obtained at all. Furthermore, the accuracy of GPS might be insufficient for location-based indoor applications and it is more likely to deploy a suitable positioning system locally, where a better accuracy is actually needed, for example in an office building or a shopping mall. To be able to create a seamless LBS for both scenarios, the data of different positioning systems and sensors have to be combined. *Sensor data fusion* methods are used to fuse the noisy data from different systems to make a better estimation than either system could do independently [LHL09, pp. 2-3]. The sensors complement each other best when their errors are independent of each other [Str+08, p. 84]. Because the position estimation relies on the *probability theory* a short introduction to the fundamentals of stochastics will be given in this section. Basically sensor data fusion methods can be classified into *static estimators* and *dynamic estimators* [Str+08, p. 83]. Representatives of both categories will be reviewed.

2.2.1. Fundamentals of Stochastics

"The aim of stochastic filters is to estimate the state of a system on the basis of measurements. The measurements, as well as the system model itself, can be afflicted with uncertainties and errors that can be described with stochastic concepts" [Wen07, p. 117]. In the following, some terms of the field of stochastics will be introduced. The definitions presented in this section are primarily taken from [Wen07, para. 5].

Random Variable

Mathematically the *random variable* is a function that assigns values to the results of a random experiment. These values are called *realizations.* Two different types of random variables can be distinguished: discrete and continuous random variables. Discrete random variables are mapped to values of a countable set, whereas continuous random variables are mapped to arbitrary values. The *probability density* is a possibility to describe random variables.

Probability Density

The *distribution function* $F_x(\xi)$ of a random variable x is used to express the probability P, that realizations of x are less or equal than a specific bound ξ:

$$F_x(\xi) = P(x \leq \xi) \tag{2.1}$$

The distribution function is also called *accumulative probability.* Its first derivate is the *probabilty density function (PDF).* The probability density is a nonnegative function and the area under an unbounded PDF is 1. The probability that a realization of a random variable falls within a specific interval $[\xi_0, \xi_1]$ is calculated by the integration of the PDF over this interval (see Equation 2.2).

$$P(\xi_0 < x <= \xi_1) = \int_{\xi_0}^{\xi_1} p_x(u)du \tag{2.2}$$

The *mean* respectively *expectation* and the *variance* are important characteristics of the PDF. The variance σ_x^2 describes the distribution of realizations of the random variable around the mean value and is a nonnegative value. The square root of the variance is called *standard deviation.*

The probability that the realization ξ occurs for the random variable x and the realization η occurs for the random variable y is called *compound possibility.* If a certain realization η of y is given, the probability that realizations ξ of x will occur under this condition is calculated by the so called *conditional probability* $p_{x|y}$. One has:

$$p_{x|y}(\xi|y = \eta) = \frac{p_{xy}(\xi, \eta)}{p_y(\eta)}. \tag{2.3}$$

This relationship is also known as *Bayes' rule.* If, in contrast, a realization ξ of x is given one has:

$$p_{y|x}(\eta|x = \xi) = \frac{p_{xy}(\xi, \eta)}{p_y(\xi)}. \tag{2.4}$$

This gives an alternative formulation of the Bayes rule:

$$p_{yx}(\xi|\eta) = p_{x|y}(\xi|y = \eta)p_y(\eta) = p_{y|x}(\eta|x = \xi)p_x(\xi). \tag{2.5}$$

Gaussian Distribution

In the area of stochastic filters, the *Gaussian* or *normal distribution* is of particular interest because this distribution is suitable to describe a plurality of random phenomena. Sources of interference of an observable parameter can be described by random variables. The distribution function of each random variable is arbitrary but they have to be independent. Under these conditions the distribution function tends to the normal distribution with an increasing number of random variables. The normal distribution can be completely described by mean and variance. Figure 2.9 illustrates the probability density function for different values of mean (μ) and variance (σ_x^2). The red curve describes the standard normal distribution with a mean value $\mu = 0$ and a variance $\sigma_x^2 = 1$.

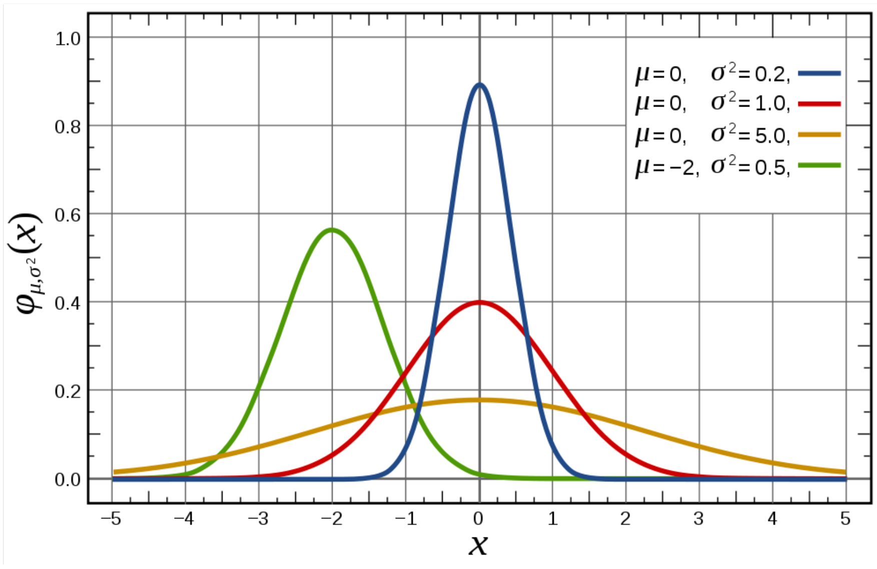


Figure 2.9.: Different normal distribution PDFs with varying values of mean and variance. [Ind08]

2.2.2. Static Estimators

Static estimators can be used to fuse data of multiple sources provided that the position estimation of the object to be tracked is nonrecurring or the object is not moving [Str+08, p. 83]. The positioning sources are described as random variables via a PDF. PDFs of multiple independent sources, that are spanned above the same state space, can be multiplied to obtain an optimal estimation [Str+08, p. 85]. Figure 2.10 shows the fusion of two sources. The PDFs of both sources are illustrated in 2D and 3D representations. The results of the multiplication of both PDFs produces high values only for those areas, where both PDFs are indicating a high probability [Str+08, pp. 85-86].

2.2.3. Dynamic Estimators

Static estimation cannot be used for tracking moving objects continuously. In this case, dynamic estimators are utilized. For navigation applications, dynamic estimators are of particular importance because the objects being tracked are likely to be moving. In the area of tracking objects and navigation, the *Kálmán filter (KF)* and the *Particle filter (PF)* are commonly implemented to combine data from different positioning and sensor sources. These two different techniques will be reviewed. Because these filters are based on *recursive Bayesian estimation*, also known as *Bayesian filter*, and *Markov chains*, these two concepts will be shortly introduced.

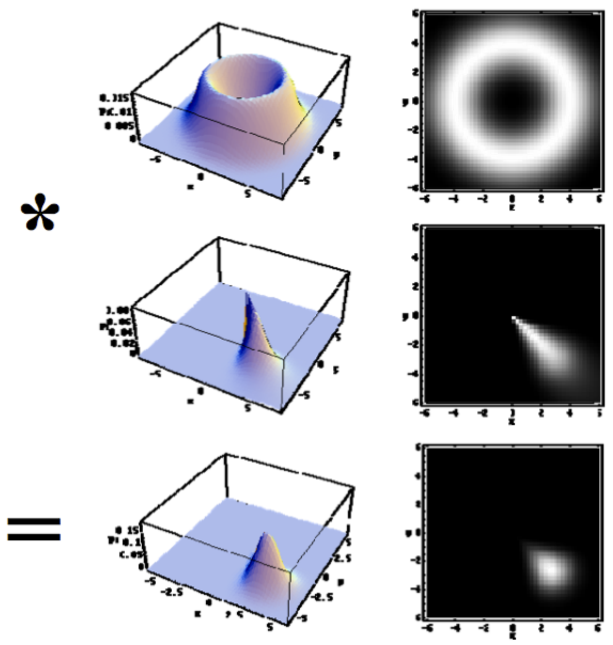


Figure 2.10.: Static estimation: multiplying the PDFs of two independent sources. [Str+08, p. 86]

Markov Chains and Bayes Filter

Markov chains are capable of inferring future states of a system from having knowledge about the present and previous states. The number of possible states is finite, all states are forming the *state space*. Furthermore, a *probability vector* describes the probabilities of states being the starting point. With each state, a *transition probability* is connected [Beh00, p. 8]. The case, that the present state is sufficient to infer the next state and no further information about previous states is needed, is known as *Markov process* [Beh00, p. 12]. According to [Han08, pp. 4-5], the *hidden Markov model* is a Markov process, "where a stochastic system is observed through noisy measurements". The parameters of the original system cannot be observed directly and the measurement might not represent the parameters exactly, hence they are only estimates. Figure 2.11 illustrates such a hidden Markov process.

Equation 2.6 and Equation 2.7 formally describe the evolution of the system (state and measurement) as discrete-time stochastic model [Wid10, p. 25]:

$$x_t = f_{t-1}(x_{t-1}, n_{t-1}) \tag{2.6}$$

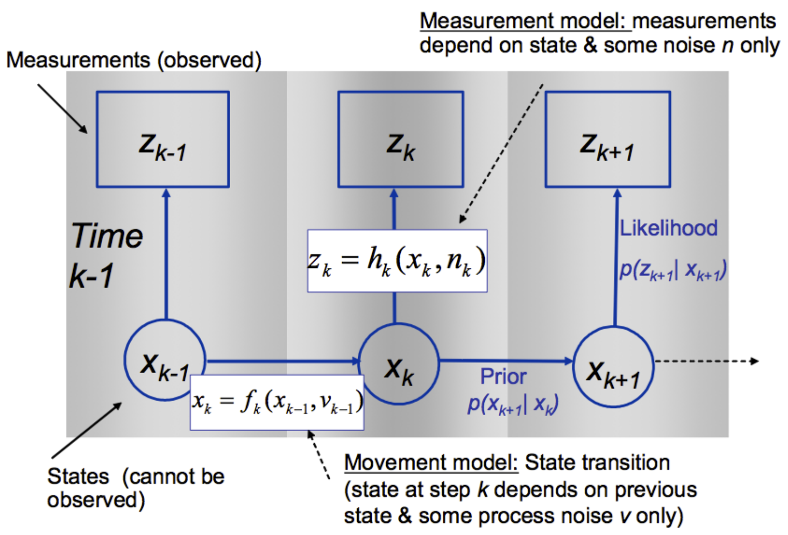

Measurement model: measurements depend on state & some noise n only
Measurements (observed)
z_{k-1}
z_k
z_{k+1}
Time k-1
$z_k = h_k(x_k, n_k)$
Likelihood
$p(z_{k+1}| x_{k+1})$
x_{k-1}
x_k
x_{k+1}
Prior
$p(x_{k+1}| x_k)$
$x_k = f_k(x_{k-1}, v_{k-1})$
States (cannot be observed)
Movement model: State transition (state at step k depends on previous state & some process noise v only)

Figure 2.11.: Illustration of a *hidden Markov process*: the system state can only be observed indirectly through measurements. [Str+08, p. 88]

$$z_t = h_t(x_t, e_t) \tag{2.7}$$

x_t denotes the state vector at time t and z_t the measurement vector at time t. The, possibly non-linear, functions f_{t-1} and h_t are known and n_{t-1} respectively e_t are "independent and identically-distributed noise" [Wid10, p. 26]. Probabilistic distributions have to be defined to obtain new state vectors and measurement vectors. It is assumed that the system is a *Markov process*, consequentially the last state is sufficient to predict the present state, which is expressed by Equation 2.8 and called *transition probability*.

$$p(x_t|x_{0:t-1}, z_{1:t-1}) = p(x_t|x_{t-1}) \tag{2.8}$$

Recursive Bayes filters are used to calculate the *posterior probability*. Hereby the "posterior probability represents the probability distribution of the state x_t given all available measurements" [Wid10, p. 26] (see Equation 2.9). The *prior probability* is often called *prediction* and can be calculated based on the previous posterior probability (see Equation 2.10).

$$p(x_t|z_{1:t}) = p(x_t|z_t) \tag{2.9}$$

$$p(x_t|z_{1:t-1}) = p(x_t|z_{t-1}) \tag{2.10}$$

A *prediction* and an *update step* is utilized to construct the posterior probability. In the prediction step, the prior probability is calculated through the *Chapman-Kolmogorov* equation (see Equation 2.11). Afterwards, this prediction is corrected through a new measurement, that is incorporated during the update step (see Equation 2.12). [Wid10, p. 27]

$$p(x_t|z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1} \tag{2.11}$$

$$p(x_t|z_{t-1}) = \frac{p(z_t|x_t)p(x_t|z_{t-1})}{\int p(z_t|x_t)p(x_t|z_{t-1})dx_t} \tag{2.12}$$

Here, $p(z_t|x_t)$ is the measurement probability, $p(x_t|x_{t-1})$ is the transition probability and $p(x_t|z_{t-1})$ the previous posterior probability [Wid10, p. 27]. In general, the recursive propagation of the posterior density cannot be determined analytically. If the restrictions of a linear system and Gaussian distributions apply, it can be solved using Kalman filters and its derivates. Particle filters can be used to approximate the problem [Wid10, p. 28].

Kálmán Filter

The *KF* was originally introduced by Kálmán in [Kál60]. It is based on a *hidden Markov process* and used to estimate the state of a linear system. Therefore, measurements are processed that are linearly related to the system state [Wen07, p. 129]. System and measurement noise, which are assumed to be zero-mean and Gaussian distributed, are taken into account within a *movement* and a *measurement model* [Str+08, pp. 89-90].

Predictions of future states are based on the movement model. These predictions are evaluated on the basis of the measurement model. If the restrictions apply of a linear system model and Gaussian-distributed, zero-mean noise apply, the Kalman filter finds the optimal weighting factor between the previous state and the predicted state based on the measurements [Str+08, p. 90].

The disadvantage of KFs is the restriction of a linear system model. Several enhancements, such as the *Extended Kalman filter (EKF)* and *Unscented Kalman filter (UKF)*, target these restrictions. EKF and UKF do not share the characteristic of being optimal estimators with the original Kalman filter [Str+08, pp. 90-91].

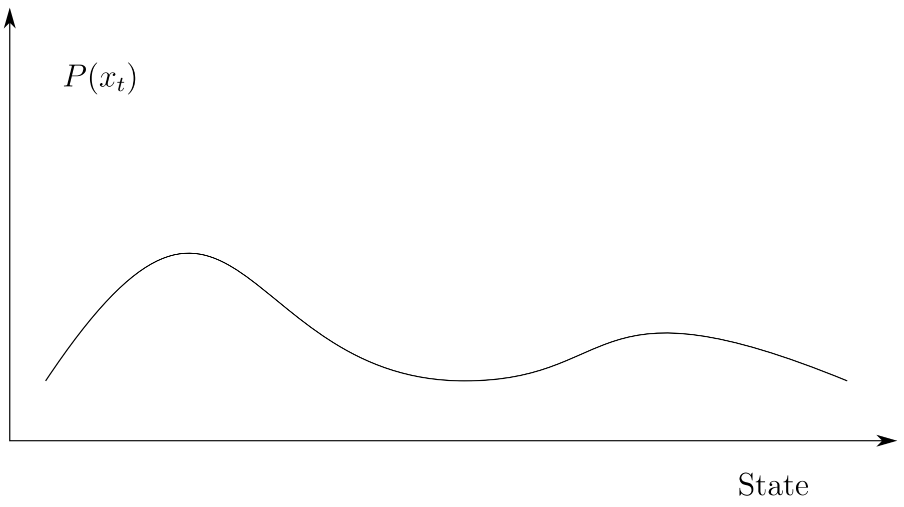


Figure 2.12.: Illustration of a non-Gaussian probability density function. [Bil12, p. 20]

Particle Filter

The previously mentioned restrictions of KFs do not apply to most praxis-relevant estimation problems. The *PF*, also known as *sequential Monte Carlo method (SMC)*, is facing the restrictions of KF and its derivates [Mit10, p. 195]. It was first introduced in [GSS93]. The Particle filter is based on the Bayesian filter. The posterior probability of the state is approximated by a finite number of *particles* which represent discrete samples of the posterior probability, each associated with a weight. For an infinite number of particles the approximation would be optimal. In the case of a finite number of particles the approximation is not optimal, but as long as the number of particles is not too low, the difference is negligible. The probability density is not restricted to be Gaussian-distributed (*non-parametric*) [Wid10, p. 28] (see Figure 2.12).

Every particle is a hypothesis of the system state. The probability of this hypothesis to be the true state is given by the associated weight [Wid10, p. 29]. Initially the particles are distributed according to some distribution function. If the initial state cannot be narrowed down, the particles are evenly distributed over the state space (see *Init* in Figure 2.13). With every iteration of the algorithm each particle is moved randomly according to the movement model (*prediction step*) (see *red arrows* in Figure 2.13). Therefore, more particles will perform rather likely movements than unlikely movements. When new measurements are received the particles are evaluated and (normalized) weights are assigned according to the likelihood (see *size of particles* in Figure 2.13). This is called the *update step.* The steps are performed iterative and thus, a *particle cloud* is *exploring* the state space, with probability density described by all particles with the associated weights [Str+08, pp. 91-92][GSS93, p. 108] (see Figure 2.13).

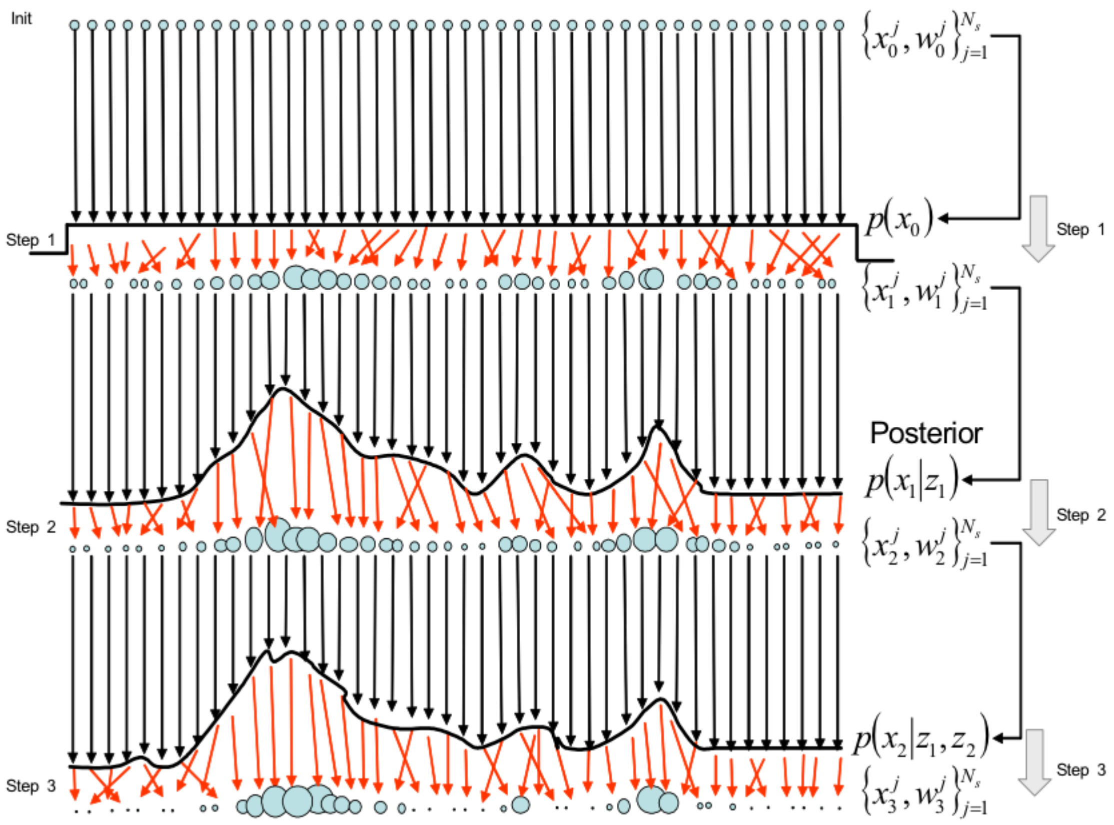

Init
$\{x_0^j, w_0^j\}_{j=1}^{N_s}$
Step 1
$p(x_0)$
Step 1
$\{x_1^j, w_1^j\}_{j=1}^{N_s}$
Posterior
$p(x_1|z_1)$
Step 2
Step 2
$\{x_2^j, w_2^j\}_{j=1}^{N_s}$
$p(x_2|z_1,z_2)$
Step 3
Step 3
$\{x_3^j, w_3^j\}_{j=1}^{N_s}$

Figure 2.13.: Ill. of particles in consecutive steps of a particle filter. [Str+08, p. 93]

The general method described as before is called *importance sampling* and was used in early implementations of the particle filter. Generally the samples can't be obtained directly from the posterior so a related distribution is used to sample the particles [Wid10, p. 30]. As is evident from Figure 2.13 after a several iterations only a few particles will contribute to the PDF. Most of the particles will have such low weights that they are insignificant for the distribution whereas a few particles with high weights will dominate it. This problem is called *degeneracy* [Wid10, pp. 30-31]. *Resampling* is a method to overcome this problem and is introduced as a additional step in the process of the particle filter. Different approaches are feasible to avoid the degeneracy problem. Widyawan describes one possible approach in [Wid10, pp. 31-32]:

Algorithm 2.1 Resampling Algorithm ($\overline{\chi}_t$) according to [Wid10, p. 31]

```
1:  $\chi_t^* = \emptyset$
2:  for $j = 1$ to $N$ do
3:      $r$ = random number uniformly distributed on $[0, 1]$    ▷ Threshold weight sum
4:      $s = 0.0$                                                ▷ Resetting weight sum
5:      for $i = 1$ to $N$ do
6:          $s = s + \omega_t^i$                                 ▷ Sum up the weights
7:          if $s \geq r$ then                                   ▷ Threshold exceeded?
8:              $x_t^{*j} = x_t^i$                               ▷ Resample particle
9:              $\omega_t^{*j} = 1/N$                            ▷ Reset weight
10:             $\chi_t^* = \chi_t^* + \{x_t^{*j}, \omega_t^{*j}\}$
11:         end if
12:     end for
13: end for
14: replacement $\overline{\chi}_t = \chi_t^*$                   ▷ Use resampled set
```

The algorithm in Algorithm 2.1 will be described in detail in the following:

- The *resampling step* is performed with every iteration of the particle filter. The set of particles $\overline{\chi}_t$, which are sorted in ascending order according to their weight, is the input for the resampling algorithm. The sorting of the set is of particular importance: the aim of the resampling step is to eliminate particles with low weights and to take particles with high weights. If the set is not sorted in ascending order it is more or less random at which iteration the sum of weights of the particles will exceed the threshold and which particle will be drawn. The output is a set of resampled particles χ_t^*.

- A uniformly distributed random number r is generated for each step of the iteration over the input particle set (see Algorithm 2.1 line 3). This number r is used as threshold to determine the particle that will be taken over to the output set of

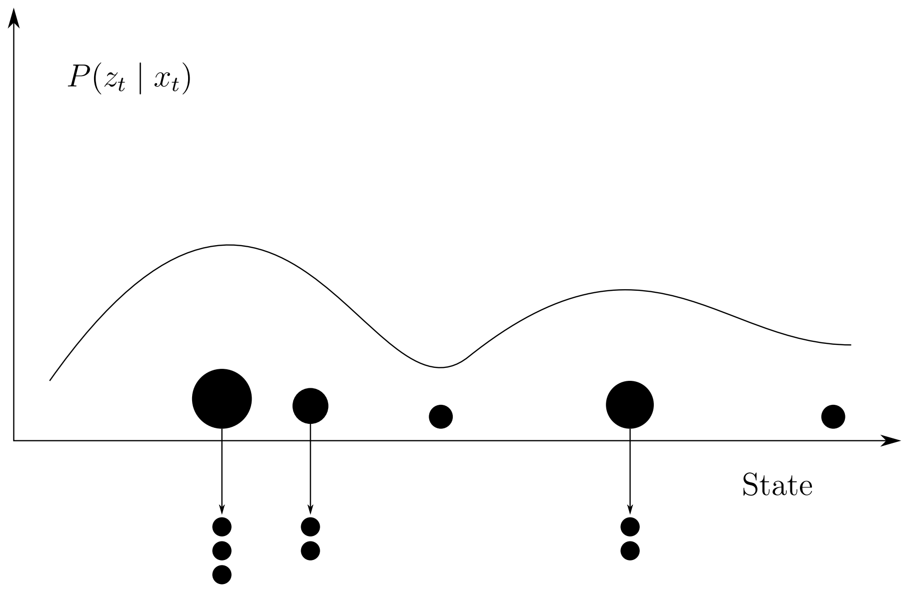


Figure 2.14.: *Resampling step* to avoid *degeneracy problem.* [Bil12, p. 24]

resampled particles. The weights of the sorted particles are summed up as long as the sum s is lower than the random number r. The particle whose weight is responsible that s is equal to or exceeds r is drawn to be transferred to the output set of resampled particles (see Algorithm 2.1 line 7 to 11).

- The chance of a particle to be drawn from the input set is in proportion to its weight (*importance factor*). The idea is, that particles with low weights hardly add to s and therefore most likely will not be drawn whereas particles with high weights will do.
- The weight of all drawn particles will be set equally because the resampling step will be performed with every iteration of the particle filter and no knowledge about the previous weight of a particle is needed (see Algorithm 2.1 line 9).
- The set of resampled particles will be used in the next step of the particle filter (see Algorithm 2.1 line 14).

Figure 2.14 visualizes the resampling process. Particles with previously high weights are multiplied and particles with low weights are discarded. The PDF is hardly changed by this process due to the insignificant weights of the discarded particles.

Different approaches are feasible to obtain a concrete state that will most likely represent the system state from all particles that approximate the PDF. The simplest method is to chose the particle with the highest weight at the questioned time.

2.3. Navigation

Navigation is one of the most important tasks of human beings. Its importance derives from the fact, that everybody is concerned in their everyday life and therefore it was one of the earliest activities of human beings [HLW03, p. 9]. In the beginning, navigation was based on observations, primarily on landmarks and celestial bodies [HLW03, p. 9][Kar11, p. 1]. Later, devices and tools have been developed to support navigational tasks [Kar11, p. 1]: the *South Pointing Chariot* is probably one of earliest navigation device and was invented around 2600 B.C. Differences of the rotation of wheels of a two-wheeled vehicle were used to let an object, that is connected to the wheels, always point in the same direction. The invention of the *odometer*, a device for measuring distances, and the magnetic compass enabled the people of that time to navigate more accurately and reliable over great distances. The creation of maps (*cartography*) and their utilization was also very important for the ease of traveling.

In this section, the fundamentals of navigation will be introduced. Because map representations are very common in navigational applications the types and characteristics of maps will be described. In addition to position determination, which was discussed in the previous sections, guidance is the other fundamental component of navigation and is primarily based on routing problems [HLW03, p. 299]. At last, the specific characteristics of indoor navigation in comparison to the well researched and established outdoor navigation will be outlined.

2.3.1. Presentation of Navigational Instructions

As previously stated, navigation consists of positioning and guidance [HLW03, p. 77]. Several technologies are available to link both. Commonly used technologies to present navigational instructions to users, particularly those suitable for adoption on mobile devices, will be described and state of the art examples given, where reasonable.

Maps

“Maps are planar representations of parts or all of the earth’s surface” [HLW03, p. 77] and the position of objects and relations between multiple objects can be visualized with maps. Maps are a “medium for communication [...], regardless of language and culture” [Abe08] and widely accepted due to their ease of use.

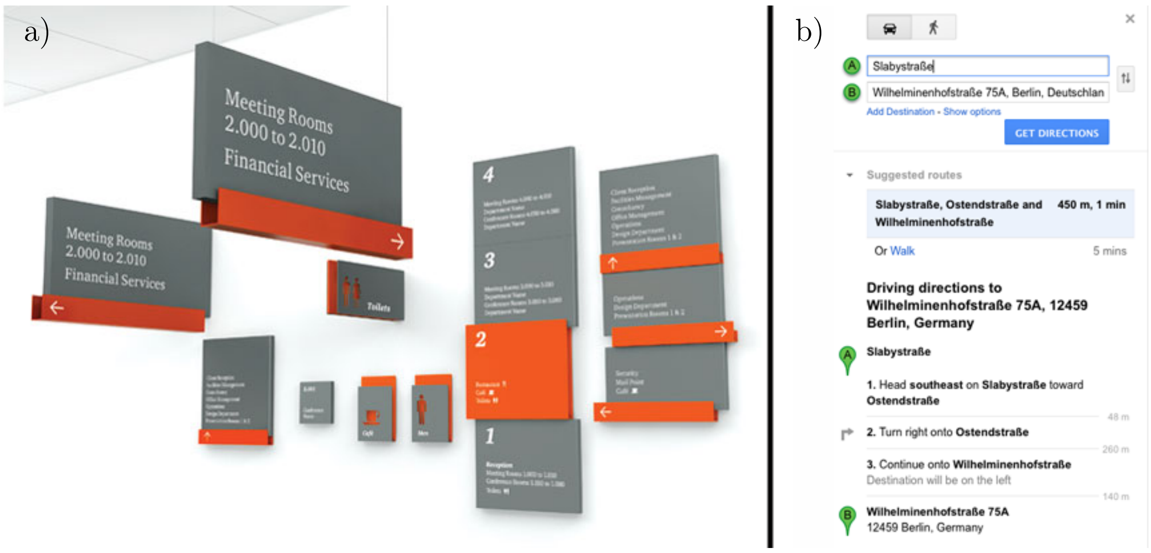
The type of a map depends on its purpose. In [HLW03, p. 77] some characteristics of maps are outlined. According to this, the main attributes of a map are the application (land-used, maritime, aeronautic), the contents (topographic, thematic) and attributes related to the appearance of the map, such as projection, scale, and orientation. Another characteristic emerged with the upcoming of computers. Until then, all maps were analog and changing the presentation or adding information as required was very difficult or simply not possible.

In the 1980s the technology of geospatial information systems (GIS) was emerging and "played a key role in the development of modern navigation technology" [Kar11, p. 3]. GIS store geographical data that can be used to model the real world. For navigational purposes the modeling of the navigational infrastructure is of particular interest [HLW03, p.83]. Nowadays, GIS provide diverse information to create digital maps on demand that can be customized to suit the users current needs and context. According to Hofmann-Wellenhof, Legat, and Wieser [HLW03, pp. 83-84], the main modeling components of a GIS are *geometry* and *semantics*. Geometry incorporates metrics, two- or three-dimensional coordinates, and topology, which specifies the properties of the objects. Semantics describes thematic information. The authors suggest to distinguish between node-specific semantics (e.g. traffic lights), node-to-node (edge) semantics (e.g. road class), and edge-to-edge semantics (e.g turn restrictions).

While GIS is very important for outdoor navigation, it plays only a minor role for indoor navigation. Information about buildings is often provided by architectures and building companies. These often use *computer-aided design (CAD)* and *Building Information Models (BIM)* to plan and maintain buildings. Information regarding the topology and semantics of a building can be extracted from the data provided by these technologies [GM03], [Miu02, pp. 35-45]. A current field of research is to seamlessly link building information with outdoor GIS. Simply establishing a centralized system, like outdoor GIS, is not practicable because the amount of data will be much higher. Furthermore, indoor environments change more often and exact details are usually known only to the operator of the building. [Sch09] proposes a decentralized, federal system, that integrates information systems for outdoors and indoors. However, even with the concern stated above regarding centralized systems, there are also efforts in *Open Street Map* to map indoor environments[5].

The amount of information shown on a map is very important, particularly on mobile devices and the restrictions that come with these. User can be overwhelmed very quickly if too much information is provided. In [Men08], [BC08], [Jok08], and [Pui+09] questions are stated, that a designer of user interfaces for mobile map-based applications should consider, and practical advices given to be able to build an effective and intuitive interfaces

[5]IndoorOSM: `http://wiki.openstreetmap.org/wiki/IndoorOSM`



Slabystraße

Wilhelminenhofstraße 75A, Berlin, Deutschlan

Suggested routes

Slabystraße, Ostendstraße and Wilhelminenhofstraße 450 m, 1 min

Or Walk 5 mins

Driving directions to Wilhelminenhofstraße 75A, 12459 Berlin, Germany

Slabystraße

1. Head **southeast** on **Slabystraße** toward **Ostendstraße**

48 m

2. Turn right onto **Ostendstraße**

260 m

3. Continue onto **Wilhelminenhofstraße**
Destination will be on the left

140 m

Wilhelminenhofstraße 75A
12459 Berlin, Germany

Figure 2.15.: a) Example for signage in public buildings. [LON11] b) Example of textual navigation instructions in *Google Maps*. [Goo12b]

that satisfies the user. One outcome is, that are more comprehensive if they show less details but emphasize landmarks in a way that these are recognizable in the surrounding [Pui+09, pp. 3-4].

Text, Audio, and Signs

Textual instructions and signs are further established ways to communicate navigational directions and, like maps, they are part of our everyday life. These kinds of representation are used in almost every public building as part of the signage system that helps visitors to navigate. Verbal instructions are often used in personal (mobile) navigation systems, such as car navigation. With respect to these systems, current research tries to generate the instructions in natural language, which is more easily understood by the user [Jok08]. This includes that the instructions do not rely on information about exact distances, such as “go 50 meters” [Pui+09], [Reh+05, pp. 3-4]. Instructions that incorporate references to visible landmarks, such as “go left at the stairs”, should be preferred. Another field of research is the combination of public displays, that are distributed in the environment, and a users mobile device to help the user to orient oneself. The authors of [HG10] name the systems *GAUDI*, *Smart Signs*, and *Rotating Compass* that fall in this category. The benefit of theses systems is, that the users do not have to look at their mobile devices “head down” while walking through unfamiliar environments [HG10, Ch. 3.3].

2.3.2. Routing and Guidance

As stated before, guidance "is very often based on routing problems as the optimal path, the travelling salesman problem (TSP), etc." [HLW03, p. 299]. Routing involves route planning and path finding which are based on shortest-path algorithms such as Dijkstra's shortest-path algorithm. These topics will be covered briefly in the following section.

Graph Theory

A *graph* $G = (V, E)$ consists of a finite set of *nodes* V (vertices) and a set of *edges* E. The edges are represented by tuples (x, y) of the set $V \times V$. In *undirected graphs* an edge (x, y) indicates a connection between the two nodes x and y, whereas in directional graphs an edge (x, y) indicates that the edge is oriented from x towards y but not vice versa [Kle97, p. 11]. This edge in an directed graph is often called an *arc* [HLW03, p. 300]. A set of edges which connects the nodes $x_1, x_2, ..., x_n$ is called *edge sequence* from x_1 to x_n. If $x_1 = x_n$ the edge sequence is called *closed*, otherwise it is called *open*. An open edge sequence is also referred as *path*, a closed edge sequence as *circle* [Har04, p. 205]. In paths, all edges are distinct from each other. Another distinguishing feature of graphs is *weighting*. In weighted graphs a weight c is assigned to every edge. The sum of the weights of all edges of a path is referred to as *length* or *cost* of a path [Har04, p. 206]. Negative weights are not generally forbidden, but in the navigational context (cost of travel) it is rarely used. The graph is extended by a cost function $c(x, y)$ that assigns the weight to an edge (x, y) $(G = (V, E, c))$ [HLW03, p. 300].

Shortest-path Problems

In navigational networks many shortest-path problems arise. The *shortest path* is defined as "the path with the minimal cost of all possible paths between [a start node and a destination node]" and its total cost is called *distance* [HLW03, p. 300]. Some of these problems are given in [HLW03, pp. 300-302]:

Single-pair shortest-path problem The most frequently posed problem is finding the shortest path between a start node x and a destination node y. It is used for pre-trip planning and during the trip itself, if the precomputed path is not longer available due to obstacles. The cost of the affected arc must be appropriately adjusted and a recomputation of the path is necessary.

Sequential shortest-path problem This problem occurs in route planning if a path between a start node and a destination node is demanded that should comprise other nodes to be visited in a specific order.

K-optimal shortest-path problem Find the second-, third-, ... best path. It can be used to find alternatives of a path that do not exceed a specified distance.

Single-source shortest-path problem Find all shortest paths between a source node and all other nodes. This is used to generate a search tree which is often used in shortest-path algorithms.

Dijkstra's shortest-path algorithm

The algorithm published by Dijkstra in 1959 [Dij59] belongs to the family of *greedy* algorithms and is one of the most frequently used to solve shortest-path problems [HLW03, p. 303]. It is restricted to directed, valuated graphs $G = (V, E, c)$ with non-negative values. The algorithm is optimal under the assumption of the restrictions, that means, that there cannot be shorter path from the start node to the destination node after the search. The idea of this algorithm is to always follow the edge that would lead to the current shortest-path section from the start node. Therefore, the nodes are extended by two properties: distance and predecessor. The distance property represent the actual distance from the start node to this specific node. The nodes can be members in three different sets. In the beginning all nodes part of the set V of the graph G. Nodes that have been visited by the search algorithm are moved to a set T, which contains nodes that are not fully reviewed and therefore have a temporary distance. Nodes that have been fully reviewed are moved to a set P. The distance of nodes in this set is permanent. The algorithm is described by the following steps (similar to [HLW03, p. pp. 303-305]):

1. Initially, the distance property of the start node is set to 0 and of all other nodes to ∞. The start node is moved to the set T and P is initialized as empty set.

2. The node with the minimal distance from set T will is set as current node. The distance to all neighbors of the current node is determined. For every adjacent node the distance property is set to the value of the distance of the current node plus the value of the arc connecting the current node and the neighbor. The neighbors are added to the set T and the current node is moved to set P.

3. Step 2 is repeated until no nodes are either in V or in T. If a neighboring node is already in T, the distance property and the predecessor property is updated only if the actual calculated distance is lower than the previously (temporary) distance stored for that node.

4. If only the shortest path between two nodes matters, the algorithm can be stopped as soon as a path was found that connects both nodes. The distance between the nodes is stored in the distance property of the destination node. The shortest path can be computed by following the predecessors starting at the destination node. This path represents the shortest path in reverse order. Otherwise the algorithm stops if all nodes have been moved to set P. In this case the distances from the starting node to all other nodes are known.

A visualization of Dijkstra's algorithm is given in [LSW09, pp. 29-34].

With algorithms that solve shortest-path problems, such as Dijkstra's algorithm or the $A*$ algorithm, a route between a single source node (starting point) and one or more destination nodes of a navigational network can be computed. This route can be computed to generate a list instructions prior to the trip. To be able to guide the user during the trip and to be able to recompute the route if the user left the precomputed route, another technique have to be utilized: *map matching.*

2.3.3. Map Aiding and Map Matching

Map aiding (MA) refers to the utilization of "suitable information of a digital map [...] to complement the information provided by one or more primary sensors" [HLW03, p. 295]. For example, the trajectory of an object can be following the current edge of a map although the primary sensors are not able to deliver new measurements, thus, the geometric information provided by the map aiding technique can be used to bridge outages of primary sensors [HLW03, p. 296]. Map matching (MM) is prerequisite for *route checking.* This task monitors the position of the object in relation to the computed path and will report when a node is reached where guidance is necessary [HLW03, p. 335].

However, *MM* refers to the projection of an object to the modeled navigational network of a digital map. Due to uncertainties of measurements of the sensors, the real trajectory of an object will not follow the path of the navigational network exactly. With the MM technique, the real trajectory will be mapped to a path of the digital representation of the network, resulting in a georeferenced trajectory (see Figure 2.16) [HLW03, p. 330]. Therefore MM is a prerequisite of MA because without matching the real trajectory of an object to a modeled path in the node-edge structure of the digital map, the current path on which the object is currently moving could not be determined [HLW03, pp. 329-330]. The steps involved in implementing MM are discussed in [HLW03, pp. 331-333].

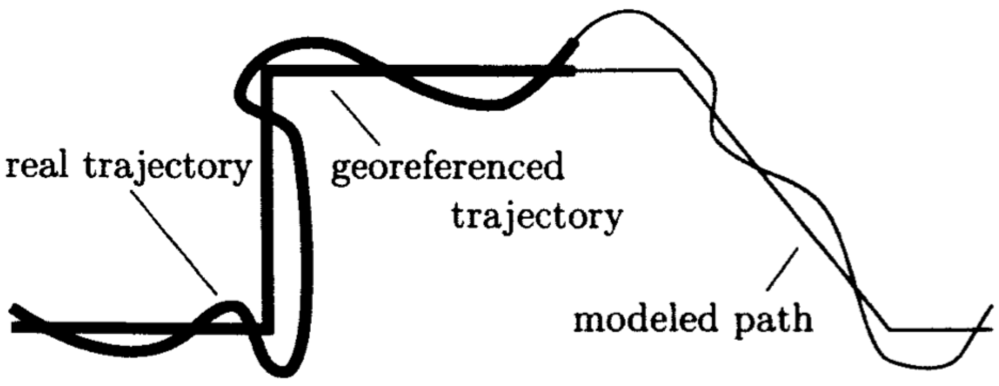


Figure 2.16.: Principle of map matching. [HLW03, p. 330]

2.3.4. Indoor Navigation

Indoor and outdoor navigation systems are similar in many ways. Nevertheless, there are a few aspects that have to be considered especially for indoor navigation systems. Most important is the high level of mobility of human beings. This "requires the use very detailed and up-to-date map data since even the smallest passages are used by pedestrians" [HLW03, p. 377]. Indoor navigation systems also demand higher accuracies to be able to reliably locate a user. On the other side, indoor navigation systems should be able to compute barrier-free routes for wheelchair users [Kar11, p. 10]. Also the two-dimensional maps, commonly used in outdoor navigation systems, are not suitable for multistory buildings, making three-dimensional maps necessary. For pedestrians a seamless transition between outdoor and indoor environments is very important as well [HLW03, p. 377] because indoor environments are usually very limited and buildings are often connected only via outdoor areas.

In comparison to outdoor environments, the complexity of the network is lower due to the smaller size of the covered area. Indoor navigation systems also feature lesser point of interest (POI) categories and do not have to support such factors as weather and traffic situation [Kar11, p. 13].

2.4. Summary

In the last section the fundamentals of positioning, sensor data fusion, and navigation have been covered. Several key technologies for positioning with smartphones have been reviewed and several state-of-the-art positioning systems for every technology were presented. Furthermore, the fundamentals of sensor data fusion were introduced and with the particle filter, a convenient technique for fusing data of different sensors was presented. Sensor data fusion is especially important for indoor navigation due to the absence of a specific technology that is able to provide reliable position information in the majority of indoor environments. To be able to guide a user through a building, the principles of navigation had to be reviewed. Common presentation forms of navigational instructions were introduced and with *map matching* and *map aiding* two techniques that are relevant for linking the two main components of navigation, which are positioning and guidance, were described. Positioning technologies that will be utilized in the prototypic implementation of the *multimodal positioning service* must meet two general requirements. Suitable technologies are commonly integrated in actual smartphones or can be easily attached/accessed via standard communication technologies. Off-the-shelf consumer hardware is preferred to avoid the need of specialized devices. Therefore, an easy and unintrusive installation and low maintenance costs are of particular importance for the operator of buildings that want to provide positioning services. For users the ease of use is most important. While they are willing to use dedicated devices for car navigation they might not be happy to carry a further device for indoor navigation. Furthermore, the positioning technologies should provide reliable and accurate position information in indoor environments and the immediate surroundings to be able to navigate the user through the environment.

Technologies that meet these requirements are RFID, NFC, and optical QR codes. Whereas cameras, that can be used to capture optical codes, are standard commodity of actual smartphones already, the distribution of phones integrating the NFC technology is steadily growing. The main drivers of this trend are applications for mobile payment. Nevertheless, this technology can also be used to obtain accurate position information. RFID technology cannot be found often in smartphones. However, an RFID-based infrastructure that is capable of providing positioning services can be implemented with the RFID tags of the *OpenBeacon* project. Some types of *OpenBeacon* RFID tags are equipped with Bluetooth which is an integral part of almost all actual smartphones. In this way, the RFID tags can be easily connected to smartphones. GPS can be used to cover the neighboring environment of indoor areas.

Other RF-based technologies, such as Wi-Fi and Bluetooth, are commonly used in indoor navigation research projects. The advantage of these approaches is that the hardware and infrastructure is available in most buildings. However, collecting the data needed for positioning (RSSI/fingerprints) and keeping it actual is very time consuming. Changes of the interior of the building and even a different number of people in the surrounding of the base stations can have a significant influence on the positioning results. Therefore, these approaches are not practical for the targeted scenario.

3. Specification Analysis

In the previous section potentially usable technologies were introduced and related available systems for positioning and navigation have been reviewed. On this basis, the system environment and requirements for the indoor navigation component to be developed will be specified.

Navigation systems rely on the position information of a positioning component. This component must be able to provide reliable information in the majority of scenarios that should be covered. Most available navigation systems integrate a positioning component that meet the requirements of the specific application. For outdoor navigation, this is commonly GPS. For indoor navigation, the technologies used in the positioning system vary substantially and are often related to the preconditions and requirements of the specific building for which the system was developed.

In this work, the positioning component and the indoor navigation component are separated and independent of each other. Separating these two components has several advantages. Some of them are stated in section 3.3. However, navigation systems, and positioning systems in particular, are often researched and developed with regard to certain positioning technologies. Several examples for research work in this area have been given in subsection 2.1.1 according to the utilized technologies. Little research has been conducted to universally combine sensor data from different sensors and positioning systems. A comparable system to the *multimodal positioning service* has been developed in [Wen+06]. Here, different sensors are attached to a mobile station, in this case a laptop, and the data is fused using a particle filter. Unlike the *multimodal positioning service*, the sensor data is not processed on the mobile device. The sensor data is rather sent to a server that executes these tasks. To be mostly independent from external online services and additional infrastructure the processing of sensor data and visualization of the position information should be performed on the mobile device. Furthermore, the system presented in [Wen+06] utilizes custom hardware for the sensor systems. The approach presented in this work relies on consumer electronics, such as the *Google Nexus S* smartphone with its integrated sensors and generally available technologies, such as QR barcodes, NFC tags, and *OpenBeacon* RFID tags. The usage of Wi-Fi and Bluetooth positioning methods based on fingerprints or RSSI values is abandoned due to the huge effort of the initial creation of the fingerprints respectively the *RSSI heat map* and maintenance of these. Moreover, these values fluctuate significantly in varying environments, in particular in the presence of a varying number of human beings.

3.1. System Environment

A positioning system, called *multimodal positioning service*, which was developed by the author in the context of a research project during the masters program at the HTW Berlin, will be utilized as positioning component and enhanced as needed. Different stages of the *multimodal positioning service* are presented in Bittins [Bit11], [Bit12b], [Bit12a]. It describes the general architecture of a positioning system that is able to generically integrate different sensors and external positioning systems. The basic features of the system are described in section 3.2. The developed architecture is not restricted to a certain mobile platform. However, the prototype of the system is implemented on Google's *Android* platform.

The positioning component and the indoor navigation component, will be operating in the same environment. In this case it will be a smartphone. smartphones integrate a variety of sensors and provide multiple communication capabilities that enables the device to connect to external systems. Moreover, targeting a current smartphone platform obliterates the need to carry another device solely for indoor navigation. Because the *multimodal positioning service* was implemented for the *Android* platform, the indoor navigation component will be developed for the same platform. The *multimodal positioning service* was designed as *Android Service*. *Android Services* are a certain type of application on an Android-based system. Generally these services run in the background performing recurring tasks that do not require any user interaction. Figure 3.1 shows the general architecture of the whole system. The basis is a smartphone. The smartphone's operating system provides access to integrated sensors systems via interfaces. External positioning systems can be accessed via standard communication technologies, such as Bluetooth and Wi-Fi. The *multimodal positioning system* uses information provided by these sources to compute the users position. Third-party applications, such as the indoor navigation component that will be developed in the context of this work, can connect to the *multimodal positioning service* to obtain position information provided by it. The architecture of the specific components will be discussed later in more detail.

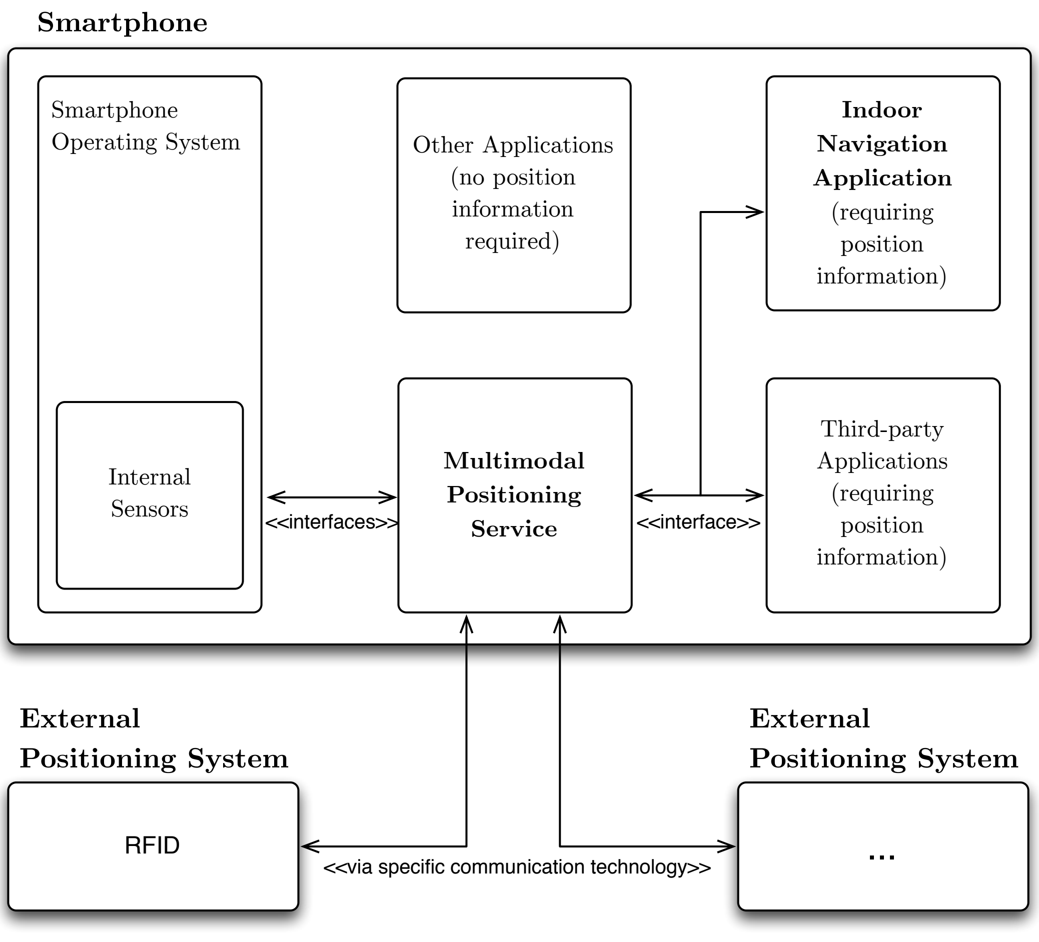

Smartphone
Smartphone Operating System
Internal Sensors
Other Applications (no position information required)
Indoor Navigation Application (requiring position information)
Multimodal Positioning Service
Third-party Applications (requiring position information)
<<interfaces>>
<<interface>>
External Positioning System
RFID
<<via specific communication technology>>
External Positioning System
...

Figure 3.1.: General architecture of the overall system.

The positioning component must operate in changing environments: indoors and outdoors. Therefore it is using multiple positioning technologies that are able to continuously provide reliable position information in the majority of vicinities. The indoor navigation component is designed primarily for indoor environments, such as office complexes, shopping malls, train stations, and others. For each building that should be covered by the indoor navigation component a navigational database must be provided. The configuration effort should be low for both systems. Navigational databases and other online information sources for positioning are assumed to be available locally or via wireless communication technologies at all times for convenient reasons.

3.2. Architecture of the Multimodal Positioning Service

Following, the architecture of the multimodal positioning service will be briefly introduced. A specification analysis and a conceptual design for a prototypic implementation have been done for the *multimodal positioning service* in the context of a research project prior to this work. The concepts will be outlined to provide an overview of the system. These insights of the system are relevant to understand the interaction between the *multimodal positioning service* and the indoor navigation component and to identify potential enhancements that could improve the reliability and accuracy of the position information.

In general there are two categories of positioning: absolute and relative positioning. Absolute position information is provided in the form of coordinates of a common, predefined coordinate system. Examples for components providing absolute position information are GPS, RF-based fingerprinting systems, and optical/RF-based tags that encode absolute position information. It is essential to be able to process relative movements of the user as well to be able to determine the position of the user even when no absolute position information is available. This leads to an improvement of the availability and reliability of the position information. Components providing relative position information are based on dead-reckoning mechanisms.

With regard to the process of retrieving position information the prototypic implementation of the *multimodal positioning service* incorporates two categories of positioning components:

- user-initiated retrieval of position information
- autonomous retrieval of position information

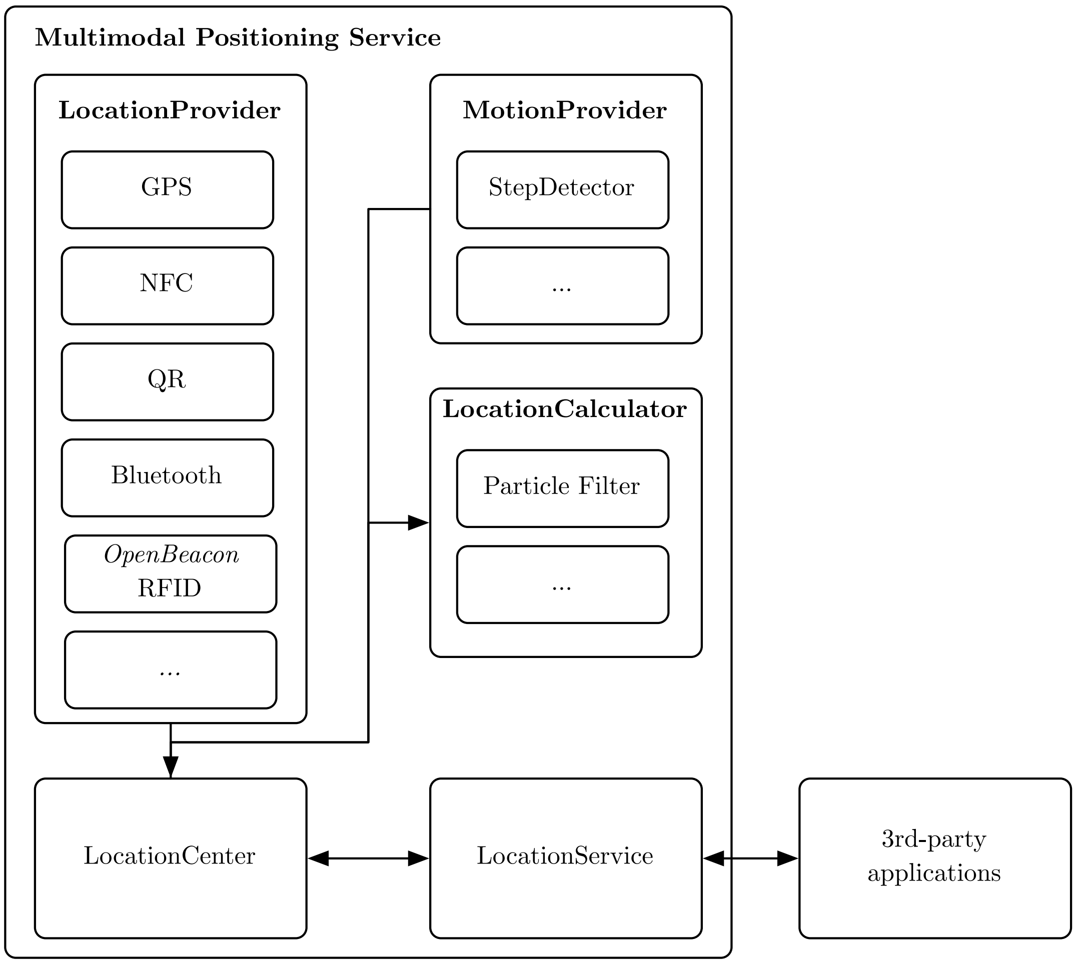


Figure 3.2.: Architecture of the *Multimodal Positioning Service.* [Bit12a]

For user-initiated components the user triggers the retrieval of position information actively, for example by scanning optical or RF-based tags, such as QR codes or NFC tags. Components of the other category do not require any user interaction. The position information is obtained continuously as long as the underlying sensor receives data. GPS, RFID and MEMS systems are representatives of this category.

The positioning service is sub-divided into three different classes of components which are responsible for providing position information (*positioning components*, see Figure 3.2)

LocationProvider Components of this class are able to process sensory data and data from other sources which provide absolute position information. That means the data can be used to derive coordinates in a specified coordinate system at any time. Representatives of this class are components based on GPS, wireless positioning systems, optical positioning systems, and others.

MotionProvider The components belonging to this class provide information about relative movements. Without information about an initial position these components are not able to provide coordinates that can be mapped to a certain position. MotionProvider components are typically based on integrated sensors, such as acceleration sensors, compass and barometer.

LocationCalculator Components of this class can use the information of components of the previous classes and fuse it to obtain more accurate, more reliable position information. By doing so the positioning service is not as susceptible to outliners of a specific providing component (*LocationProvider* or *MotionProvider*). Components implementing sensor fusion methods, such as Kálmán filter or Particle filter, are possible representatives. In Figure 3.3, the sequence diagram of the implemented *LocationCalculator* component „Particle Filter“ is presented.

By using positioning components of different types, it is very likely that position information will be available in the majority of environments. The *multimodal positioning service* integrates *LocationProviders* for GPS, QR, and NFC. A step and bearing detection component provides information about relative movements (*MotionProvider*). Furthermore, it is very probable, that the different sensors and positioning systems complement each other in a way, that after the process of data fusion, more accurate position information is available. The architecture is designed to be extensible to support the integration of further positioning components. To be able to fuse data from different sources, common data structures, that represent the position information of the different components in a uniform manner, had to be introduced. For each category of position information, absolute and relative, a data structure was developed. For representing absolute position information the coordinates and the accuracy of the position are essential. In order to be able to process information of a specific sources in a special way, for example to exclude GPS in indoor environments, it is necessary to provide information about the source of the position information. Furthermore, the data structure optionally holds information about the distance and direction (with respect to the last position) and other additional information (address, building, room, etc.). To represent relative movements different information is necessary. The distance and the direction of the movement and accuracies for both are essential. Listing 3.1 and Listing 3.2 show examples for each type of data structure.

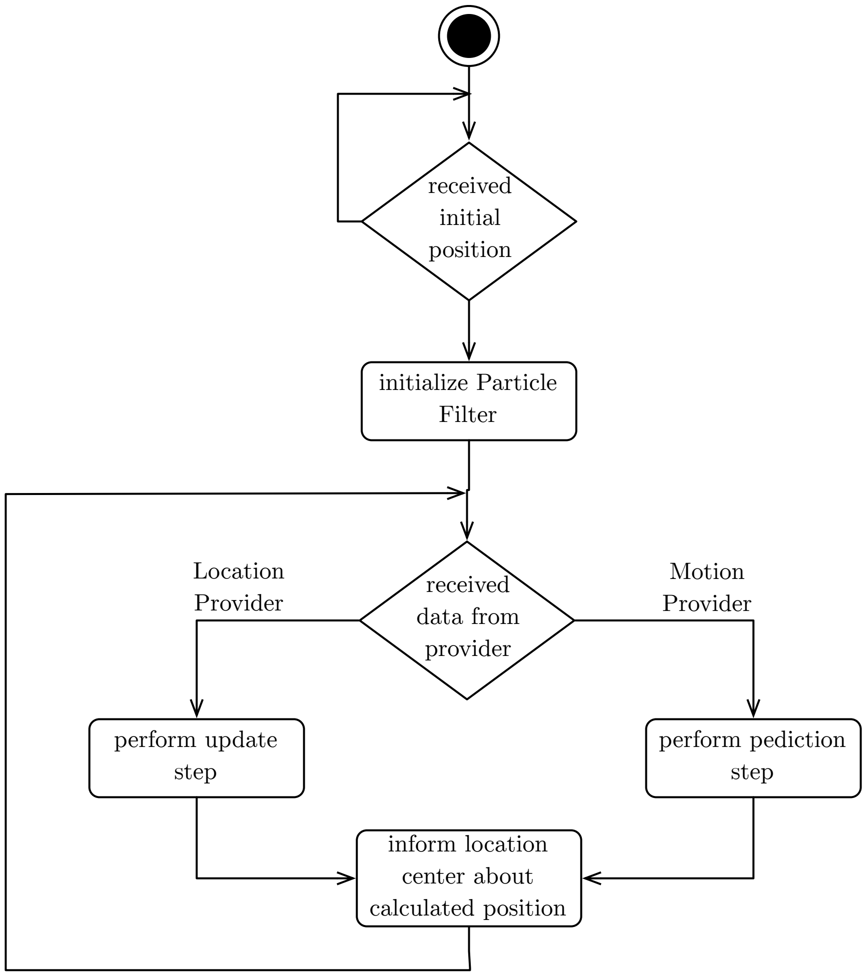

received initial position
initialize Particle Filter
Location Provider
received data from provider
Motion Provider
perform update step
perform pediction step
inform location center about calculated position

Figure 3.3.: Sequence diagram of *LocationCalculator* component „Particle Filter“. [Bit12a]

```
{ "position":
   { "-id":"Pos1",
     "lon":"13.533215671777725",
     "lat":"52.45765771339113",
     "alt":"0.0",
     "distance":"",
     "dir":"",
     "acc":
      { "-sensor":"qr",
        "-unit":"m",
        "-value":"0.1"
      },
   "detail":
      { "addr":"Slabystr. 25, 12459, Berlin",
        "building":"TGS building 2",
        "level": "0"
      }
   }
}
```

Listing 3.1: Example of the data structure representing absolute positions obtained from a QR code (JSON notation).

```
{ "motion":
   { "direction":"254.759",
     "dir_acc":"10.0",
     "distance":"0.75",
     "dist_acc":"0.05"
     "type":"StepDetection"
   }
}
```

Listing 3.2: Example of the data structure representing relative movements obtained from the step detection component (JSON notation).

As described in section 2.2.3, the Particle filter uses two steps to fuse data from different sensor systems: the *prediction step* and the *update step*. During the update step, absolute position information, provided by *LocationProvider* components, is processed, whereas the particles are moved based on the relative movements detected by *MotionProvider* components during the prediction step. After both steps position information will be distributed to the connected third-party applications. The resulting position information differs primarily in the presence respectively absence of information about the travelled distance and the heading. After processing relative movements during the prediction step, this information is always available, whereas it depends on the *LocationProvider* component, that delivered the new position information for the update step, whether information about distance and heading is available. To distinguish between these two cases, the position information distributed after the update step will be called *position update* and after the prediction step *position calculation*. If it is irrelevant whether the position information was distributed after the prediction step or update step it will be generally referred to as *position information* or *position*.

The *LocationCenter* gathers the information of all *positioning components* and coordinates the provision of position information requested by internal (e.g. *LocationCalculator* components) and external (e.g. third-party applications) components. The *Location-Service* component provides interfaces for external components and is responsible for directing requests from these components to the *LocationCenter* as well as granting access to systems and services of the operating system, such as sensors and communication capabilities. All requests for these services have to be done through the *LocationCenter* component. Figure 3.2 illustrates the architecture of the multimodal positioning service.

Third-party applications are able to request position information of a certain quality (respectively accuracy). The positioning service notifies registered third-party applications when new position information is available that meet these requirements (see Figure 3.5).

The more sensors and technologies are integrated in a smartphone, the more likely it is, that position information can be obtained in the majority of vicinities. However, not every device integrates every specific positioning technology. Standard commodity of actual smartphones is the availability of GPS, Bluetooth, Wi-Fi, acceleration and/or gyroscope sensors, a compass, and one or more cameras. Only a few smartphone models support further technologies like NFC or ZigBee. Users with a smartphone that does not support a specific technology would not be able to receive position information provided by this technology. For this reason, a mechanism was developed that enables the smartphones to exchange position information in a collaborative way amongst other nearby smartphones. Because most smartphones support the Bluetooth communications technology, the collaborative exchange uses it to share the position information. The smartphones connect automatically to other smartphones in the immediate vicinity, whose RSSI values are better than a defined threshold. Test, that were conducted during the research project, showed, that the remote device is within 4 meters if the RSSI

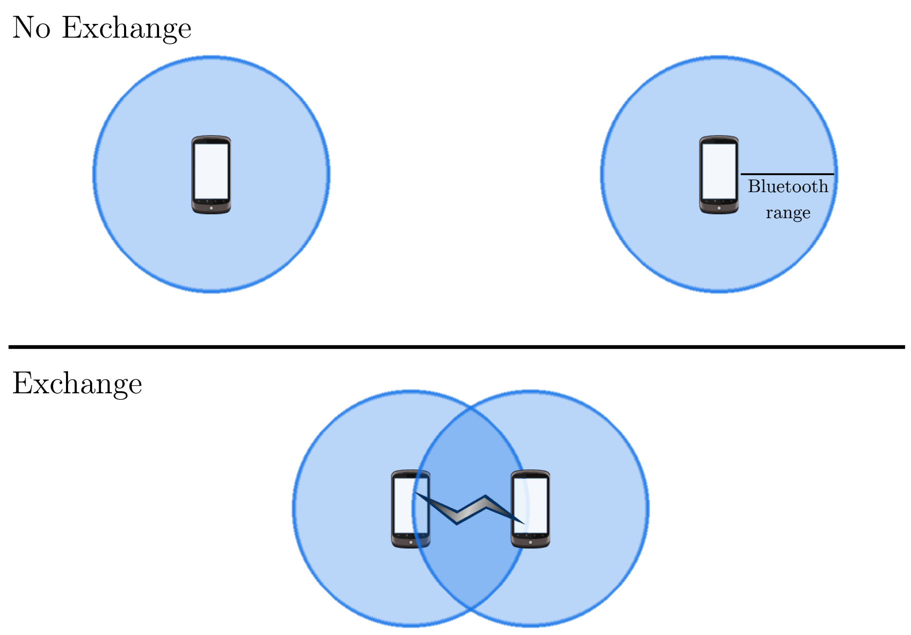


Figure 3.4.: Visualization of the collaborative exchange of position information via wireless communication.

value for this device is better than $-45\,\text{dB}$. Position information from other devices is processed if the accuracy of the position is better than the own currently determined position. With the collaborative exchange, smartphones that lack the availability of a certain technology can still benefit from position information provided by this technology as long as one mobile device in the vicinity integrates the specific positioning technology. Figure 3.4 visualizes this mechanism of collaborative position information exchange via a wireless communication: when two devices are within a specified range (e.g. based on the signal strength) the position information is exchanged amongst both devices. The user can enable or disable the collaborative sharing of position information (see Figure 3.5).

Sensor data fusion methods can be used to significantly improve the availability and quality of the position information. Position data from several, possibly complementary, sensors can be fused to obtain more accurate information. The positioning service is able to generically fuse position information provided by different kinds of sensors and positioning systems.

According to the results of the research project, the reliability of the position information provided by the *multimodal positioning service* has to be improved, especially in indoor environments. Possible options to achieve this will be discussed in chapter 5. Regarding the *multimodal positioning service*, the utilization of RFID beacons is promising to improve the accuracy and availability of position information in indoor environments.

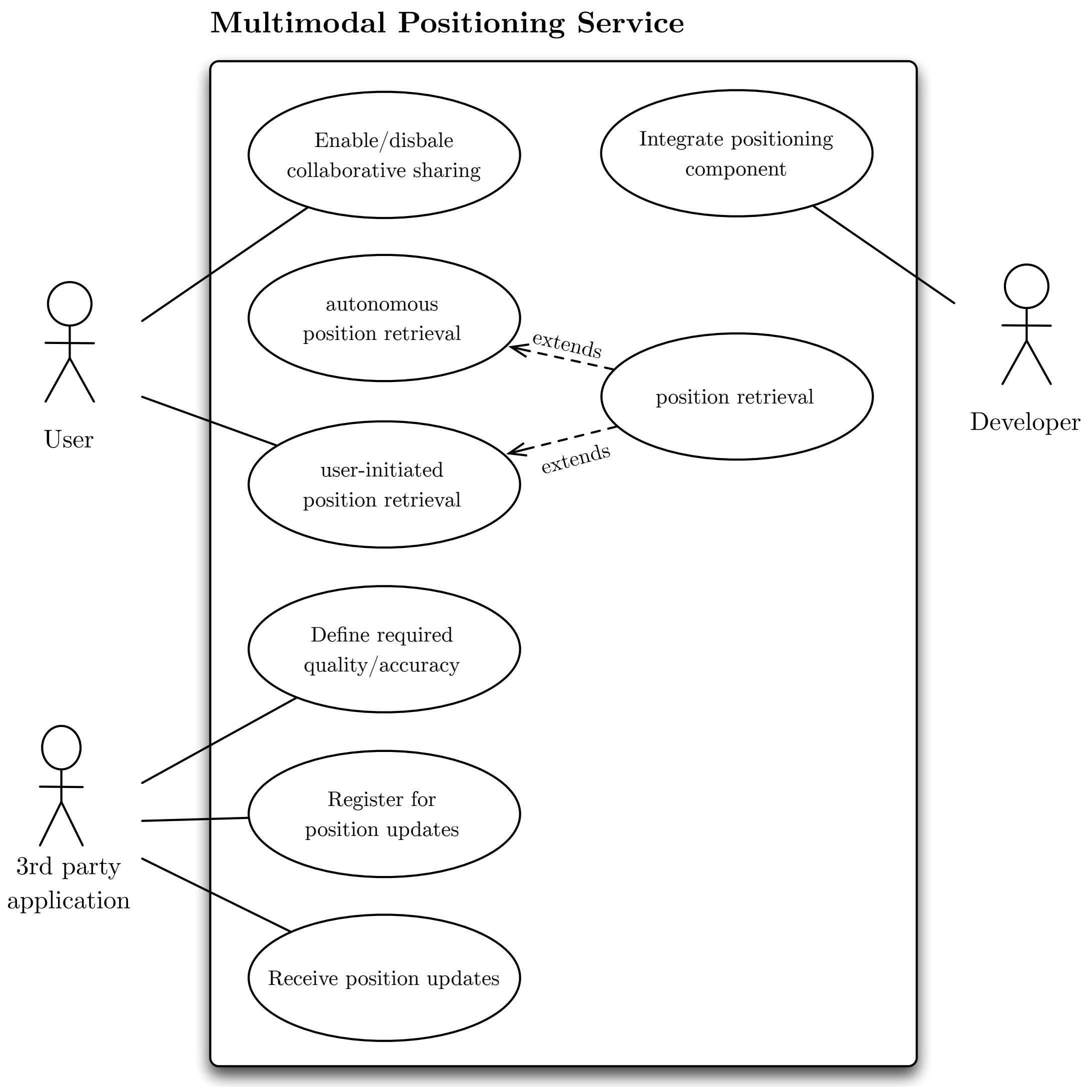

Multimodal Positioning Service
Enable/disbale collaborative sharing
Integrate positioning component
autonomous position retrieval
extends
position retrieval
extends
user-initiated position retrieval
Define required quality/accuracy
Register for position updates
Receive position updates
User
Developer
3rd party application

Figure 3.5.: Use cases of the *multimodal positioning service*.

RFID is a standardized and mature technology and its field of application is steadily growing, thus, the availability of the technology is granted and the prices of the hardware is decreasing.

3.3. Indoor Navigation Component

Like outdoor navigation, indoor navigation is based on positioning technologies and navigational data. In contrast to GPS for outdoor navigation, there is not a certain positioning technology that is suitable for most indoor environments. In fact, many of the research projects in the area of indoor navigation concentrate predominantly on the development of appropriate positioning strategies for the specific environment of the project. Furthermore, it is not distinguished between data used for positioning from the positioning system, such as GPS signals and Wi-Fi fingerprints, and data from the navigation component, such as information derived from MM and MA (e.g. [Col+11]).

This work distinguishes between the positioning component, the *multimodal positioning service* and the navigation component. Both components run as independent applications on the smartphone. The positioning component can serve other, even multiple, location dependent applications. The navigation component relies on position information provided by the positioning service and does not need to integrate a positioning component by itself. It will be designed as independent application that connects to the *multimodal positioning service* to realize the architecture conceived in the research project. In fact, a separation of positioning system and navigation component would be beneficial for several reasons:

- The positioning component can be further developed, integrating state-of-the-art positioning technologies and features, allowing an improvement of the quality of the information, without the need of customizing the applications on top of it, such as navigation systems, LBSs, and others.
- There is currently no centralized system to manage geospatial of buildings comparable to GIS for outdoor navigation. For this reason, every indoor navigation system for a single building (or at most the buildings of a limited territory) integrates its own positioning system that is adapted for the specific environment. This is a big drawback in comparison to outdoor navigation systems. Scholze [Sch09] investigated the possibilities of a network of servers providing information about buildings for an integrated indoor/outdoor positioning.

- A discrete positioning system with third-party applications on top, consuming the position information, will reduce the complexity of each individual component. Thus, the effort of implementing and maintaining the components can be reduced.

In the following, the features of a navigation component that is independent of a specific underlying positioning system will be identified. In this context, the communication capabilities between the navigation component and the positioning system are of particular interest: how can the navigation component receive position information from the positioning system and how can the navigation component contribute to improve the quality of the information provided by the positioning service. The use case diagram in Figure 3.6 visualizes the basic and enhanced features of the navigation component.

3.3.1. Basic Features and Requirements

To determine the users' current position is a fundamental part of navigation and guidance. Because the navigation component does not integrate a positioning system it has to be able to process position information from an independent, external [1] positioning system. In this case, this is the *multimodal positioning service* described in the previous section. The information has to be provided in a predefined data format.

To calculate a route and generate a list of navigational instructions, the user has to provide several parameters. This includes the definition of at least one waypoint that serves as destination. Adding further waypoints, that should be visited during the navigation, is optional. The *mode of travel* defines whether the user prefers a normal route or is dependent on a barrier-free route. In addition, the user should be able to define the type of the route. In indoor environments this could be, for example, the shortest or fastest route, the route with the least steps, or the most scenic route, leading along the exhibition areas of the building. To calculate shortest paths in the network, Dijkstra's algorithm is very well suitable for indoor environments. Although the $A*$ algorithm uses a heuristic to determine the edges that would most probably lead to the destination node and therefore might perform better than the original algorithm of Dijkstra, performance is not the main focus in small networks of buildings.

A representation of the navigational network of at least the building, in which the tests will be performed, must be provided in an appropriate form. Very common are databases that contain information about the nodes and links of the navigational network, together with their properties and additional useful information. Table 3.1 lists typical elements

[1]External in the sense of not being part of the navigation component.

Element	Property
Node	Node type (POI) Adjacents Coordinates
Link	Link type (hallway, stair, ...) Start/end node Geometry (height, width, length)

Table 3.1.: Typical properties of elements of a navigation network

of a navigational network and their properties. Karimi describes the data needed for indoor navigation and gives an example for an Entity-relationship model (ERM) that is suitable for indoor navigation [Kar11, pp. 65]. However, this model is too extensive for the purpose of a prototypic application.

After defining the necessary parameters, the user can start the navigation. This includes the route calculation based on the given parameters. A list of navigation instructions is generated and presented to the user in an appropriate form. Textual or verbal instructions or a map-based presentation are very common.

3.3.2. Enhanced Features

Enhanced features of the navigation component are not necessary for the task of navigation itself. These features rather contribute to a more intuitive operation or an improved quality of the service.

The presentation of the navigation instructions can be context-sensitive in the sense of adapting the form of presentation to the users' actual situation. Thus, for example, the map-based presentation can be discarded and a textual presentation shown instead, when the accuracy of the position information drops below a certain level of quality. In an indoor environment it does not make much sense to visualize a position with an accuracy of 10 to 20 meters on a floor plan. A textual presentation that includes references to landmarks of the immediate environment (e.g. „turn left in front of the fire door“) would be more suitable in this context. Showing a more detailed map or a photo of the destination area when the user approaches it is another example where a context-sensitive presentation would be beneficial.

Whereas *map-aiding* techniques are commonly used in outdoor navigation applications, it has not been utilized in indoor navigation applications very often. The geometry of the building, represented by the navigational network stored in the database, is used to eliminate impossible movements caused by faulty sensory data. Abdulrahim et al. [Abd+11] make assumptions about the shape of the building and the layout of rooms and corridors. A simplified geometry of buildings is used to improve the accuracy of the positioning based on MEMS. Buildings are assumed to have primarily turns of 90 degrees. By assuming this geometry, only 90°-turns have to be detected by the MEMS and can be directly mapped to an intersection of edges of the graph. The direction and angle of the detected turn indicates which edge the user will follow next. Since most of the navigation systems integrate a positioning system suitable for their specific environment, the information used for map-aiding is an integrated part of the positioning system as well. With the approach presented in this work, every application on top of the *multimodal positioning system* can include information that is necessary for the specific purpose of this particular application. With regard to this work, the navigation component will include geospatial information about the building that houses the offices of the INKA research group. The information derived from map-aiding will be reported to the underlying *multimodal positioning system*. An appropriate form and mechanism to report and process this information will be developed.

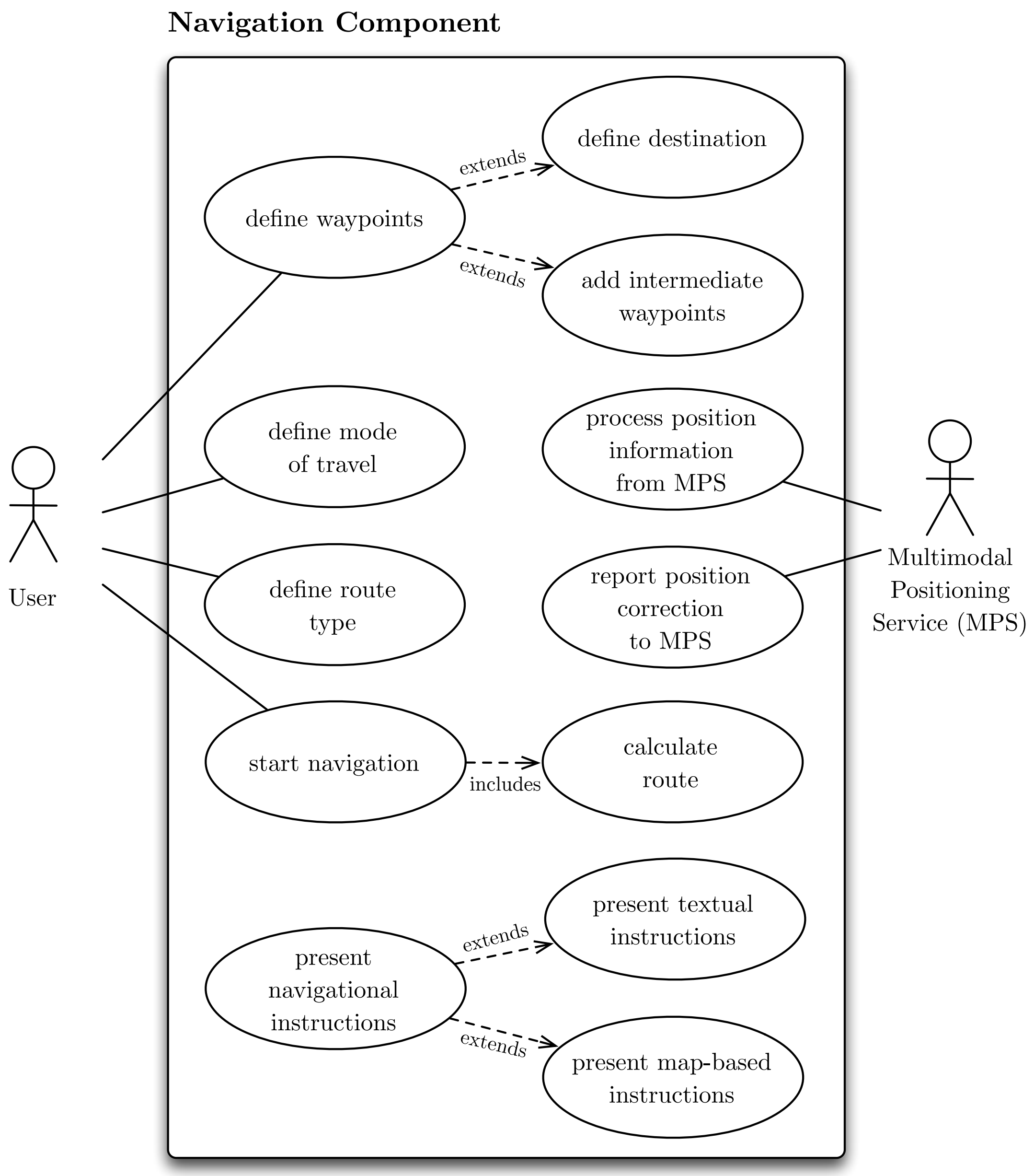

Navigation Component
define destination
define waypoints
extends
extends
add intermediate waypoints
define mode of travel
process position information from MPS
User
define route type
report position correction to MPS
Multimodal Positioning Service (MPS)
start navigation
includes
calculate route
present textual instructions
present navigational instructions
extends
extends
present map-based instructions

Figure 3.6.: Use cases of the navigation component.

4. Enhancements for the Multimodal Positioning Service

The general architecture of the *multimodal positioning service* was described in the last section. One outcome of the research was that the compasses of standard consumer smartphones, such as the *Google Nexus S* used in the research project, do not provide accurate information about the heading in indoor environments, producing an average error of up to 30 degrees. This is caused by disturbances of the earth' magnetic field due to metal structures of the building and sources of electromagnetic fields, such as computers and monitors, which are typically found in indoor environments. The error of the compass can be compensated in several ways. On the one hand, the data of another sensor can be used to stabilize the compass direction. On the other hand, the directions obtained from the compass sensor can be corrected by determining the direction between two or more known positions. On the basis of the direction calculated from the known positions, a correction value that can be added to the raw values of the compass can be computed.

The results of the research project revealed another shortcoming of the *multimodal positioning service*. For indoor environments only user-initiated *LocationProvider* components exist. The user has to trigger the position retrieval actively by capturing optical QR barcodes with the smartphone's camera or by placing it near to NFC tags. The shifting of the smartphone, that is necessary to scan the bar codes and tags, could lead to the false positive detection of steps and falsely determined directions. An autonomously working *LocationProvider* component that works well in indoor environments would be beneficial in many ways, reducing the error caused by unintentional shifting of the device and therefore improving the accuracy and the ease of use. RFID is a promising technology to build up an infrastructure that is easily installed and suitable for positioning. The RFID hardware of the *OpenBeacon* project is generally available and provides features necessary for positioning.

4.1. Compass Stabilization

The main issue that arose from the research project was the poor performance of the compass sensor in indoor environments. The results showed errors of 40 degrees in the test environment of the *Technologie und Gründerzentrum Spreeknie Berlin* (TGS), where the offices of the INKA research group are located. A reliable heading detection, which is an integral part of the step detection algorithm, is not possible with such haphazard values provided by the compass sensor.

Different approaches exist that are conceivable to overcome the problem of the error-prone compass sensor in indoor environments. One possibility is to stabilize the values of the compass sensor by using another sensor that is not susceptible to local changes of the earth's magnetic field. Acceleration sensors and gyroscopes are suitable to compensate the error of the compass sensor. Both types of sensors are available on Google's *Nexus S* smartphone. Acceleration sensors measure the acceleration on one or more axis, in actual smartphones typically three axis, to determine the current orientation (based on gravity) and movements along the different axis of the device. The acceleration sensor is used in the step detection algorithm to detect steps based on typical patterns of acceleration along the y- and z-axis of the device (see Figure 4.1). The gyroscope sensor measures the rate of rotation along the devices axis [Goob]. In this way, changes of the direction of the device can be tracked by integrating the values of the gyroscope sensor over time. Values of the compass sensor are only used to determine the initial heading. Changes of the direction are tracked by the gyroscope sensor and added to the initial heading with every time step. With this approach, the heading determination is not susceptible to local changes of the earth's magnetic field. However, the heading determination still relies on the direction that is initially determined by the compass sensor. Further approaches to overcome this problem will be discussed in subsection 5.1.3 and subsection 5.1.4.

4.2. OpenBeacon RFID Extension

To improve the reliability and availability of position information especially in indoor environments, the multimodal positioning service will be extended by external active RFID-based components. The *OpenBeacon* RFID technology will be utilized. The technology meets the requirements of positioning capabilities and due to the *open platform* approach the hardware is generally available. *OpenBeacon proximity tags* will be sparsely distributed in the indoor environment. These tags are able to identify

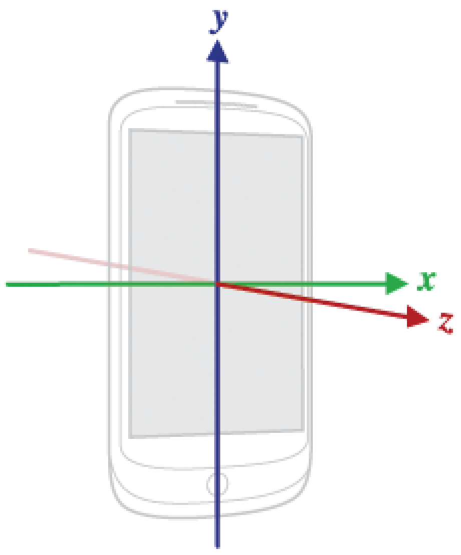


Figure 4.1.: Coordinate system used by Android's *SensorEvent* API [Goo12a]

other *OpenBeacon* RFID tags in the vicinity. The multimodal positioning service will connect to an *OpenBeacon USB2* tag which basically is similar to a proximity tag, but provides additional features, such as Bluetooth. Tag sightings, based on the proximity to nearby tags, are reported via Bluetooth by the *OpenBeacon USB2* tag. The multimodal positioning service has to be extended by a component, which is able to connect to and receive data from an *OpenBeacon USB2* tag and download information about the recognized tag from an online service. The distance between the mobile device (respectively the *OpenBeacon USB2* tag) and a proximity tag is based on the power level that was used to transmit a packet from the proximity tag to the *OpenBeacon USB2* tag. The *OpenBeacon* tags send packets with up to eight different power levels. The *OpenBeacon* tags support four distinct power levels natively. Another four power levels are provided by damping the antenna output by enabling a resistor. The number of packets received at a certain power level in relation to the number of packets received at another power level indicates the distance between the mobile device and the *OpenBeacon* tag. Furthermore, the characteristic ranges, at which a reception of packets is possible, can be determined for each power level.

5. Conceptual Design of the Indoor Navigation Application

Based on the given architecture of the *multimodal positioning service* and the requirements of indoor navigation component specified in the previous section the conceptual design of the indoor navigation component will be outlined in this section. Furthermore, several options that could improve the accuracy of the positioning will be discussed. These options incorporate the integration of additional positioning components in the *multimodal positioning system* as well as using additional information from the indoor navigation component for positioning purposes.

5.1. Indoor Navigation System

For the prototypic indoor navigation component, which will be developed in the context of this work, a representation of the test buildings' navigational network must be provided as well as a shortest-path algorithm, producing a list of navigational instructions. Furthermore, a concept of a context-sensitive presentation of these navigational instructions has to be compiled and, at least in parts, implemented. The main contribution of this work will be the development of strategies to use additional information of the independent navigation component to improve the accuracy of the underlying *multimodal positioning service.* The concepts are outlined and the parts of it, that actually will be implemented, are stated in the next sections.

With regard to the concept of the indoor navigation application, as described in section 3.3, the prototype will provide a limited set of features to the user. These features include:

- Determine the user's current position.
- Specify a destination.
- Specify a cost function as basis for the route calculation.

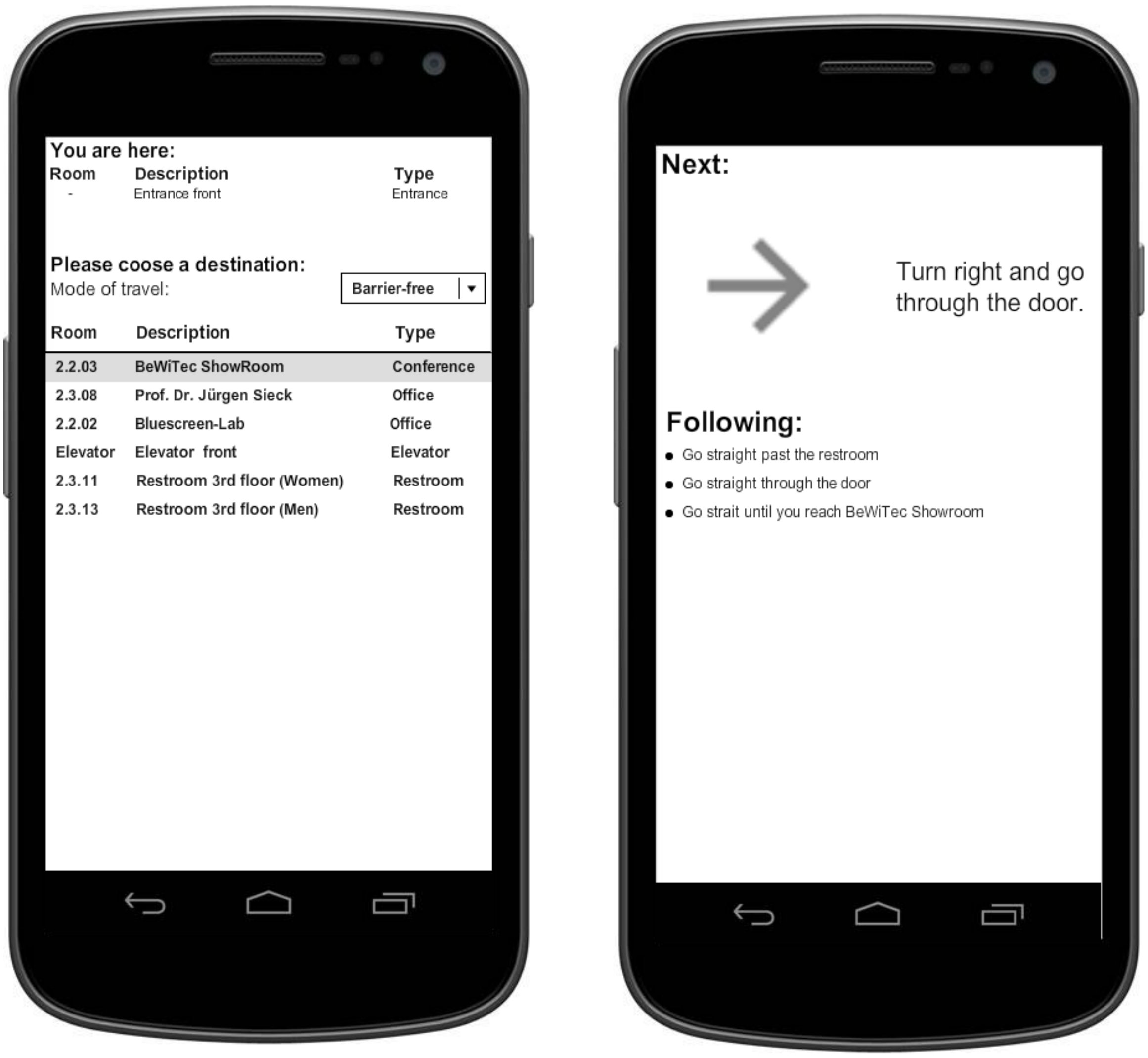
- Start the navigation.
- Provide a list of textual navigation instructions.
- Provide guidance through turn-by-turn instructions based on the user's current position and the computed path.

5.1.1. Navigational Network

Instead of road and sidewalk networks in outdoor navigation systems, indoor navigation systems cover hallway networks of buildings. A general approach for modeling such networks was given in section 2.3.1. According to Lorenz and Ohlbach [LO06, p. 102], a graph representation should be preferred because shortest-path algorithms work well with graphs. A hierarchical graph representation, as it is proposed by the authors, would be suitable for complex buildings and larger networks. In the context of this work, a simple graph representation will be sufficient. Elements of this graph will be points, link segments, nodes, and POIs. Points represent a position, defined by coordinates. A link segment connects two nodes. Link segments represent hallways, stairs, elevators, and so forth and often have a straightforward geometry (straight lines) [Kar11, p. 62]. Link segments are represented by a set of points. A node is specified by a point and represent a *decision point*. This can be a multi-way decision point where the user can change hallway segments, e.g. a corner or a junction, and terminal decision point with no further link segment that follows. POIs mark points with semantic information, such as offices, restrooms, and others. They are mapped to link segments.

Different cost functions can be applied to link segments, according to its type. This is important to calculate the shortest path based on different user requirements. For example, for ordinary users, a link segment representing a stair will have a cost value that is slightly higher in comparison to a link segment representing a hallway. This does not apply for wheelchair riders. They cannot use stairs and this fact has to be taken into account by the cost function.

Dijkstra's shortest-path algorithm will be used to calculate a route from a start node to a destination node. A list of navigation instructions is generated from the results of the algorithm.



You are here:

Room	Description	Type
-	Entrance front	Entrance

Please coose a destination:

Mode of travel: Barrier-free ▾

Room	Description	Type
2.2.03	BeWiTec ShowRoom	Conference
2.3.08	Prof. Dr. Jürgen Sieck	Office
2.2.02	Bluescreen-Lab	Office
Elevator	Elevator front	Elevator
2.3.11	Restroom 3rd floor (Women)	Restroom
2.3.13	Restroom 3rd floor (Men)	Restroom

Next:

Turn right and go through the door.

Following:

- Go straight past the restroom
- Go straight through the door
- Go strait until you reach BeWiTec Showroom

Figure 5.1.: Mockups showing the user interface of the indoor navigation application.

5.1.2. Presentation of Navigational Instructions

The presentation of navigation instructions is kept very simple for the prototypic implementation. The user is able to choose a destination from a list of POIs. The cost function that is used by Dijkstra's algorithm is chosen via a drop-down menu, providing the available possibilities (see Figure 5.1 left screen). Textual instructions that indicate visual landmarks, such as intersections, stairs, elevators, and doors will be presented along with direction indicators at decision points. Subsequent instructions are presented in a list (see Figure 5.1 right screen).

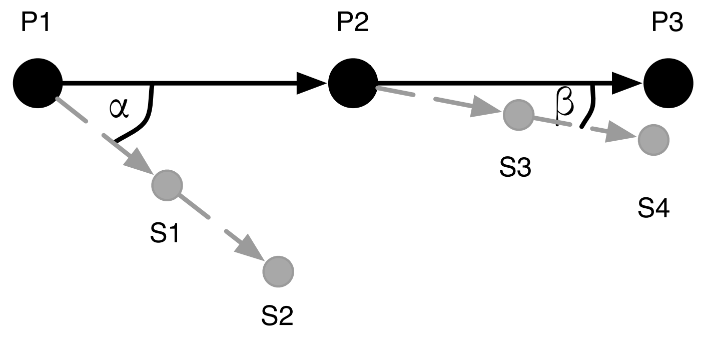


Figure 5.2.: Visualization of the compass correction concept.

5.1.3. Heading Correction

In section 4.1 an approach to stabilize the values of the compass sensor was described. Due to the fact, that this approach uses a compass value as initial heading, the results will be affected by the error of the initial compass value as well. Particularly in indoor environments where the earth's magnetic field is influenced by different sources of interferences, such as metallic structures of the building and electronic equipment, this error can be significant.

To overcome this problem, topologic information provided by the position system can be used. The heading between two consecutively detected positions $P1$ and $P2$ is calculated and compared with the heading provided by *step and heading detection algorithm* that uses the (stabilized) compass sensor. If the difference α between the two directions exceeds a specified threshold, a correction of $(-\alpha)$ is calculated and applied to every subsequent compass value. After the first correction was calculated and applied to the compass values, the heading provided by the *step and heading detection algorithm* should be closer to the real heading between the next consecutive positions. Figure 5.2 shows the process of the compass correction. The black circles $P1$, $P2$, and $P3$ represent consecutive positions detected by the positioning system. The gray circles $S1$, $S2$, $S3$, and $S4$ represent the positions after each step, that was detected by the *step and heading detection algorithm*. The angles α and β represent the difference between the heading calculated by the *step and heading detection algorithm* and the heading calculated between two consecutively detected positions.

5.1.4. Map-Aiding

The indoor navigation system includes navigational networks which provide topological and geographical information of the covered buildings. On the one hand, the position obtained from the positioning system can be mapped to a position on the graph of the navigational network. This process is called *map-matching* and is primarily used to visualize the user‘s position. On the other hand, after the user‘s position is mapped to an edge of the graph, the user will likely navigate along this edge and its adjacents. This relation can be used to derive further position information from the navigational network. This process is called *map-aiding*.

The *map-aiding* algorithm requires, that the user’s position is clearly mapped to an edge of the navigational network during the *map-matching* process. Furthermore, the direction of travel (heading) must be determined. This can be achieved by utilizing the information provided by the positioning system. The direction is directly available if a detected step was the basis for the position information provided by the *multimodal positioning service* (position calculation). If a *LocationProvider* component provided new position information, the direction might not be directly available. However, in this case the heading can be calculated from two consecutive position updates.

Contrary to the assumption of 90°-turns in buildings in Abdulrahim et al. [Abd+11], no such assumptions will be made in the approach presented in this work. The indoor navigation component tracks changes of the heading by using the gyroscope sensor. The *map-aiding* algorithm identifies the most likely edge in the direction of the detected heading. Movements, that are reported after the most likely edge was identified, will be mapped to this edge. The position derived from this mechanism is reported to the *multimodal positioning service* and processed by the Particle filter. In this way, the information derived from the navigational network is used to improve the accuracy and reliability of the underlying positioning system. Movements detected by the *multimodal positioning system*, which does not include any map information by itself, will correlate to movements that are feasible according to the geometry of the building.

5.2. Architecture of the Indoor Navigation Application

Based on the concepts described in the last sections, a basic architecture has been developed (see Figure 5.3). The *controller* component communicates with the *multimodal positioning service* via a defined interface. It also handles the interaction of the user interface and provides access to the model classes. The *NavigationHandler* component processes the position information received from the *multimodal positioning service*, calculates paths, and implements the *map-matching*/*map-aiding* algorithm. The results

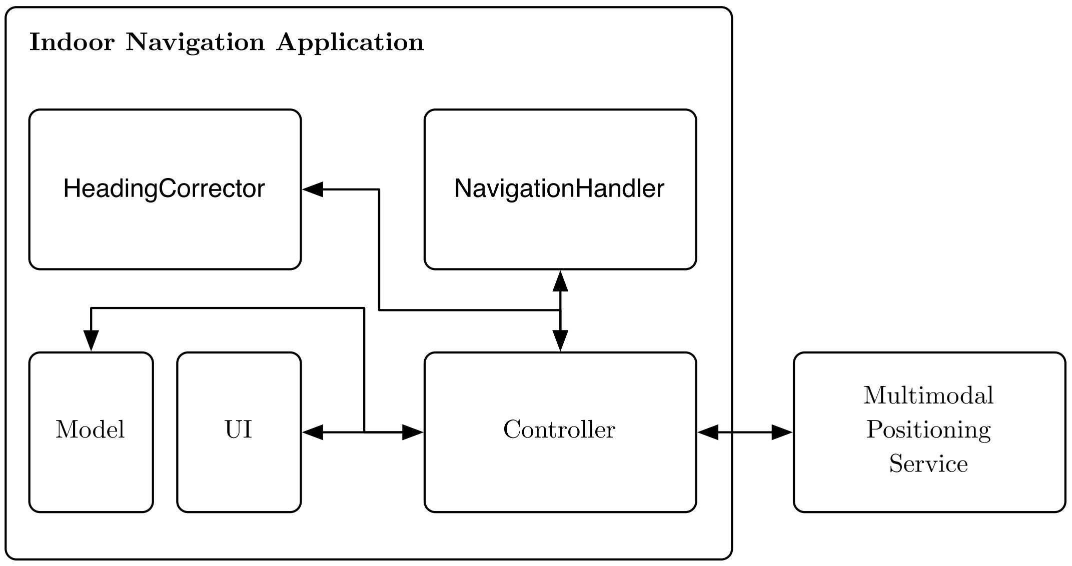


Figure 5.3.: Architecture of the indoor navigation application.

of the algorithm are reported to the *Controller*. The *HeadingCorrector* component processes the received position information as well, calculates corrections for the heading, and reports these to the *Controller*.

6. Prototypic Implementation

This chapter covers the implementation of enhancements of the *multimodal positioning service* and the prototype of the indoor navigation application. The focus lies on the presentation of noteworthy problems and tasks, that occurred during the implementation process, and the solutions found for these problems.

First of all, the development environment and the mobile platform, for which the indoor navigation application is developed, will be described. After that, specific details of the implementation process will be covered in more detail.

6.1. Development Environment

As stated in section 3.1, the target platform of the prototype is a current smartphone. The *multimodal positioning service* and the indoor navigation application are utilizing many different sensors and positioning systems. These are integrated into the smartphone or external systems, which are connected to the mobile device via a standard communication technology. The platform must provide convenient interfaces to access the sensors and to connect to the external components.

The *multimodal positioning service* was developed on Google's Android platform. smartphones that run the Android operating system (OS) are widely spread, claiming a market share of 68 % at the time of this work [Pep]. Google provides a software development kit (SDK) that allows to access services and functionalities provided by the OS in a convenient way. Moreover, Google provides additional tools, such as a SDK manager and a debugger, for the Eclipse *integrated development environment (IDE)* to support developers. The version of the current SDK at the time of this work is Android 4.1. Android devices running newer versions of the *SDK* are able to execute applications that were developed for previous versions of the Android SDK. The *multimodal positioning service* and the navigation component are developed for the version of Android 2.3.3, because this version provides access to all necessary features, such as sensor, NFC, and Bluetooth support. Moreover, a share of more than 75 % of all Android devices are compatible to this version [Gooa].

Applications for the Android platform are written in the Java programming language. Google uses a special virtual machine (VM), the Dalvik-VM, that executes the Java byte-code on Android devices.

The prototype will be tested on Google's smartphone *Nexus S*. However, the prototype should be executable on comparable devices that support the used sensors (GPS, NFC, acceleration sensor, gyroscope, compass, and camera) and communication technologies (Wi-Fi and Bluetooth). To be able to capture QR barcodes, the *multimodal positioning service* utilizes the third-party application *Barcode Scanner* of the developer *ZXing*, which can be downloaded for free (at the time this work was written) from Google's *Play Store*. The scanning of QR barcodes is not possible if this application is not available on the smartphone.

6.2. Software Design Patterns

The architecture of the *multimodal positioning service* and the indoor navigation component adopt common software design patterns which have to be considered for the implementation of the prototype. In addition, the Android platform specifies rules and mechanisms of how different components communicate with each other.

The *multimodal positioning service* is realized as *Android Remote Service*. Applications of this type generally run in the background and perform recurring tasks that do not need any user interaction. Due to this fact, the *multimodal positioning service* provides no user interface except for configuration tasks. The architecture of the service is highly component-based to facilitate the integration of additional *LocationProvider*, *MotionProvider*, and *LocationCalculator* components (cp. Figure 3.2). Other applications can connect to the service via an interface that is based on the Android Interface Definition Language (AIDL). Listing 6.1 shows the definition of the interface to which the *multimodal positioning service* has to conform to. The Android SDK generates the actual interface during the *build process*. The specification of the type of the parameters is very important. The prefix *in* defines, that the Java object represented by this parameter is only used within the service and will not be passed back to the original application. The applications, that want to connect to the service, can create a so called *stub* (proxy) of the service and use the methods defined in the interface. The implementation of the methods in the *multimodal positioning service* defined in the interface can be found in the appendix (Appendix, Listing A.1).

```
package de.bittins.bjoern.locationService;

import de.bittins.bjoern.locationService.model.Location;
import de.bittins.bjoern.locationService.model.LocationRequest;
import de.bittins.bjoern.locationService.model.Correction;

interface ILocationService {
    de.bittins.bjoern.locationService.model.Location getLocation();
    de.bittins.bjoern.locationService.model.LocationRequest ↩
        getLocationRequest();
    void reportCorrection(in de.bittins.bjoern.locationService.model.↩
        Correction correction);
    void reportLocation(in de.bittins.bjoern.locationService.model.↩
        Location location);
}
```

Listing 6.1: AIDL-interface of the *multimodal position service* (LocationService).

The indoor navigation component conforms to Android's model-view-controller (MVC) pattern. The user interface and the resources are defined in XML. Special classes, *Activities*, handle the user interaction and the provision of required data from the model classes to the user interface. Further classes, that implement the business logic of a particular scope, have been implemented to support the exchangeability of specific components.

6.3. Implementation Challenges

The following sections will highlight selected challenges along with their solution. The components of the *multimodal positioning service* and the indoor navigation application, which were implemented to improve the accuracy and reliability of the positioning, are the main focus of this section.

6.3.1. Compass Stabilization

The stabilization of the compass values is implemented for the *step and heading detection component*, a *MotionProvider* component (cp. Figure 3.2). The component identifies the steps the user takes by analyzing the acceleration sensor's y- and z-values. In the original implementation of this algorithm, which was done in the context of the research project prior to this work, compass values were recorded as soon as the first mark of the pattern was detected. If the pattern could not be completed within a specific time frame, which was determined during the research project, the recorded compass values were discarded. If the pattern was successfully completed and a step detected[1], an average value for the heading of this step was calculated from the compass values that have been recorded. Equation 6.1 shows, how the average heading was originally calculated from N compass values that were recorded from the first peak of the step-detection pattern until a step was successfully detected.

$$a = atan(\frac{\sum_{i=1}^{N} sin(c_i)}{\sum_{i=1}^{N} cos(c_i)}) \tag{6.1}$$

However, due to the fluctuation of the compass values in indoor environments, the calculated heading changed unpredictably even on a straight path. To overcome this problem, the gyroscope sensor of the smartphone was utilized to stabilize the results of the heading determination. After the *step and heading detection component* is instantiated, the compass sensor provides a single value for the current heading. Changes of the heading are calculated by integrating the angular velocities that are reported by the gyroscope sensor in a specific time frame. In this case, the values of the last 250 ms are integrated and added to a member variable, *gyroSum*, that stores previous results of the integration to cover a greater time period. Because the raw values of the gyroscope are subject to noise they must be filtered to avoid a drift of the summed up values. A threshold of $0.025°$ is used to discard values close to zero, which occur even if the device is not rotated. In this way, the heading determination could be improved in comparison to the values provided by the compass sensor, especially indoors in the vicinity of sources of electromagnetic interferences.

6.3.2. Haversine formula

To represent a position, the *multimodal positioning service* uses longitude and latitude coordinates based on the World Geodetic System 84 (WGS 84), which is also used by GPS. To calculate distances and headings on a sphere, the simplified model of the earth, the *Haversine formula* and other formulas related to navigation are utilized.

The *Haversine formula* is used to calculate great-circle distances (air-line) between two points on the surface of a sphere, given their longitude and latitude coordinates (see Figure 6.1). Equation 6.2 shows how to calculate the distance between two points $P1$ and $P2$, with R being the earth's radius [Movb].

[1]The typical pattern of detected steps is illustrated in Figure A.1

Figure 6.1.: Distance between two points on a sphere. [Mova]

$$
\begin{aligned}
a &= \sin^2(\frac{\Delta lat}{2}) + \cos(lat_{P1})\cos(lat_{P2})\sin^2(\frac{\Delta lon}{P2}) \\
c &= 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}) \\
d &= R \cdot c
\end{aligned}
\tag{6.2}
$$

If only the coordinates of the origin x, the heading θ and the distance d is given, and the destination y is unknown, it can be calculated using Equation 6.3 [Movb].

$$
\begin{aligned}
lat_y &= \arcsin(\sin(lat_x) \cdot \cos(d/R) + \cos(lat_x) \cdot \sin(\frac{d}{R}) \cdot \cos(\theta)) \\
lon_y &= lon_x + atan2(\sin(\theta) \cdot \sin(\frac{d}{R}) \cdot \cos(lat_x), \cos(\frac{d}{R}) - \sin(lat_x) \cdot \sin(lat_y))
\end{aligned}
\tag{6.3}
$$

Implementations of the formulas can be found in section A.1.

6.3.3. Heading Correction

The previously described approach to stabilize the heading determination is not capable of eliminating the error introduced by the initial heading provided by the compass sensor (see subsection 6.3.1). This error can be significant if sources of electromagnetic interferences are nearby.

With the current, error-prone sensors integrated in current smartphones, this error cannot be eliminated without external systems or additional information. However, if the *multimodal positioning service* is able to provide position information from two

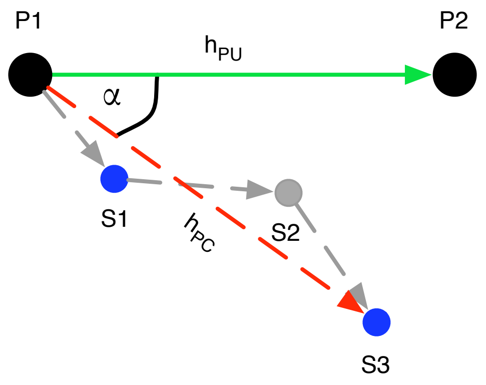


Figure 6.2.: Visualization of the implemented compass correction process.

consecutive *position updates*[2], the heading between these two points can be calculated and compared to the heading determined by the (stabilized) compass.

Differing from the concept of the compass correction as described in chapter subsection 5.1.3, it is not practical to use the heading information provided by the *position calculations*, which are received between the consecutive *position updates*, directly. In fact, the first *position calculation* $S1$ after a *position update* $P1$ and the last *position calculation* $S3$ before the next *position update* $P2$ are stored (blue points in Figure 6.2) to calculate the average heading h_{PC} of the steps. The difference α between the headings h_{PU}, which represents the heading between subsequent *position updates* $P1$ and $P2$, and h_{PC} is calculated and reported as correction to the *multimodal positioning service*. Due to the fact, that the correction is calculated after every two consecutive *location updates*, it adjusts quickly on changes.

6.3.4. Navigation Network and Dijkstra's Algorithm

The navigation network of the test environment, which consists of the 2^{nd} and 3^{rd} floor of the TGS *Building 2*, is modeled with the help of objects of the class *NavigationVertex*. Each *NavigationVertex* object describes a node of the navigation network, which represents a *decision point* or one or more POIs. A *NavigationVertex* object stores all adjacent *NavigationVertex* objects along with a *LinkType* in an adjacency list, defining the links to other nodes of the network. Furthermore, it stores information required by Dijkstra's algorithm, such as the (weighted) distance and the predecessor node.

[2]For a definition of *position updates* and *position calculations*, see chapter 3.2.

```java
public List<NavigationVertex> computeDijkstraShortestPathsWithCostFuntion(
        NavigationVertex source, NavigationVertex destination, CostFunction cf) {
    this.resetDijkstraInfoOfAllVertices();
    PriorityQueue<NavigationVertex> vertexQueue = new PriorityQueue<NavigationVertex>();
    source.setDistance(0.0);
    source.setWeightedDistance(0.0);
    vertexQueue.add(source);
    // compute shortest path to all vertices from source vertex
    while (!vertexQueue.isEmpty()) {
        NavigationVertex s = vertexQueue.poll();
        // Visit each link that is adjacent to v
        for (Pair<NavigationVertex, LinkType> a : s.getAllAdjacentVerticesWithLinkType()){
            double distance = v.distanceToVertex(a.first);
            double weight = NavigationCostCalculator.calculateCostForDistanceViaLinkTypeWithCostFunction(distance, a.second, cf);
            double weightedDistanceThroughV = v.getWeightedDistance() + weight;
            if (weightedDistanceThroughV < a.first.getWeightedDistance()) {
                // if new distance is lower than current distance, then
                // remove node from priority queue to set new distance
                // and new predecessor
                vertexQueue.remove(a.first);
                a.first.setDistance(v.getDistance() + distance);
                a.first.setWeightedDistance(weightedDistanceThroughV);
                a.first.setPrevious(v);
                // sorted reinsert node to priority queue
                vertexQueue.add(a.first);
            }
          }
    }
    return this.computeShortestPathToDestination(destination);
}
```

Listing 6.2: Implementation of Dijkstra's shortest path algorithm.

The implementation of Dijkstra's shortest path algorithm is given in Listing 6.2. It follows the implementation presented in [Alg] and is modified to conform to the data structures used in the indoor navigation application. Moreover, it is extended to support different cost functions that will influence the calculation of the shortest path according to predefined constraints, such as the avoidance of stairs and escalators to provide a barrier-free path. A detailed explanation of the algorithm is given below.

- In line 2, the information stored for each *NavigationVertex* object, that is required by Dijkstra's algorithm, is reset before a new shortest path is calculated. This includes the predecessor of the node, which is set to *null*, and the unweighted and weighted distance to the source node, which both are set to *infinity*. Distinguishing between weighted and unweighted distance is important to be able to compute the shortest path for a given cost function, based on the weighted distance, and to be able to present the user the real, unweighted distance of the path.
- The instruction of line 3–6 initialize Dijkstra's algorithm. The priority queue, which provides the functionality of a sorted insertion of *NavigationVertex* objects, according to their weighted distance, is initialized and the source node is inserted. The distance and weighted distance of the source node is set from *infinity* to 0. No predecessor is set for the source node.
- With each cycle of the *while*-loop (line 10), the node v with the currently shortest distance to the source node is drawn from the priority queue (line 10). All adjacents a of v are visited (line 12).
- In line 13–15, the unweighted and weighted distances are computed, based on the distance from the source to v and v to the adjacent node a. Additionally, the cost function is applied to the distance for the weighted distance to calculate the weight of the link between v and a (line 14).
- If the just calculated weighted distance is less than the weighted distance that is stored for the adjacent node a, then the distances will be updated and the predecessor of a will be set to v. The node a has to be removed from and reinserted to the priority queue to ensure the correct order of the priority queue (lines 20 and 25).
- After the shortest path from the source node is calculated for all nodes of the navigation network, the path can be computed by following the predecessor nodes from the destination node to the source node and inverting this path. This is done in the method *computeShortestPathToDestination* (line 29).

Navigation instructions can be generated from the list of *NavigationVertices* computed by Dijkstra's algorithm. If the user leaves this path, a new path has to be computed, given the user's current position as new source node for the algorithm.

6.3.5. Map-Matching/Map-Aiding Algorithm

The main contribution of this work is a *map-matching/map-aiding* algorithm, implemented in the indoor navigation application. This algorithm is intended to provide additional position information to improve the accuracy and reliability of the position information of the underlying, independent *multimodal positioning service*. This is achieved by utilizing information from the navigation network of the indoor navigation application to infer a valid position on the network.

First of all, the positions received from the *multimodal positioning service* have to be mapped to the navigational network. Due to the fact, that the user will follow links between the nodes of the navigation network, the position on such a link and the direction in which the user heads can be determined. The position on a link between two *NavigationVertex* objects is represented by an object of the class *LinkDistanceTuple*. A *LinkDistanceTupel* holds information about the start and the destination node of the link, the distance to the link ($d_{link}(P_x, P_y)$ in Figure 6.3) and the distance on the link, measured from the start node (d_{P_x} in Figure 6.3). The process of mapping a received position P_R to the most likely link between two nodes of the navigation network is visualized in Figure 6.3. The first step of this process is to determine the distance of the received position P_R to nearby nodes ($P1$, $P2$, and $P3$). If the distance to one of the nodes is within a defined threshold, in the case of this work 0.5 m, the received position is mapped directly to the position of the node. If the distance to all nearby nodes is greater than the threshold, then the distance to the links between the nearby nodes and its adjacents is calculated. The distance to a potential link must be less than 3.0 m. In the case shown in Figure 6.3, two potential links are nearby: the link between nodes P_1 and P_2 and the link between P_2 and P_3. The received position P_R is mapped to the link with the shortest distance. In this case, this is the link between the nodes P_1 and P_2, because the distance $d_{link(P_1,P_2)}$ is shorter than the distance $d_{link(P_2,P_3)}$.

Depending on the degree of knowledge about the current link and the type of the position information (*location update* or *location calculation*) provided by the *multimodal positioning service*, the received position is mapped to the navigation network. Possible degrees of knowledge are (cp. Figure 6.4):

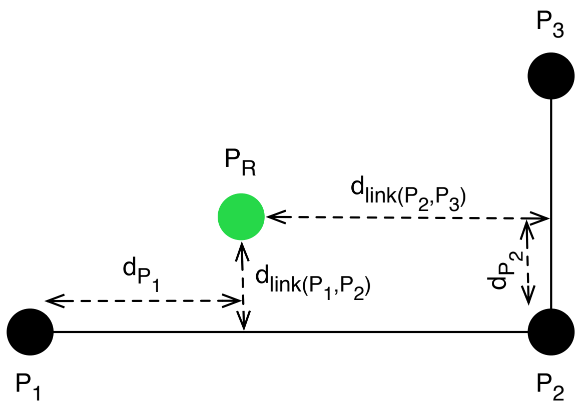


Figure 6.3.: Process of mapping a received position to the most likely link between two nodes of the navigation network.

- No current link could be determined. Neither the start nor the destination node and the direction are known.
- Only the start node is known. This happens if a *position update* was received and directly mapped to a node of the navigation network.
- The start and destination nodes are known, but the direction cannot be determined. This happens if a position could be matched to a link, but no reliable heading information is available to determine the direction in which the user follows the link.
- All information about the current link is available. This includes the start and destination nodes of the link and the direction. The direction property of *LinkDistanceTupel* objects is not given in degrees, but as a boolean value, indicating whether the direction from the start node to the destination node was determined reliably. If the direction changes by 180°, start and destination node have to be switched.

The sequence diagram shown in Figure 6.4 describes the process of determining the degree of knowledge about the current link. If a new position is received from the *multimodal positioning service*, but the link the user is currently following could not be determined yet, then the currently received position is initially mapped to the navigation network. The sequence diagrams of the initial mapping process and the other processes of mapping a received position can be found in Figure A.3, Figure A.4, Figure A.5, and Figure A.6 in the appendix.

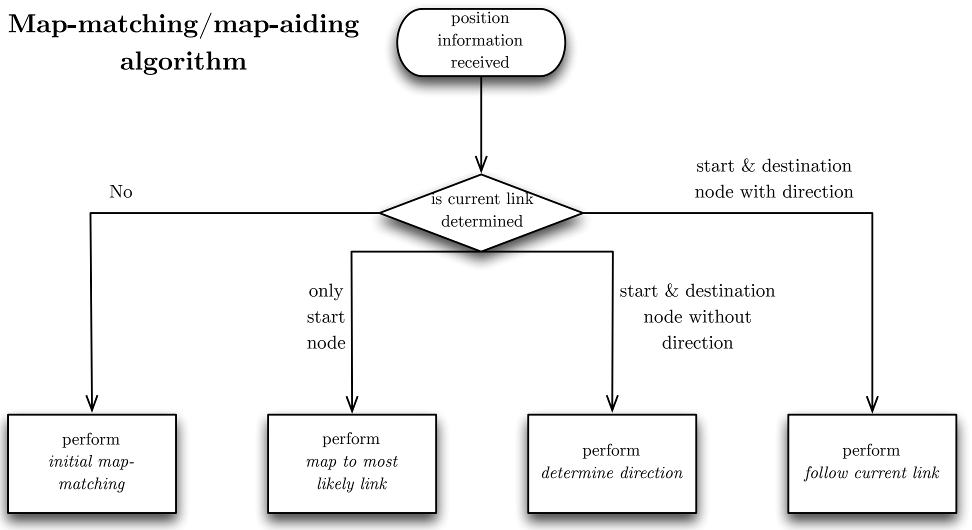

Map-matching/map-aiding algorithm
position information received
is current link determined
No
only start node
start & destination node without direction
start & destination node with direction
perform initial map-matching
perform map to most likely link
perform determine direction
perform follow current link

Figure 6.4.: Sequence diagram of the *map-matching/map-aiding* algorithm, part 1: determining degree of knowledge about the current link.

The most interesting case of the *map-matching/map-aiding* algorithm is this of having complete knowledge about the current link. The start and destination nodes are known and it is ensured, that the user is following the link in the direction from the start node to the destination node. In this case, if a new *location calculation* (a position derived from a detected step) is received, the possibly error-prone information about the heading can be ignored. Only the distance property of the *location calculation* is used to move the position along the current edge. This scenario is described in Figure 6.5.

If the *map-matching/map-aiding* algorithm was able to determine the current link completely, then the mapped position is reported to the *multimodal positioning system.* In this way, the particles of the Particle filter are more likely moved to a position that conforms to a valid position derived from the navigational network. Unlikely movements, for example movements through walls and obstacles, that could not have been detected without the knowledge of the indoor navigation application, are downgraded by passing the map-matched positions to the *multimodal positioning service.* Therefore, these unlikely movements will have a lower impact on the positioning results.

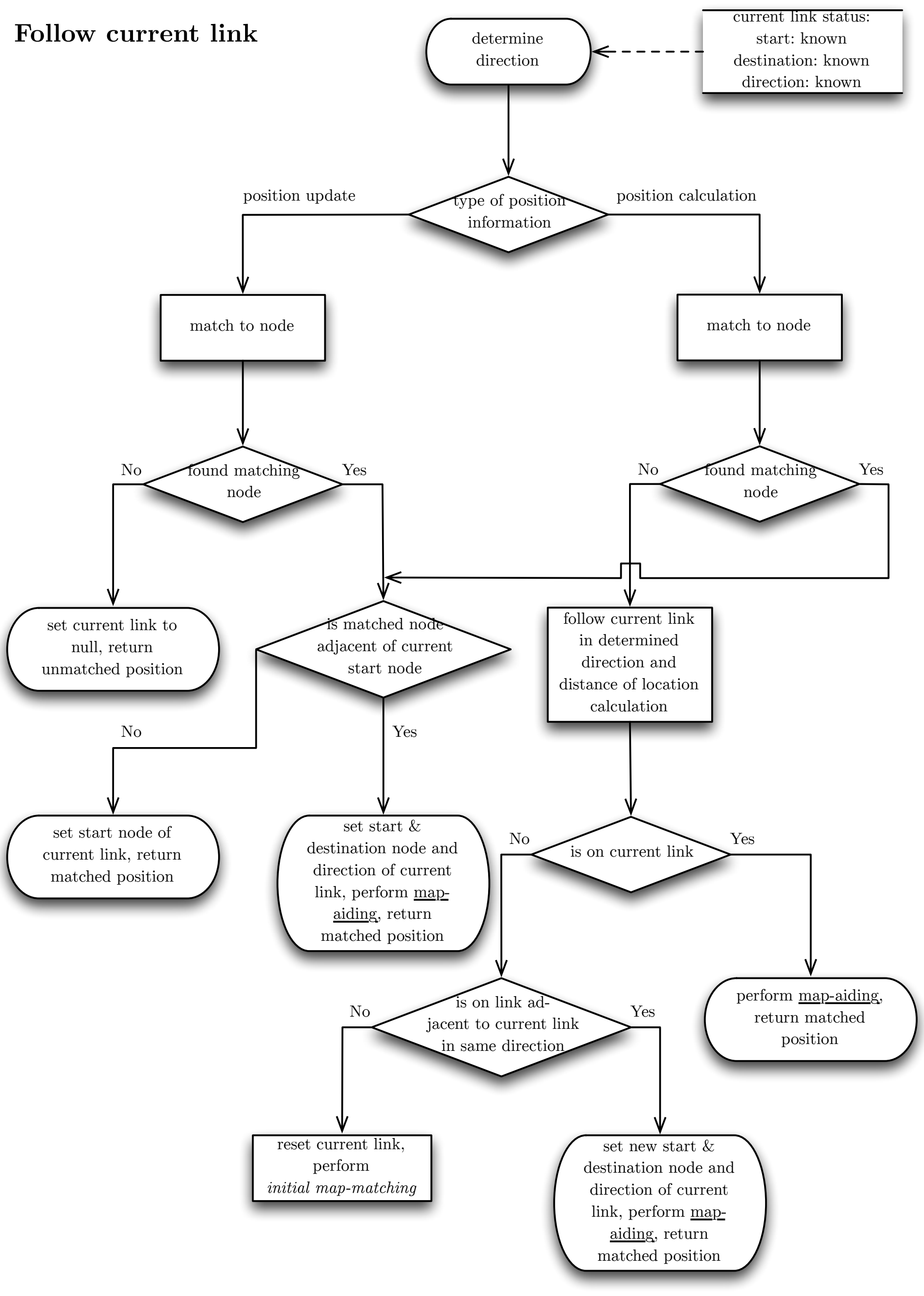

Follow current link
determine direction
current link status: start: known destination: known direction: known
type of position information
position update
position calculation
match to node
match to node
found matching node
No
Yes
found matching node
No
Yes
set current link to null, return unmatched position
is matched node adjacent of current start node
No
Yes
follow current link in determined direction and distance of location calculation
set start node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position
is on current link
No
Yes
perform map-aiding, return matched position
is on link adjacent to current link in same direction
No
Yes
reset current link, perform initial map-matching
set new start & destination node and direction of current link, perform map-aiding, return matched position

Figure 6.5.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 5: following the current link.

7. Tests and Evaluation

In this chapter, the tests that were conducted to analyze the performance of the indoor navigation application and the different approaches to improve the accuracy of the positioning, will be described. The results of each approach will be evaluated individually as well as combinations of multiple approaches to identify the best solution. The performance will be compared to the results of other indoor navigation systems.

7.1. Test Methodology

The test took place in the office complex of the *Technologie und Gründerzentrum Spreeknie Berlin* (TGS), where the offices of the INKA research group of the HTW Berlin are located. The test environment covers the second and third floor of TGS building 2. The performance of the positioning was tested for each floor individually, due to the fact that the vertical links (stairs, elevators) between the floors are not considered in the *map-matching* process. Figure 7.1 shows the floor plan of the third floor with its navigation network.

The *multimodal positioning service* was configured to use the *LocationProvider* component based on the *OpenBeacon* RFID technology and the *MotionProvider* component based on the *step and heading detection algorithm* only. The performance of the other *LocationProvider* components (GPS, QRn and NFC) was already presented in [Bit12b] and

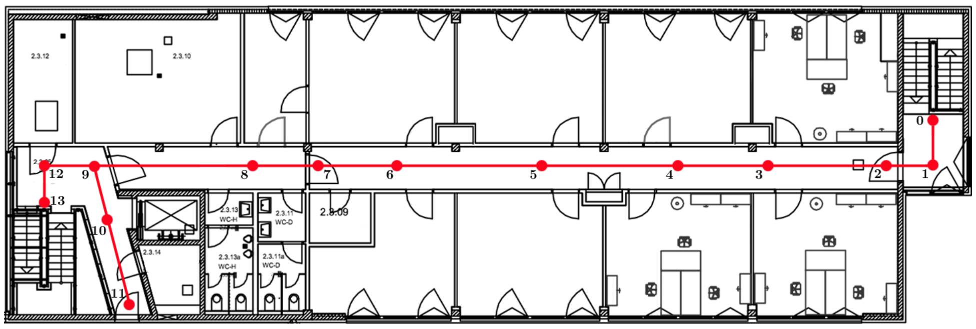


Figure 7.1.: Floor plan of the third floor of the TGS building 2 (with navigation network).

[Bit12a]. Moreover, by utilizing an autonomously working *LocationProvider* component, the user-initiated scanning of NFC tags and QR barcodes is not necessary. Consequently, the measured data is not subject to errors, such as falsely detected steps, that can be introduced during the process of scanning.

The tests are divided into two main scenarios:

Simulation Simulated position data is utilized to evaluate the robustness and performance of the *compass correction* feature and the *map-matching/map-aiding* feature of the indoor navigation application. The presence of different heading errors is represented by the simulated position data. The aim of this test is to evaluate whether the mentioned features are able work properly under the presence of a specific error. The quality of the positioning results is analyzed. The test scenarios for the simulation are as follows:

- SIM–1: Average heading error of 0°.
- SIM–2: Average heading error of 45°.
- SIM–3: Average heading error of 90°.

Field tests During the field test, the test person walks along a defined path on the third floor of the TGS building 2. The user's position is determined by the *multimodal positioning service*. All *position updates* and *position calculations*, that are received from the *multimodal positioning service* are recorded. Additionally, the position information derived from the *compass correction* feature and the *map-matching/map-aiding* feature is stored. The aim of the field test is to evaluate the performance of the entire system under real conditions. Differing in the utilization of enhancements, several configurations that were developed in the context of this work are tested. Position information is provided by the *OpenBeacon RFID* component and the *StepDetector* component (see Figure 3.2) solely. A separate test run had to be done for each configuration because the *multimodal positioning service* provides no features to simulate a pre-recorded test run. The results of each test run are compared to a modeled run that represents the path. The configurations of the most interesting combinations of features are:

- FT–1: None of the enhancements are utilized.
- FT–2: The *compass stabilization* feature of the *multimodal positioning service* is utilized.

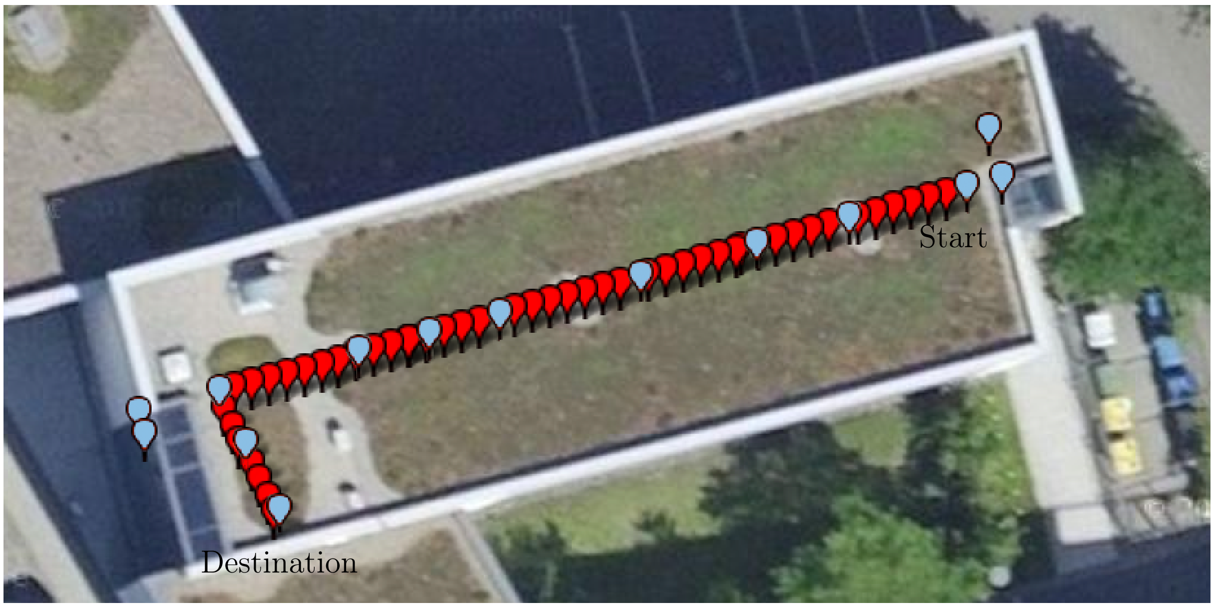


Figure 7.2.: Visualization of the modeled path on *Google Maps*.

- FT–3: A combination of the *compass stabilization* feature and the *compass correction* feature is utilized.
- FT–4: A combination of the *compass stabilization* feature, the *compass correction* feature, and the *map-matching/map-aiding* feature is utilized.

The path of the modeled run is shown in Figure 7.2. The blue marks represent the positions of the *OpenBeacon* RFID tags, which are used by the *multimodal positioning service* to locate the user. The red marks indicate the positions of steps taken by the test person. The path runs along the main hallway, starting from the rear entrance and heading to the front entrance. After the front entrance, a left turn is taken and the path leads past the elevator to the destination of the terrace door. The path has a total distance of 37.5 m. The test person needed 48 steps to walk along the path.

The recorded position information is used to compare the result of each test run to the modeled run. The position information is visualized using the web service of *GPSVisualizer*[1]. With this service, position information in the form of a list of longitude and latitude values can be drawn on a map. The map data is based on *Google Maps*[2]. Furthermore, the Root Mean Square Error (RMSE) of each test configuration is calculated. This measure is frequently used to calculate the average error of values that are estimated by a model. The basis for calculating the RMSE of a configuration is the modeled run.

[1]GPSVisualizer: `http://www.gpsvisualizer.com/`
[2]Google Maps: `https://maps.google.com/`

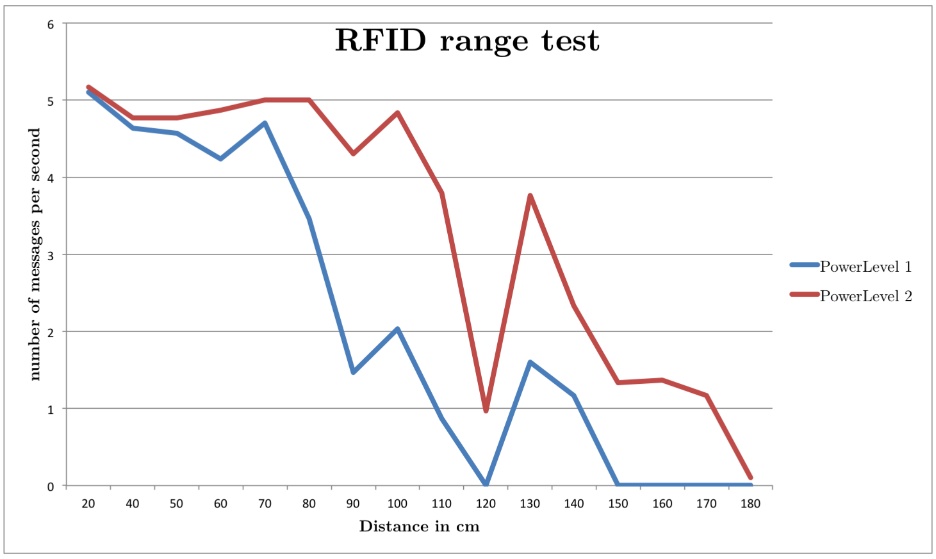


Figure 7.3.: Results of the RFID range test.

The RMSE is defined by

$$RSME = \sqrt{\frac{\sum_{i=1}^{N} \|p_i^r - p_i^e\|^2}{N}} \tag{7.1}$$

where N is the number of positions, p_i^r the position of the i^{th} step of the modeled path and p_i^e the position of the i^{th} step of the estimated path.

7.2. Results of the RFID Test

Separately from the tests of the overall system, the range of the utilized *OpenBeacon* RFID hardware was evaluated. For this test, the maximum receiving range of an *OpenBeacon Proximity* tag and an *OpenBeacon USB2* tag was determined. The *OpenBeacon Proximity* tags act as beacons and are distributed in the indoor environment (blue marks in Figure 7.2). An *OpenBeacon USB2* tag is connected to the smartphone via Bluetooth. It receives the messages sent by the *OpenBeacon Proximity* tags and forwards them to the smartphone.

The *OpenBeacon Proximity* tag sends messages on two power levels, with eight messages for each power level per second. The number of received messages per second is determined at different distances. It is assumed that two messages per second are necessary to detect an *OpenBeacon Proximity* tag reliably.

The results of the RFID range test are illustrated in Figure 7.3. at *power level 1*, more than two messages per second are received up to a distance of 1 m. At *power level 2*, the distance is 1.4 m. These results can be adopted to specify the accuracy of the implemented *OpenBeacon RFID* positioning component.

7.3. Results of the Simulation

For the simulation, the position of each step of a typical test run along the modeled path was calculated and given to the indoor navigation application. The test was run for different initial heading errors (0°, 45°, and 90°), to simulate faulty data of the compass sensor. For each test run, the position information calculated with the help a different set of enabled features were stored. The following sets of features have been tested:

- No features enabled.
- *Compass correction* feature enabled.
- *Compass correction* and *map-matching*/*map-aiding* features enabled.

In this way, the performance of the features and the robustness in regard to initial heading errors can be evaluated. Table 7.1 summarizes the RMSE values for each test configuration.

7.3.1. Test Configuration SIM–1

This test configuration assumes that the heading can be determined exactly. That means, that the initial heading error is 0°. It is obvious, that the results of the simulation should be identical with the positions of the modeled run.

The visualization of the results of this test configuration is shown in the appendix (Figure A.8). All feature combinations provided results that are identical to the modeled path. The RMSE is 0.0 m for all simulations of this test configuration. The *compass correction* and the *map-aiding*/*map-matching* feature work as expected if the correct initial heading is provided.

7.3.2. Test Configuration SIM–2

For this test configuration, an initial heading error of 45°, with regard to the correct heading of the path, was specified. If neither the *compass correction* feature nor the *map-aiding*/*map-matching* feature are utilized, then the estimated positions deviate from

Test Configuration	Features used		
	No features	*Compass correction*	*compass correction & map-aiding/map-matching*
SIM–1	0.0 m	0.0 m	0.0 m
SIM–2	2.16 m	0.79 m	0.12 m
SIM–3	3.99 m	1.45 m	1.58 m

Table 7.1.: RMSE values for the test configurations of the simulation.

the modeled path in an angle of the initial heading error. Whenever the test person is in the proximity of a RFID tag, the position of the test person is reset to the position of the tag (see Figure A.9a). For this case, a RMSE of 2.16 m was calculated.

With the *compass correction* feature enabled, a better positioning result was computed. The heading error was corrected as soon as the second *OpenBeacon* RFID tag was reached. After that, the heading was determined correctly (see Figure A.9b). For the whole path, a RMSE of 0.79 m was calculated.

The deviation from the modeled path was marginal if both, the *compass correction* and the *map-aiding/map-matching* feature, were enabled. Even with the initial heading error of 45°, the *map-aiding/map-matching* algorithm was able to determine the correct link (see Figure A.9c). The RMSE for this case is 0.12 m.

7.3.3. Test Configuration SIM–3

For this test configuration, an initial heading error of 90° was specified. This is not uncommon for indoor environments with many sources of electromagnetic interferences. Again, the raw position values, that were calculated without utilizing the *compass correction* or the *map-aiding/map-matching*, show a deviation in the order of the initial heading error (Figure A.10a). The RMSE for this simulation is 3.99 m.

A similar behavior as in SIM–2 can be observed if only the *compass correction* feature is enabled: the faulty heading could be corrected after the second RFID tag was received. The RMSE was 1.45 m.

With both features enabled, the result was slightly worse in comparison to the configuration which uses the *compass correction* feature only. The RMSE of this run was 1.59 m. The result can be attributed to the fact, that the *map-matching/map-aiding* algorithm identified the wrong link as the link the test person is most likely following.

7.4. Results of the Field Test

In this section, the results of the field test, as described in section 7.1, will be presented and discussed. The RMSE values of the field tests are summarized in Table 7.2.

7.4.1. Test Configuration FT–1

For this test run, no enhancements were utilized. The *multimodal positioning service* provided position information on the basis of the *OpenBeaconRFIDProvider* component and the *StepDirectionDetector* component. The *StepDirectionDetector* component used the values from the compass sensor as provided by the operating system.

Figure 7.4 visualizes the result of this test configuration. Due to the haphazard values, that the compass sensor provided in the given environment, no path is recognizable. In fact, the position was always reset as soon as the test person was in the proximity of an *OpenBeacon* RFID tag, which happened on average every 4.7 m (blue marks in Figure 7.2). The RMSE of this test configuration was 3.20 m. This result is based primarily on the position information derived from the RFID tags. Navigation would not be possible, if the RFID tags are distributed more sparsely and, for this reason, the distances between subsequent RFID tags are farther.

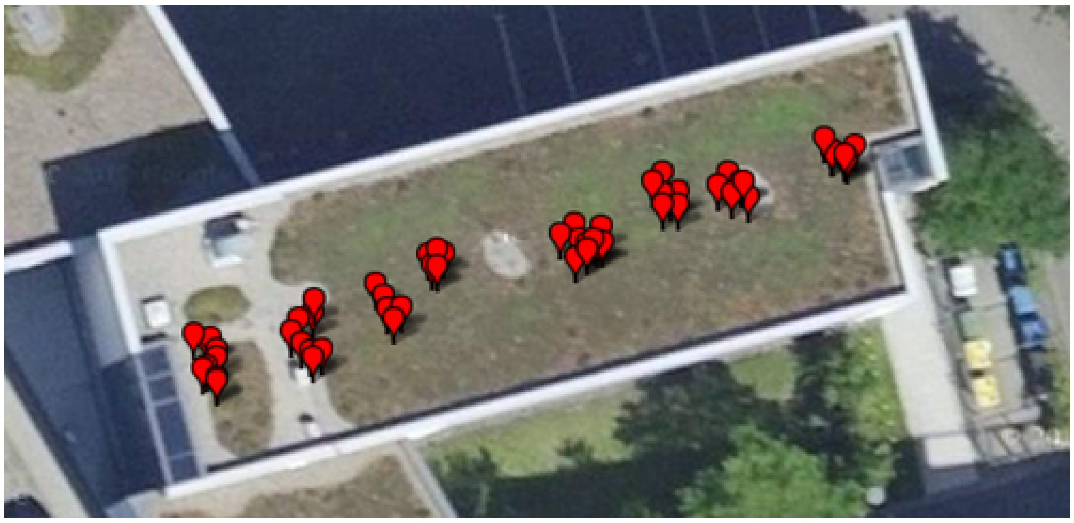


Figure 7.4.: Visualization of the result of test configuration FT–1.

7.4.2. Test Configuration FT–2

For this test configuration only the enhancement of the *compass stabilization* for the *StepDirectionDetector* component was utilized. This results in reasonable values for the heading. The path the test person was walking is recognizable, although the initial heading error is responsible for a significant deviation of the heading (see Figure 7.5). For this configuration, a RMSE of 1.67 m was calculated. Just by stabilizing the raw values of the compass sensor with the help of the gyroscope, the quality of the positioning could be improved notably.

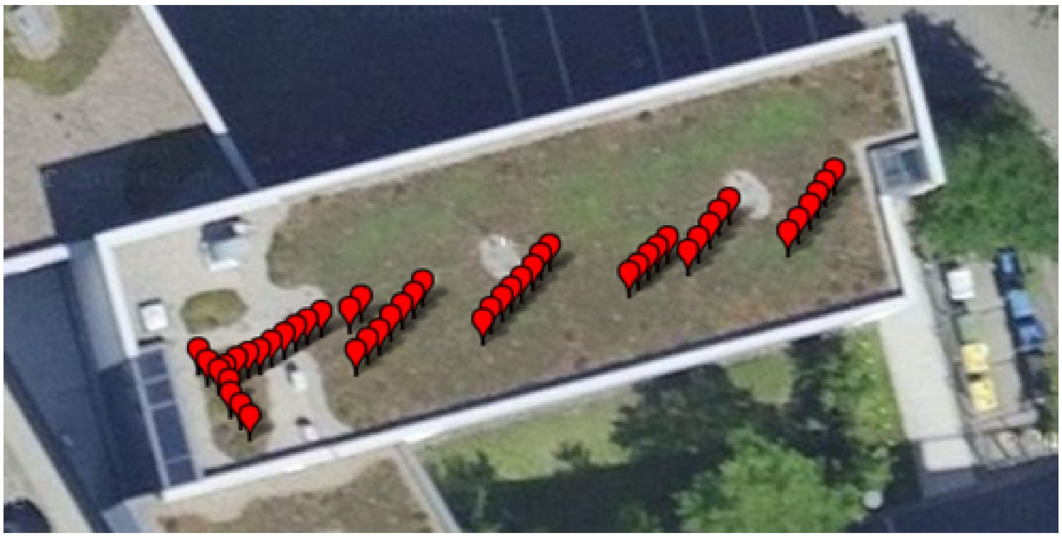


Figure 7.5.: Visualization of the result of test configuration FT–2.

7.4.3. Test Configuration FT–3

For this test configuration, the *compass correction* feature was utilized in addition to the *compass stabilization* feature used in FT–2. The principle of this enhancement is described in subsection 5.1.3. In Figure 7.6, it is clearly visible, that the utilization of this feature can increase the positioning accuracy significantly. The RMSE of this run was 1.05 m, which is a further improvement of the positioning accuracy in comparison to the test configurations FT–2 and FT–1. Except for the last passage before the test person turned left, the correction of the heading values provided accurate results that deviated less than ±10°. The passage, where the results showed an error of above ±10° could be explained with a reset of the *compass correction* process, which is done every two minutes. This is done to avoid large drift errors of the gyroscope. The gyroscope values are reset to zero and a new initial heading is determined using the compass sensor.

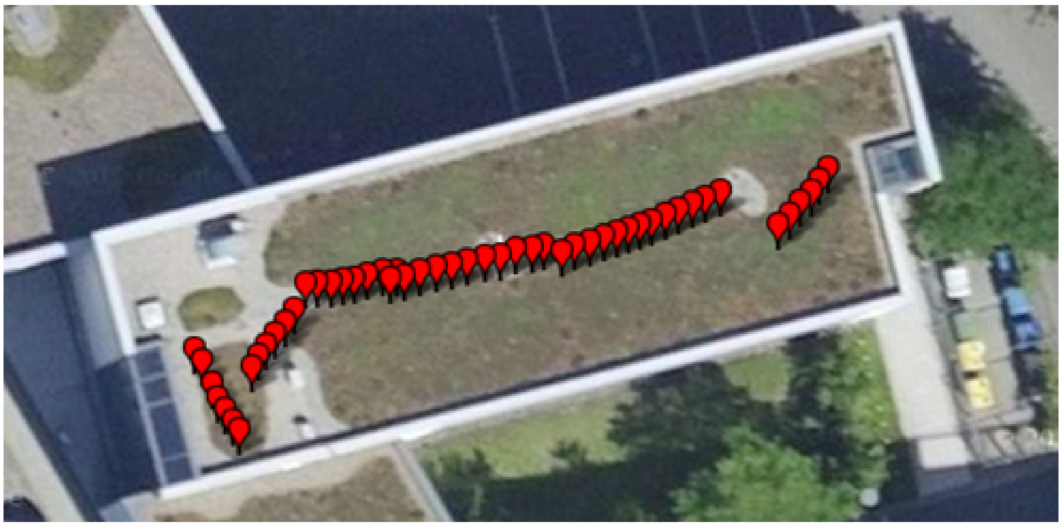


Figure 7.6.: Visualization of the result of test configuration FT–3.

7.4.4. Test Configuration FT–4

For the last test configuration all available enhancements have been utilized. Figure 7.7 visualizes the results of this test run. By enabling the *map-matching*/*map-aiding* feature, the accuracy of the positioning can be further increased. The RMSE of this run is 0.66 m. Due to the position information that is derived from the navigation network, the particles of the Particle filter will move along the links of the network very probably.

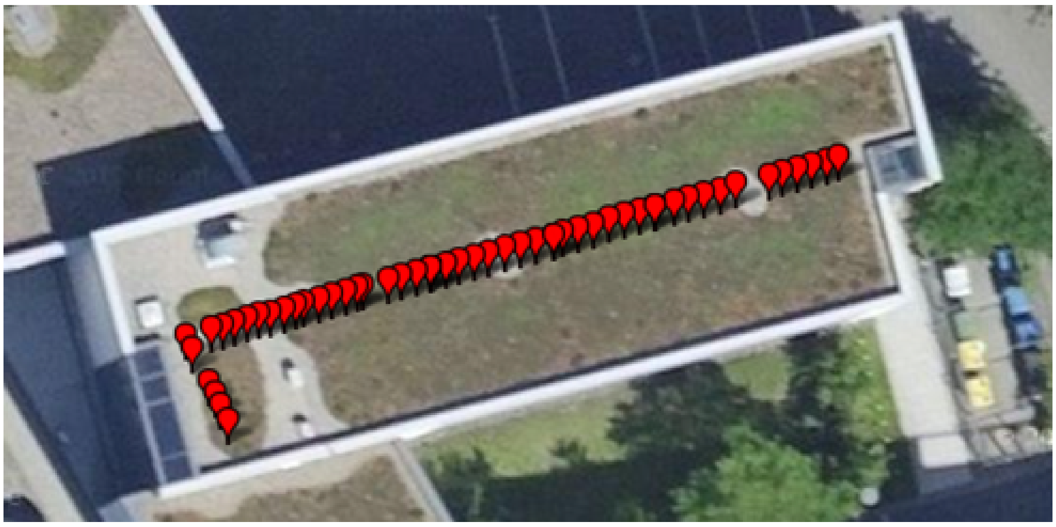


Figure 7.7.: Visualization of the result of test configuration FT–4.

Test Configuration	RMSE
FT–1	3.20 m
FT–2	1.66 m
FT–3	1.05 m
FT–4	0.66 m

Table 7.2.: RMSE values for the test configurations of the simulation.

7.5. Navigation Results

To compute shortest paths, based on the user's current position and a destination which is defined by the user, Dijkstra's algorithm was implemented as described in subsection 6.3.4. The algorithm supports different types of links between the nodes, such as hallways, stairs, and elevators. The graph represents the navigation network of the second and third floor of the TGS building 2. A visualization of the graph can be found in the appendix, Figure A.7a and Figure A.7b. The resulting graph is a connected and valuated graph. As described in section 2.3.2, the path computed by Dijkstra's algorithm is optimal.

7.6. Conclusion

The proposed enhancements have been tested in the environment of the TGS building 2. Each enhancement increased the accuracy of the position information provided by the *multimodal positioning service* (see Table 7.2).

Using only the *compass stabilization* feature produces significant errors in the case of big errors of the initial heading. The positioning might be too inaccurate to provide reliable navigation services.

With the feature combinations of FT–3 (*compass stabilization* and *compass correction*) and FT–4 (*compass stabilization*, *compass correction*, and *map-matching*/*map-aiding*), the accuracy of the calculated position information meets the requirements for indoor navigation purposes. A possible constraint of the *map-aiding* feature arises with regard to the quality and the completeness of the navigation network. With the *map-matching*/*map-aiding* feature enabled, it must be ensured, that the navigation network represents the building very accurately. In particular, open areas might be hard to represent in an indoor navigation network. In these areas, such as entrance halls, the user can move less restricted in comparison to narrow hallways. The process of *map-matching* will be more complicated and could lead to poor positioning results. However, for the less complex office environment of the TGS building 2, the best positioning results were achieved with the *map-matching*/*map-aiding* feature enabled.

8. Conclusion and Outlook

This chapter summarizes the results of this work. Finally, possible improvements and future developments of the system are identified to picture prospective application areas.

8.1. Conclusion

This work describes the conceptual design and implementation of an indoor navigation application that is based on an independent positioning system, called *multimodal positioning service*. This positioning system was designed and developed during a research project in the context of the master's program at the HTW Berlin. It integrates several positioning components and fuses the data received from these to provide accurate position information in the majority of vicinities. Prior to this work, the *multimodal positioning service* could process GPS data and position information encoded in QR bar codes and on NFC tags. The bar codes and tags had to be scanned with the smartphone to obtain the encoded information. Moreover, a step and heading detection was available to track movements of the user. The architecture and the features of the *multimodal positioning service* are described in section 3.2.

To eliminate several problems that arose from the results of the research project, an additional positioning component based on the *OpenBeacon* RFID technology was developed and integrated into the *multimodal positioning service*. With this component the need of manually scanning QR bar codes and NFC tags can be avoided, which eliminates errors that are introduced by motions of the smartphone during the scan process.

The developed indoor navigation application is able to process position information provided by the *multimodal positioning service*. Routes are calculated depending on the user's current position and a specified destination using Dijkstra's shortest path algorithm. Different cost functions have been implemented to customize the route according to the needs of the user, for example to provide barrier-free routes. The indoor navigation application provides guidance through simple navigation instructions based on the computed route and the user's current position.

In addition to the implementation of the indoor navigation application, the development of strategies to improve the quality, that means the accuracy, availability, and reliability, of the position information provided by the *multimodal positioning service*, is the main focus of this work. To achieve these goals, several approaches have been evaluated. The *multimodal positioning service* can be extended to support additional positioning components that are suitable for indoor environments. Moreover, the information that is available in the indoor navigation application, such as topographic and geographic information from navigation networks, can be potentially used to improve the positioning quality of *multimodal positioning service.*

The developed strategies to improve the quality of the positioning resulted in the implementation of several features:

Compass stabilization The compass sensor of actual smartphones is subject to errors in the presence of electromagnetic interference. This is the case in most indoor environments. The output of the compass sensor is stabilized using the gyroscope sensor. The compass sensor provides an initial heading whereas changes in direction are tracked by the gyroscope.

Compass correction Due to the interferences in indoor environments the compass sensor often provides faulty values. To compensate these errors, the heading of the path between subsequent RFID positioning results is calculated. A correction value is computed based on the heading provided by the compass sensor and the calculated heading of the path.

Map-matching/map-aiding To provide navigation services, the indoor navigation application must provide topological and geographical information of buildings in the covered environment in the form of navigation networks. The positions received from the *multimodal positioning system* can be mapped to links of the navigation networks. Because the movements of users are restricted by the structure of the building, the mapped positions can be used to improve the quality of the positioning.

The proposed features have been evaluated individually and in promising combinations with regard to the positioning quality. The results of the tests, which were conducted in the *Technologie und Gründerzentrum Spreeknie Berlin* (TGS), show that each feature leads to an improvement of the positioning quality. For the test environment, the best results were achieved when all proposed features were utilized. In this case, the RMSE of the positioning was 0.66 m. However, even without using the *map-matching/map-aiding* feature, the quality of the position information is suitable for indoor navigation, providing position information with a RMSE of 1.05 m.

8.2. Outlook

The prototype of the indoor navigation application is kept very simple and was focused on the implementation of the previously described enhancements. The accuracy of the positioning was satisfactory for providing navigation capabilities in the test environment. Further tests have to be conducted in environments that, on the one hand, are more complex in regard to the navigation network and, on the other hand, provide a more sparsely distributed positioning infrastructure. The representation of the navigation network can be enhanced to improve the performance in complex indoor environments. Lorenz and Ohlbach [LO06] present a potential enhanced model for this scenario.

A major problem of the positioning system is the noisy data provided by the internal sensors of the smartphone. The detection of movements through the integrated sensors of the smartphone is particularly difficult. The error that is introduced by the noisy sensor data propagates with each motion of the device. This is a problem if the smartphone is used to scan QR bar codes and NFC tags. A potential solution is the utilization of an external device that performs the task of detecting the user's heading and steps. However, this solution would be contrary to the concept of providing the navigation service on a single consumer smartphone.

Moreover, the indoor navigation system could benefit from a context-sensitive user interface. Until now, only the textual presentation of the navigation instructions is implemented. Several research studies deal with the presentation of navigation instructions on mobile devices, for example [MZW08]. Different presentation forms, such as audio/text, maps, and augmented reality (AR), can be used in varying scenarios, depending on the user's current task, position and its accuracy.

Bibliography

[Abd+11] Khairi Abdulrahim et al. “Integrating Low Cost IMU with Building Heading In Indoor Pedestrian Navigation”. In: *Journal of Global Positioning Systems* 10.1 (2011), pp. 30–38.

[Abe08] James S. Aber. *Brief History of Maps and Cartography.* 2008. URL: http://academic.emporia.edu/aberjame/map/h_map/h_map.htm (visited on 06/30/2012).

[Alg] AlgoList. *Dijkstra's algorithm in Java.* URL: http://www.algolist.com/code/java/Dijkstra%27s_algorithm (visited on 08/12/2012).

[BC08] Stefano Burigat and Luca Chittaro. “Decluttering of Icons Based on Aggregation in Mobile Maps”. In: *Map-based Mobile Services.* Ed. by Liqiu Meng, Alexander Zipf, and Stephan Winter. Lecture Notes in Geoinformation and Cartography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. Chap. 2, pp. 13–32. ISBN: 978-3-540-37109-0.

[Beh00] Ehrhard Behrends. *Introduction to Markov Chains.* Vieweg, 2000.

[Bil12] Andreas Bilke. “Ortung und Navigation mit mobilen Geräten in Gebäuden”. Master's thesis. Hochschule für Technik und Wirtschaft Berlin, 2012.

[Bit11] Björn Bittins. “Multisensor and Collaborative Localization for Diverse Environments”. In: *Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS), 2011.* 2011.

[Bit12a] Björn Bittins. “Multimodal and Collaborative Localisation Service for Diverse Environments”. In: 1^{st} *IEEE International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems, 2012.* 2012.

[Bit12b] Björn Bittins. “Multimodal Localization System for Diverse Environments”. In: 13^{th} *International Conference Modern Information and Electronic Technologies (MIET), 2012.* 2012.

[BS08] M. Bouet and A.L. dos Santos. “RFID tags: Positioning principles and localization techniques”. In: *Wireless Days, 2008. WD '08. 1st IFIP.* Nov. 2008, pp. 1–5. DOI: 10.1109/WD.2008.4812905.

[BSH10] Stephan Bergemann, Jürgen Sieck, and Michael Herzog. "Contact Based Wireless Identification of Moving Objects Using Active RFID Technology". In: *IEEE Second International Conference on Computational Intelligence, Modelling and Simulation.* 2010.

[BV10] Joydeep Biswas and Manuela Veloso. "Wifi localization and navigation for autonomous indoor mobile robots". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* May 2010, pp. 4379–4384.

[Cis08] Cisco Systems, Inc. *Wi-Fi Location-Based Services 4.1 Design Guide.* May 2008. URL: `http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/lbswifig_external.pdf`.

[Col+11] Audrey Colbrant et al. *One Idea and Three Concepts for Indoor-Outdoor Navigation.* Tech. rep. RR-7849. INRIA, Dec. 2011, p. 12.

[Dep08] Department of Defense. *GPS SPS Performance Standard.* 4^{th} Edition. Washington, 2008.

[Dij59] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271.

[ElR02] El-Rabbany. *Introduction to GPS: the Global Positioning System.* Artech House, 2002.

[Fin06] Klaus Finkenzeller. *RFID Handbuch.* 4^{th} Edition. Hanser, 2006.

[FM10] Klaus Finkenzeller and Dörte Müller. *RFID Handbook.* 3^{rd} Edition. John Wiley & Sons, 2010.

[Gar] Gartner Inc. *Gartner Identifies 10 Consumer Mobile Applications to Watch in 2012.* URL: `http://www.gartner.com/it/page.jsp?id=1544815` (visited on 04/27/2012).

[GM03] Pierre-Yves Gilliéron and Bertrand Merminod. "Personal Navigation System for Indoor Applications". In: *Proceedings of the* 11^{th} *IAIN World Congress.* Berlin (Germany), 2003.

[Gooa] Google Inc. *Android Developers - Dashboards.* URL: `http://developer.android.com/about/dashboards/index.html` (visited on 08/10/2012).

[Goob] Google Inc. *Android Developers - Reference - SensorEvent.* URL: `http://developer.android.com/reference/android/hardware/SensorEvent.html` (visited on 08/07/2012).

[Goo12b] Google Inc. *Google Maps*. 2012. URL: `http://maps.google.com` (visited on 07/01/2012).

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEEE Proceedings F, Radar and Signal Processing* 140.2 (1993), pp. 107–113.

[Han08] Ramon van Handel. *Hidden Markov Models - Lecture Notes*. July 2008. URL: `http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf` (visited on 05/09/2012).

[Har04] Prof. Dr. Peter Hartmann. *Mathematik für Informatiker*. Vol. 3rd Edition. Friedrich Vieweg & Sohn, 2004.

[HG10] Haosheng Huang and Georg Gartner. “A Survey of Mobile Indoor Navigation Systems”. In: *Cartography in Central and Eastern Europe*. Ed. by Georg Gartner and Felix Ortag. Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 2010, pp. 305–319.

[HLW03] B. Hofmann-Wellenhof, K. Legat, and M. Wieser. *Navigation: Principles of Navigation and Guidance*. Springer, 2003.

[ISO00] ISO/IEC. *ISO/IEC 18004*. June 2000. URL: `raidenii.net/files/datasheets/misc/qr_code.pdf` (visited on 05/05/2012).

[Jok08] Kristiina Jokinen. “User Interaction in Mobile Navigation Applications”. In: *Map-based Mobile Services*. Ed. by Liqiu Meng, Alexander Zipf, and Stephan Winter. Lecture Notes in Geoinformation and Cartography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. Chap. 9, pp. 168–197. ISBN: 978-3-540-37109-0.

[Kál60] R. E. Kálmán. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME – Journal of Basic Engineering* 82 (1960), pp. 35–45.

[Kar11] Hassan A. Karimi. *Universal Navigation on Smart Phones*. Springer-Verlag New York Inc., 2011.

[Kle97] Rolf Klein. *Algorithmische Geometrie*. Vol. 1st Edition. Addison Wesley Longman, 1997.

[Kot+03] Antti Kotanen et al. “Experiments on Local Positioning with Bluetooth”. In: *Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC’03)*. 2003.

[Küp05] Axel Küpper. *Location-Based Services : Fundamentals and Operation.* Wiley, 2005.

[Kus+05] Manish Kushwaha et al. "Sensor Node Localization Using Mobile Acoustic Beacons". In: *The* 2^{nd} *IEEE International Conference on Mobile Ad-hoc and Sensor Systems.* 2005.

[Lam] Philip R. Lamb. *ARToolit.* URL: `www.hitl.washington.edu/artoolkit/` (visited on 05/05/2012).

[LHL09] Martin E. Liggins, David L. Hall, and James Llinas, eds. *Handbook of Multisensor Data Fuison.* 2^{nd} Edition. CRC Press, 2009.

[LO06] Bernhard Lorenz and Hans Jürgen Ohlbach. "A Hybrid Spatial Model for Representing Indoor Environments". In: *In Proceedings of the* 6^{th} *International Symposium on Web and Wireless Geographical Information Systems (W2GIS 2006) – forthcoming – December 4-5, 2006, Hong Kong, China, Lecture Notes in Computer Science.* Springer, 2006, pp. 102–112.

[LSW09] Manh Hung V. Le, Dimitris Saragas, and Nathan Webb. "Indoor Navigation System for Handheld Devices". Bachelor's thesis. Worcester Polytechnic Institute, 2009.

[Men08] Liqiu Meng. "The State of the Art of Map-Based Mobile Services". In: *Map-based Mobile Services.* Ed. by Liqiu Meng, Alexander Zipf, and Stephan Winter. Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 2008. Chap. 1, pp. 1–12.

[Mit10] H. B. Mitchell. *Multi-Sensor Data Fusion.* Springer Berlin, 2010.

[Miu02] Allen Ka Lun Miu. "Design and Implementation of an Indoor Mobile Navigation System". Master's thesis. Massachusetts Institure of Technology, 2002.

[Movb] Movable Type Ltd. *Calculate distance, bearing and more between Latitude/-Longitude points.* URL: `http://www.movable-type.co.uk/scripts/latlong.html` (visited on 08/11/2012).

[MZW08] Prof. Dr. Liqui Meng, Prof. Dr. Alexander Zipf, and Dr. Stephan Winter, eds. *Map-based Mobile Services - Design, Interaction and Usability.* Lecture Notes in Geoinformation and Cartography. Springer-Verlag, 2008.

[Oxf] Oxford University Press. *Oxford Dictionary: Augemented Reality.* URL: `http://oxforddictionaries.com/definition/augmented%2Breality` (visited on 05/05/2012).

[Ozd+11] Busra Ozdeizci et al. “Development of an Indoor Navigation System Using NFC Technology”. In: *Fourth International Conference on Information and Computing (ICIC)*. 2011, pp. 11–14.

[Pei+10] Ling Pei et al. “Using Inquiry-based Bluetooth RSSI Probability Distributions for Indoor Positioning”. In: *Journal of Global Positioning Systems* 9.2 (2010), pp. 122–130.

[Pep] Julianne Pepitone. *Android races past Apple in smartphone market share*. URL: `http://money.cnn.com/2012/08/08/technology/smartphone-market-share/index.html` (visited on 08/12/2012).

[PKC01] P. Prasithsangaree, P. Krishnamurthy, and P. K. Chrysanthis. “On Indoor Position Location With Wireless Lans”. In: 13th *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2002)*. 2001, pp. 720–724.

[Pui+09] Arto Puikkonen et al. “Towards designing better maps for indoor navigation: experiences from a case study”. In: *Proceedings of the* 8th *International Conference on Mobile and Ubiquitous Multimedia*. MUM ’09. Cambridge, United Kingdom: ACM, 2009, 16:1–16:4. ISBN: 978-1-60558-846-9.

[Reh+05] Karl Rehrl et al. “Combined Indoor/Outdoor Smartphone Navigation for Public Transport Travellers”. In: *Location Based Services & Telecartography - Proceedings of the Symposium 2005*. 2005.

[REI08] Javier Rodas, Carlos J. Escudero, and Daniel I. Iglesia. “Bayesian Filtering for a Bluetooth Positioning System”. In: *IEEE International Symposium on Wireless Communication Systems*. 2008, pp. 618–622.

[Ren+07] Valérie Renaudin et al. “Indoor Navigation of Emergency Agents”. In: *European Journal of Navigation* 5.3 (July 2007).

[Sch09] Jan Scholze. “Entwicklung eines integrierten Ansatzes zur Indoor- und Outdoor-Positionsbestimmung auf Basis einer Föderation von Gebäudedatenservern”. Diploma thesis. Technische Universität Dresden, 2009.

[SCM10] Alberto Serra, Davide Carboni, and Valentina Marotto. “Indoor Pedestrian Navigation System Using a Modern Smartphoneedestrian navigation system using a modern smartphone”. In: *Proceedings of the* 12th *International Conference on Human Computer Interaction with Mobile Devices and Services*. 2010, pp. 397–398.

[Str+08] T. Strang et al. *Lokalisierungsverfahren*. Deutsches Zentrum für Luft- und Raumfahrt (DLR), 2008.

[Str10] Christian Stropp. “Kontextbasierte Multimodale Positionsbestimmung mit Hilfe von Smartphones”. Bachelor’s thesis. Hochschule für Technik und Wirtschaft Berlin, 2010.

[Wen+06] K. Wendlandt et al. “Continuous location and direction estimation with multiple sensors using particle filtering”. In: *Proc. 2006 International Conference in Multisensor Fusion and Integration for Intelligent Systems (MFI 2006)*. 2006.

[Wen07] Jan Wendel. *Integrierte Navigationssysteme*. Oldenburg Verlag, 2007.

[Wid10] Widyawan. *Indoor Navigation: State of the Art and Novel Algorithms*. Lambert Academic Publishing, 2010.

Image References

[Egl] Chuck Eglinton. *QR code structure*. URL: http://www.qrcodeshowto.com/wp-content/uploads/2011/07/QR-code-specifications.jpg (visited on 05/05/2012).

[Goo12a] Google Inc. *Coordinate system used by Android's SensorEvent API*. 2012. URL: http://developer.android.com/images/axis_device.png (visited on 08/07/2012).

[Ind08] Inductiveload. *Normal Distribution PDF*. 2008. URL: http://en.wikipedia.org/wiki/File:Normal_Distribution_PDF.svg.

[Isl12] Mubashar Islam. *How GSM Netwok work*. Jan. 2012. URL: http://www.techviral.com/wp-content/uploads/2012/01/telephonie-mobile-images-reseau-cellulaire.png (visited on 05/03/2012).

[LON11] LONO Creative Ltd. *PWC Office Singage Design*. 2011. URL: http://www.lonocreative.com/blog/wp-content/uploads/2011/03/pwc-office-signage-design.jpg (visited on 07/02/2012).

[Mer10] Milosch Meriac. *OpenBeacon Proximity Tag*. Sept. 2010. URL: http://www.openbeacon.org/images/Sputnik-25C3.jpg (visited on 05/06/2012).

[Mova] Movable Type Ltd. *Baghdad to Osaka*. URL: http://www.movable-type.co.uk/scripts/baghdad-to-osaka.jpg (visited on 08/11/2012).

List of Figures

List of Tables

Index of Algorithms

Index of Listings

A. Appendix

A.1. Listings

```
@Override
	public IBinder onBind(Intent intent) {
		return new ILocationService.Stub() {
			// Stub methods generated from AIDL interface
			@Override
			public de.bittins.bjoern.locationService.model.Location
				getLocation() throws RemoteException {
				return locationCenter.getMostAccurateLocation();
			}

			@Override
			public de.bittins.bjoern.locationService.model.LocationRequest
				getLocationRequest() {
				return locationCenter.getLocationRequest();
			}

			@Override
			public void reportCorrection(Correction correction)
					throws RemoteException {
				manageCorrections(correction);
			}

			@Override
			public void reportLocation(Location location)
					throws RemoteException {
				manageLocations(location);
			}
		};
	}
```

Listing A.1: Implementation of the AIDL interface in the *multimodal positioning service*

```java
/**
 * Calculates the distance between two given coordinates (lat/lon) in meters
 * @param lat1 latitude of first coordinate
 * @param lng1 longitude of first coordinate
 * @param lat2 latitude of second coordinate
 * @param lng2 longitude of second coordinate
 * @return distance in meters
 */
public static double calculatDistanceBetweenCoordinates(double lat1, double lng1, double lat2, double lng2) {
    double dLat = Math.toRadians(lat2-lat1);
    double dLng = Math.toRadians(lng2-lng1);
    double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
        Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *
        Math.sin(dLng/2) * Math.sin(dLng/2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    double dist = EARTH_RADIUS * c;
    return dist;
}
```

Listing A.2: Method used to calculate the distance between two points, given their coordinates.

```java
/**
 * Calculate the destination coordinates from start coordinates, distance
    and bearing.
 * @param lat Latitude of start point
 * @param lon Longitude of start point
 * @param dist Distance in meters
 * @param bear bearing in degrees
 * @return destination coordinates double[]{lat, lon}
 */
public static double[]
    calculateDestinationCoordinatesFromStartCoordinatesAndDistanceAndBearing
    (double lat, double lon, double dist, double bear) {
    double lat1 = Math.toRadians(lat);
    double lon1 = Math.toRadians(lon);
    double dist1 = dist/(EARTH_RADIUS); //Earth's radius in km
    double bear1 = Math.toRadians(bear);

    double lat2 = Math.asin(Math.sin(lat1)*Math.cos(dist1)+Math.cos(lat1)
        * Math.sin(dist1)*Math.cos(bear1));
    double lon2 = lon1 + Math.atan2(Math.sin(bear1)*Math.sin(dist1)*Math.
        cos(lat1), Math.cos(dist1)-Math.sin(lat1)*Math.sin(lat2));

    return (new double[]{Math.toDegrees(lat2), Math.toDegrees(lon2)});
}
```

Listing A.3: Method used to calculate destination coordinates, given the origin coordinates, the heading, and the distance

A.2. Figures

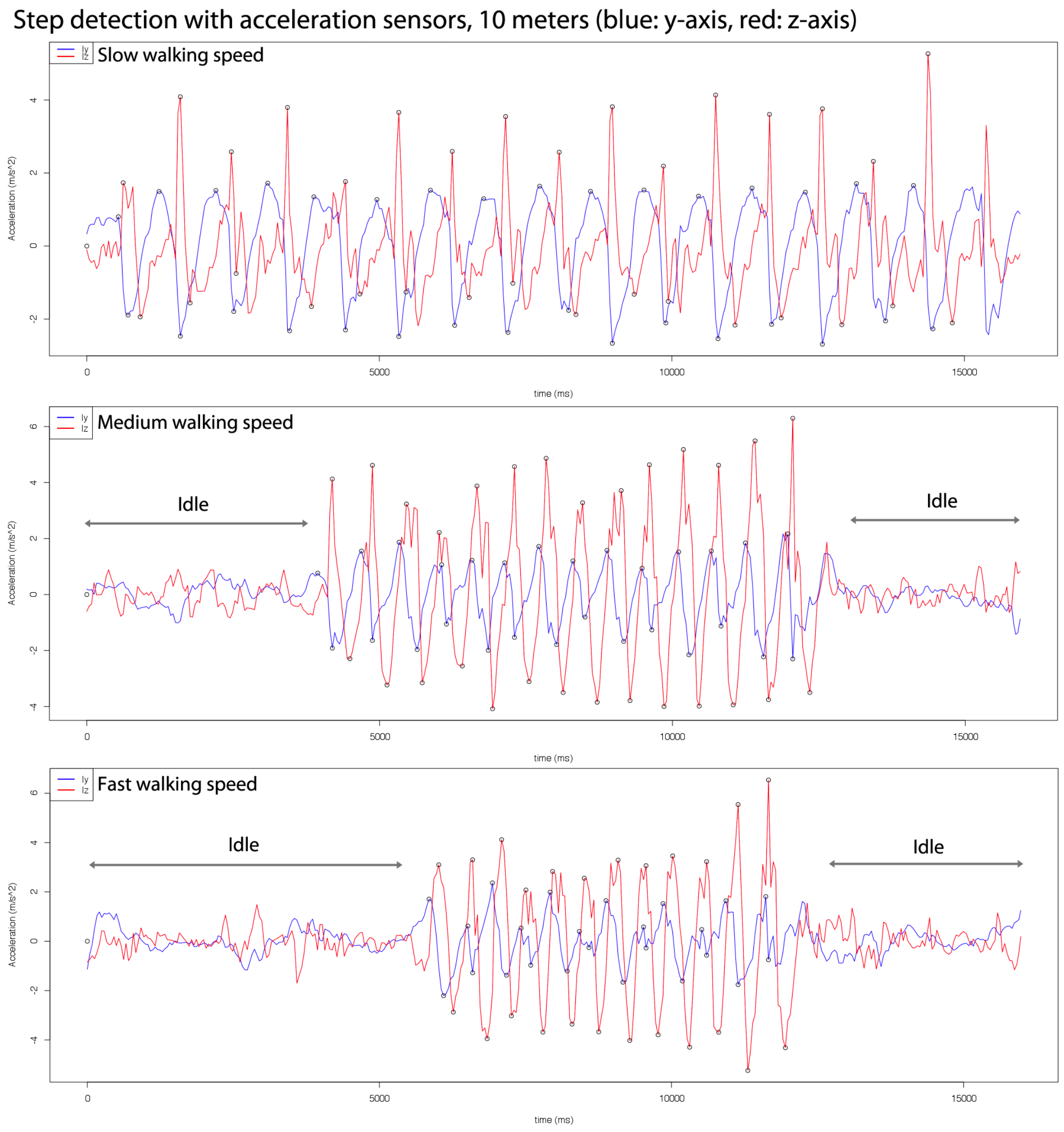


Figure A.1.: Different modes of walking speed detected by step detection algorithm. [Bit12a]

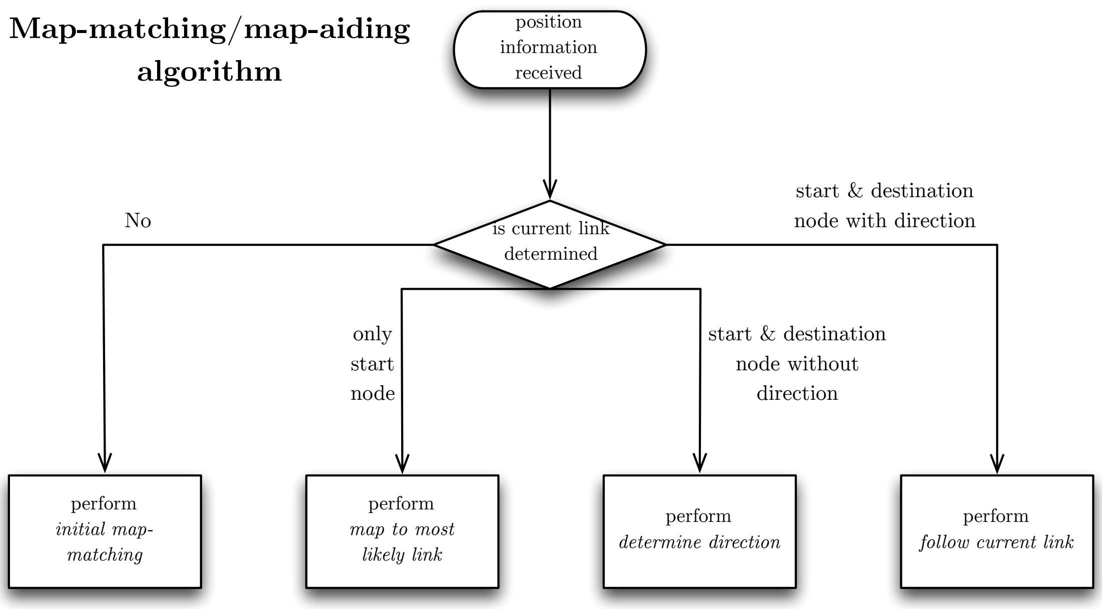

Map-matching/map-aiding algorithm
position information received
is current link determined
No
only start node
start & destination node without direction
start & destination node with direction
perform *initial map-matching*
perform *map to most likely link*
perform *determine direction*
perform *follow current link*

Figure A.2.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 1: determining degree of knowledge about the current link.

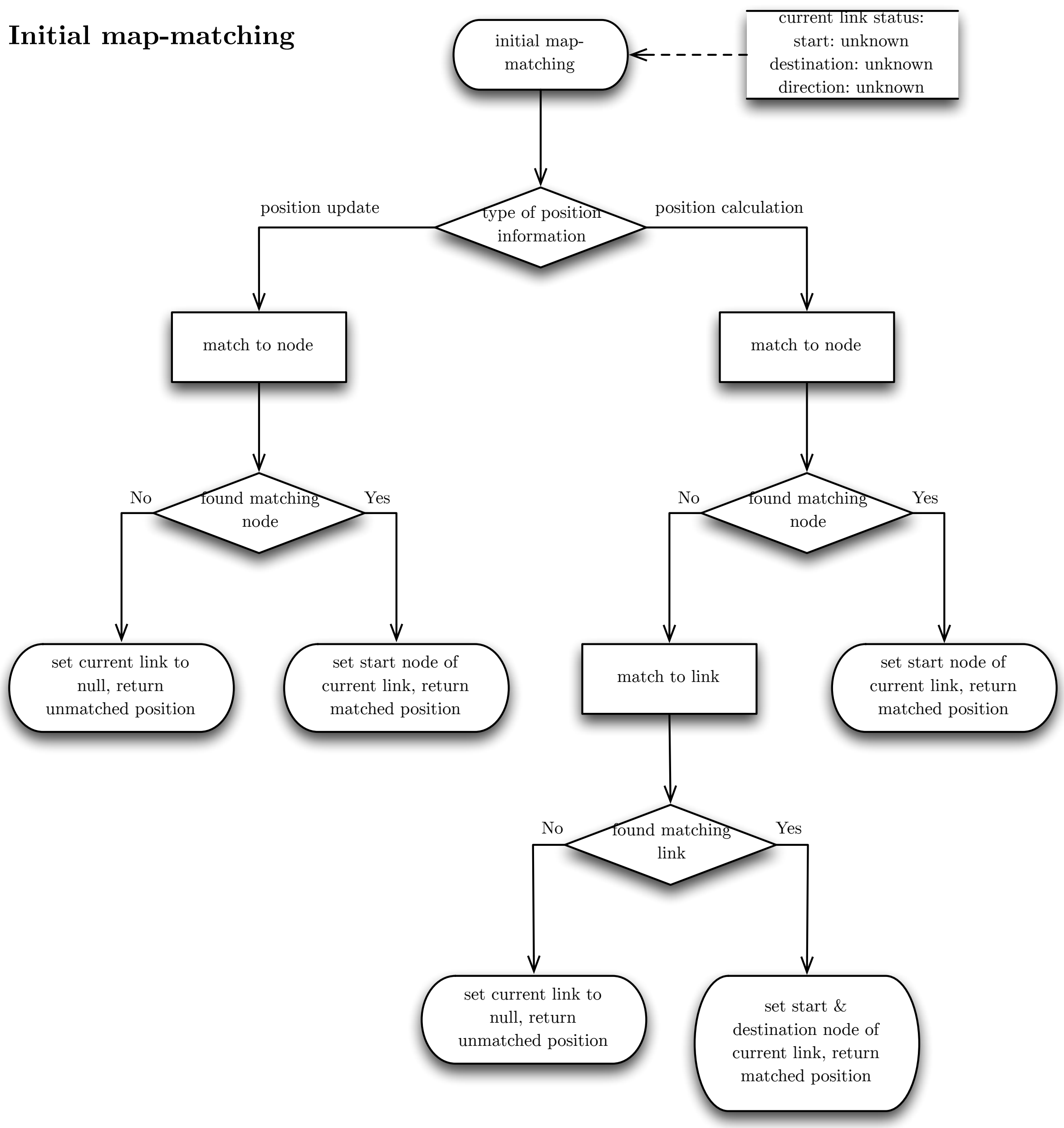

Initial map-matching
initial map-matching
current link status:
start: unknown
destination: unknown
direction: unknown
type of position information
position update
position calculation
match to node
match to node
found matching node
No
Yes
found matching node
No
Yes
set current link to null, return unmatched position
set start node of current link, return matched position
match to link
set start node of current link, return matched position
found matching link
No
Yes
set current link to null, return unmatched position
set start & destination node of current link, return matched position

Figure A.3.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 2: initial mapping of received position to the navigation network.

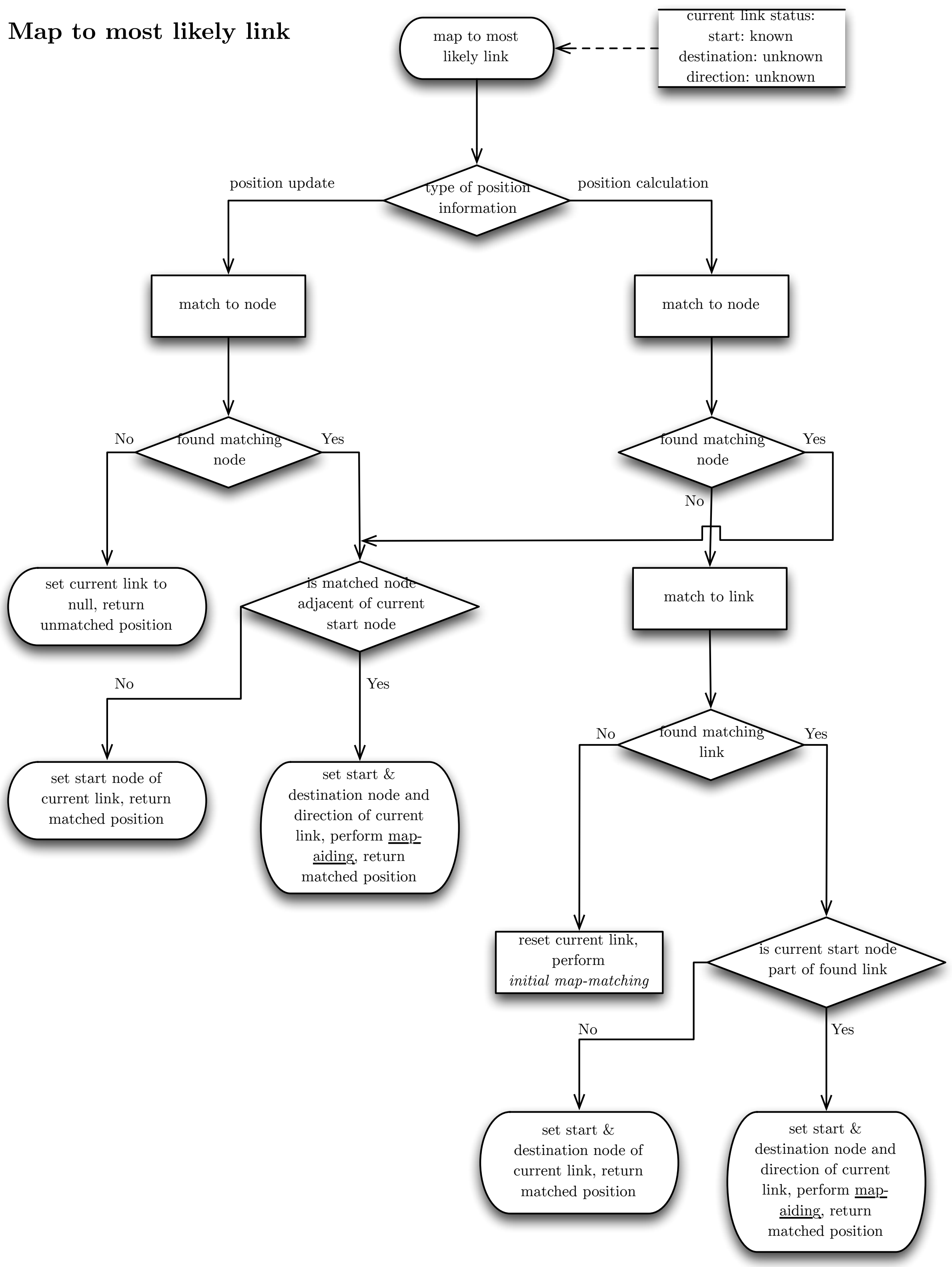

Map to most likely link
map to most likely link
current link status:
start: known
destination: unknown
direction: unknown
type of position information
position update
position calculation
match to node
match to node
found matching node
No
Yes
found matching node
Yes
No
set current link to null, return unmatched position
is matched node adjacent of current start node
match to link
No
Yes
found matching link
No
Yes
set start node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position
reset current link, perform initial map-matching
is current start node part of found link
No
Yes
set start & destination node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position

Figure A.4.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 3: mapping the received position to a link of the navigation network.

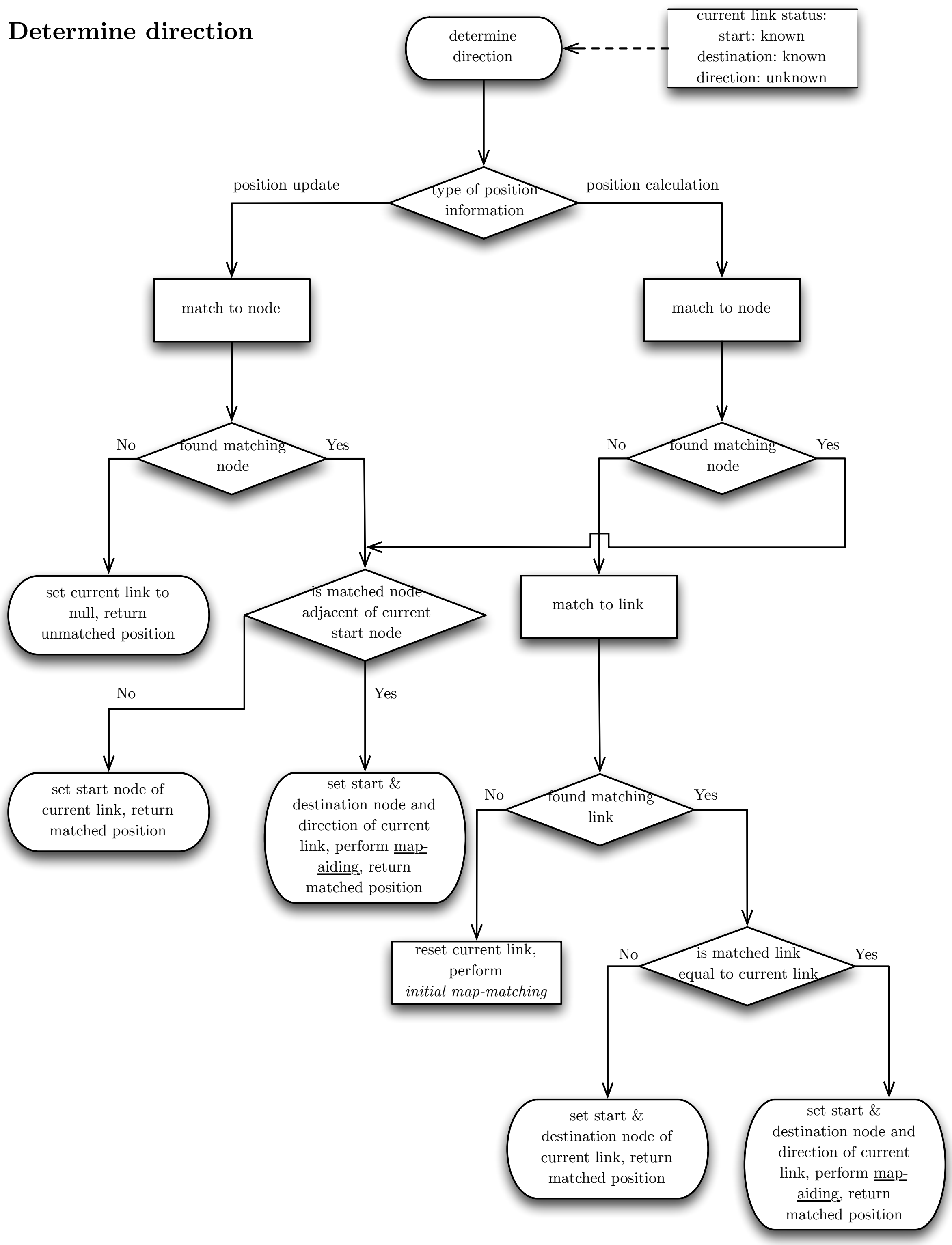

Determine direction
determine direction
current link status:
start: known
destination: known
direction: unknown
type of position information
position update
position calculation
match to node
match to node
found matching node
No
Yes
found matching node
No
Yes
set current link to null, return unmatched position
is matched node adjacent of current start node
match to link
No
Yes
set start node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position
found matching link
No
Yes
reset current link, perform initial map-matching
is matched link equal to current link
No
Yes
set start & destination node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position

Figure A.5.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 4: determining the direction based on the current link and the received position.

Follow current link

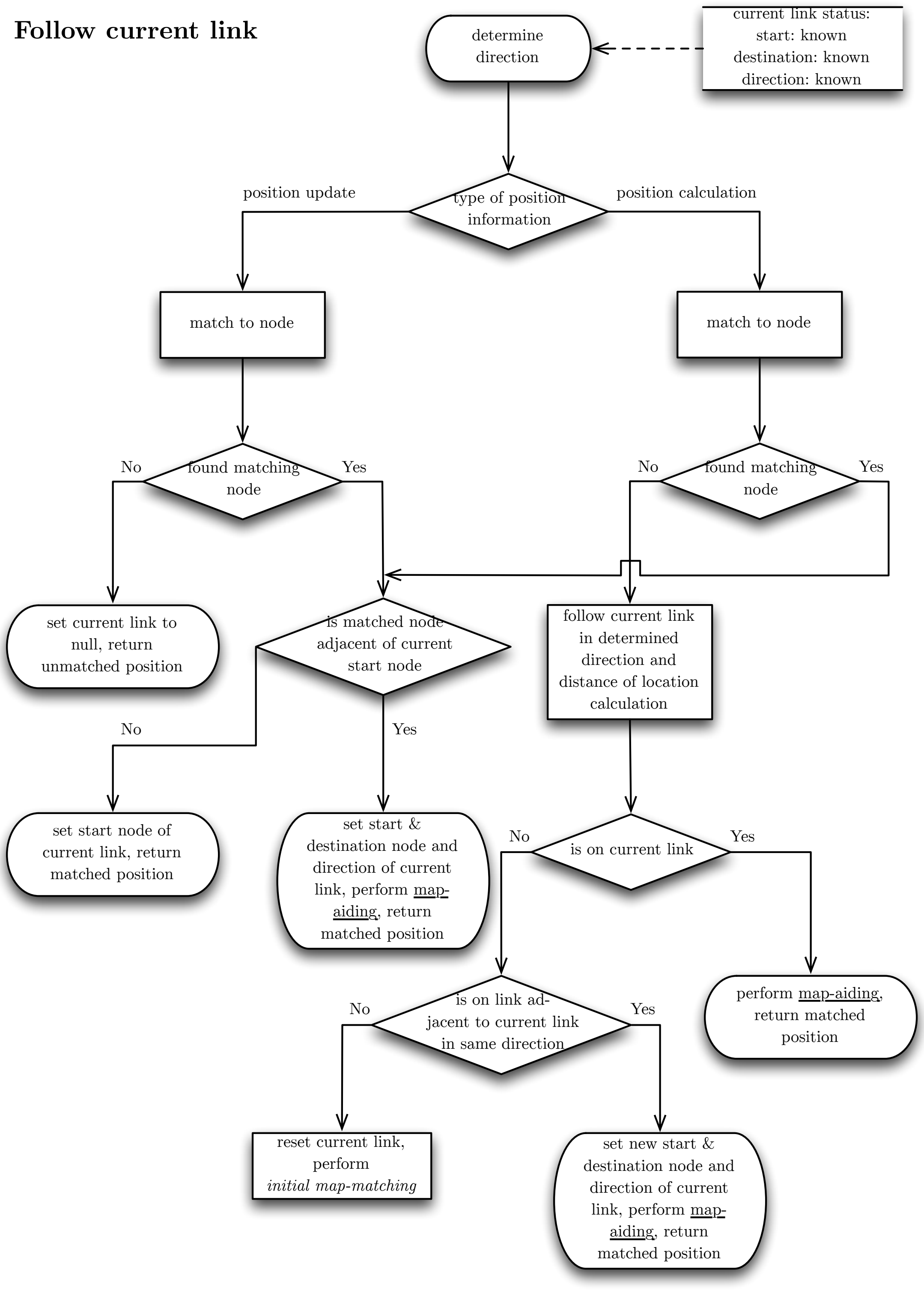
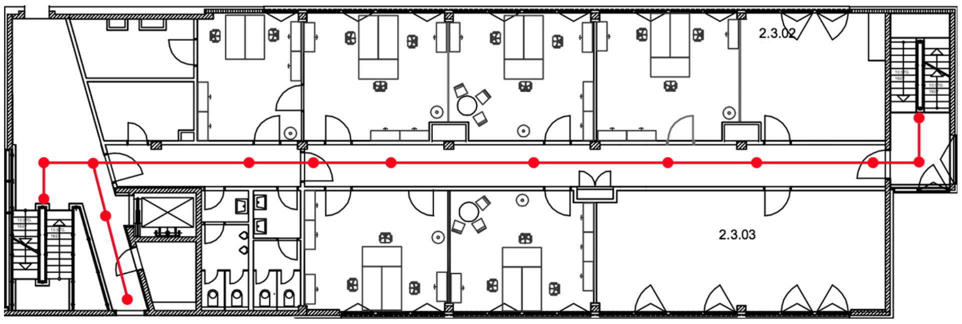
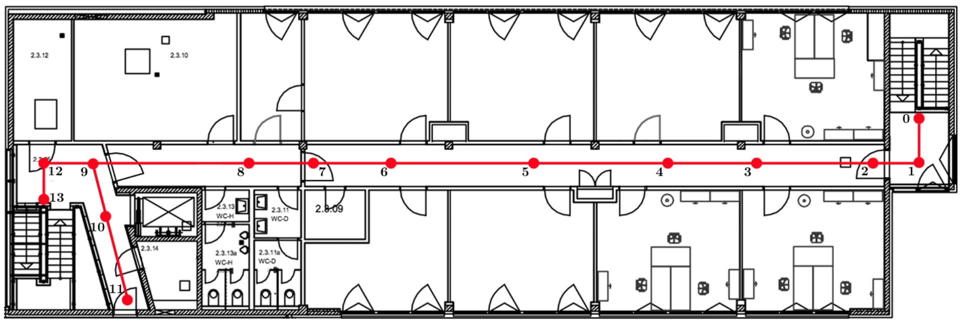

current link status:
start: known
destination: known
direction: known
determine direction
type of position information
position update
position calculation
match to node
match to node
found matching node
No
Yes
found matching node
No
Yes
set current link to null, return unmatched position
is matched node adjacent of current start node
No
Yes
follow current link in determined direction and distance of location calculation
set start node of current link, return matched position
set start & destination node and direction of current link, perform map-aiding, return matched position
is on current link
No
Yes
perform map-aiding, return matched position
is on link adjacent to current link in same direction
No
Yes
reset current link, perform initial map-matching
set new start & destination node and direction of current link, perform map-aiding, return matched position

Figure A.6.: Sequence diagram of the *map-matching*/*map-aiding* algorithm, part 5: following the current link.

Navigation Network



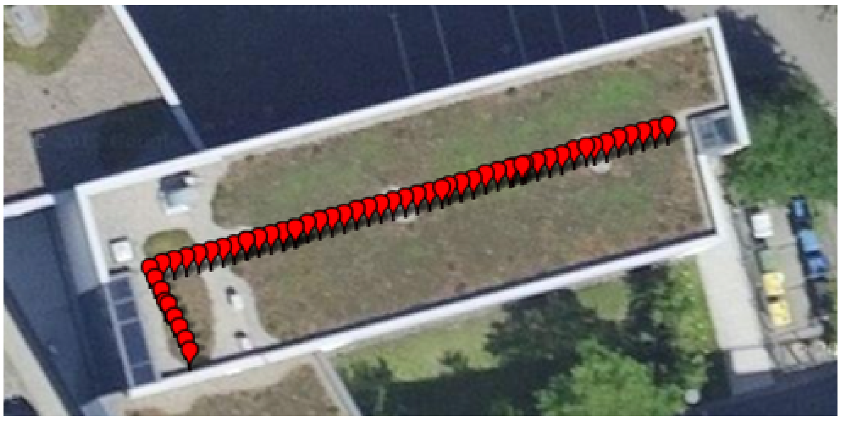
(a) Navigation network of the second floor.

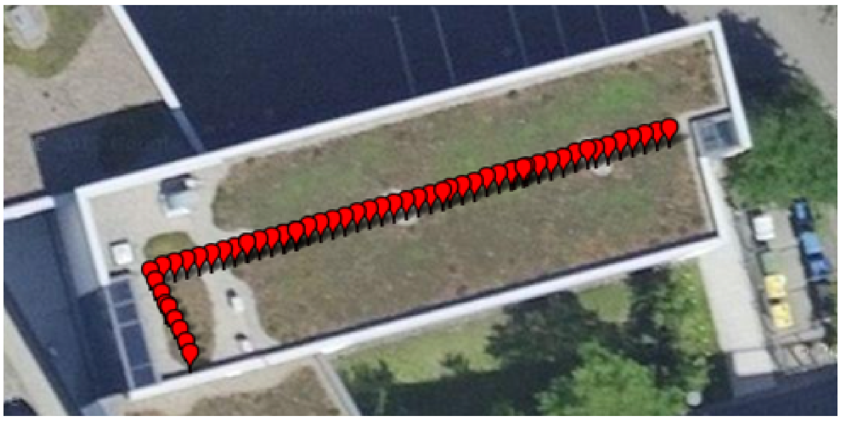


(b) Navigation network of the third floor.

Figure A.7.: Visualization of the navigation network of the TGS.

Test Results


(a) Raw positions.


(b) Results of *compass correction* feature.

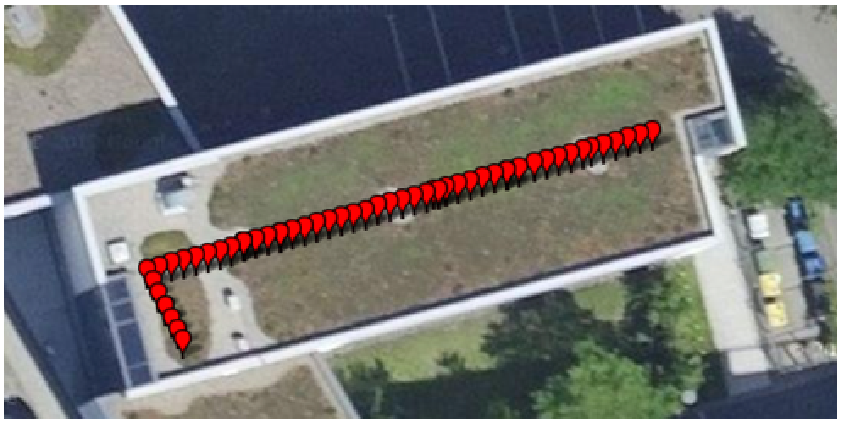
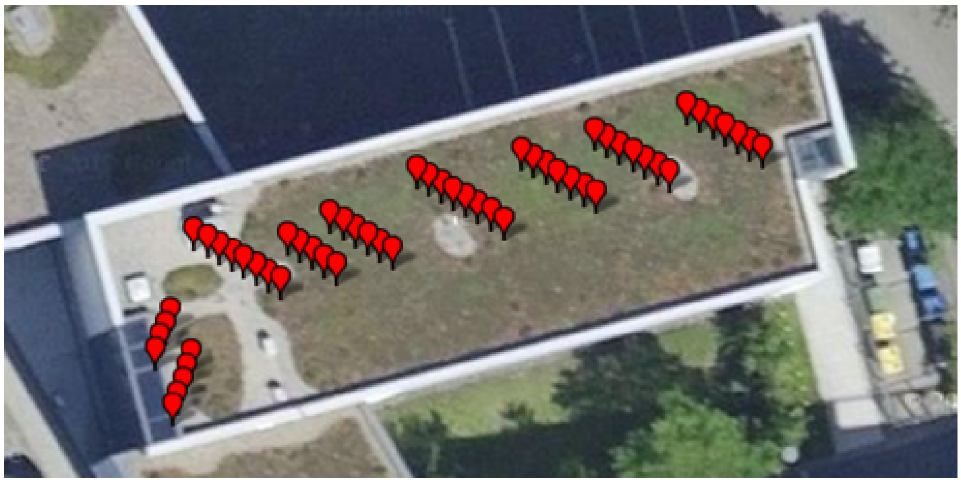
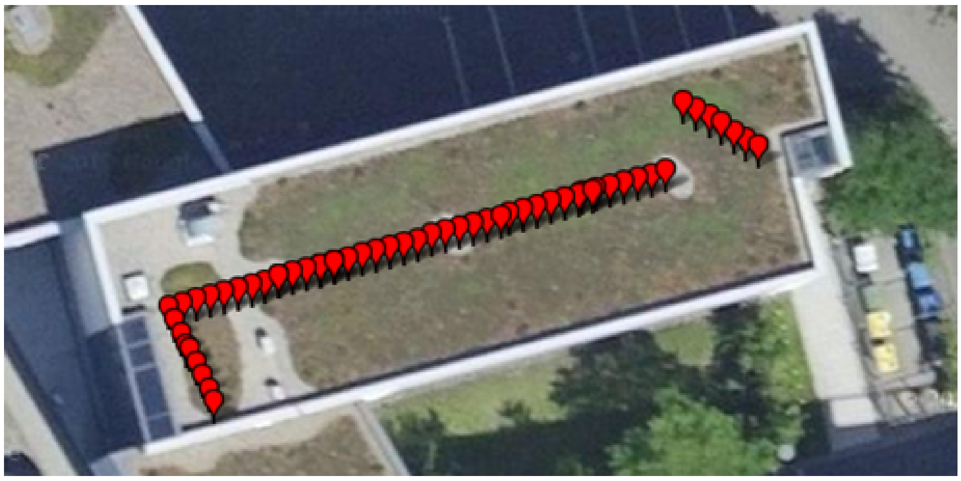

(c) Results of *compass correction* and *map-matching*/*map-aiding* features.

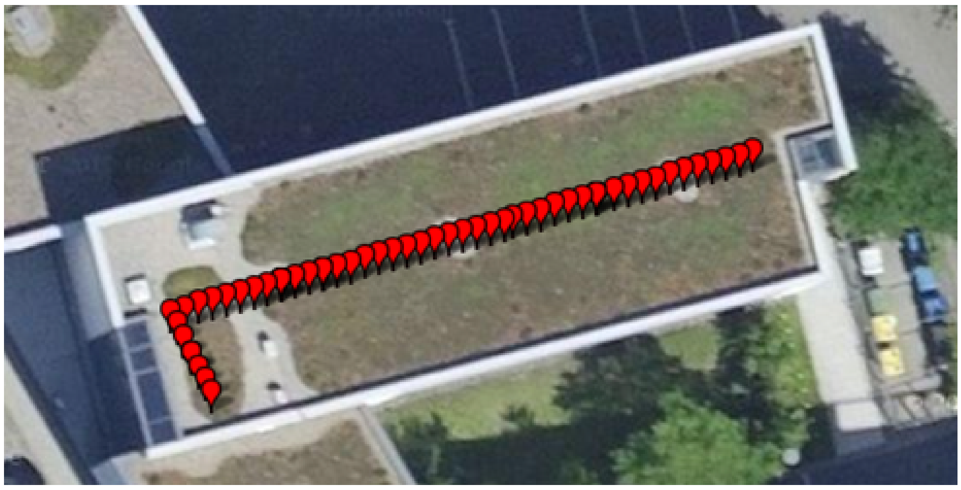
Figure A.8.: Results of the test configuration SIM-1 (initial heading error of 0°)



(a) Raw positions.

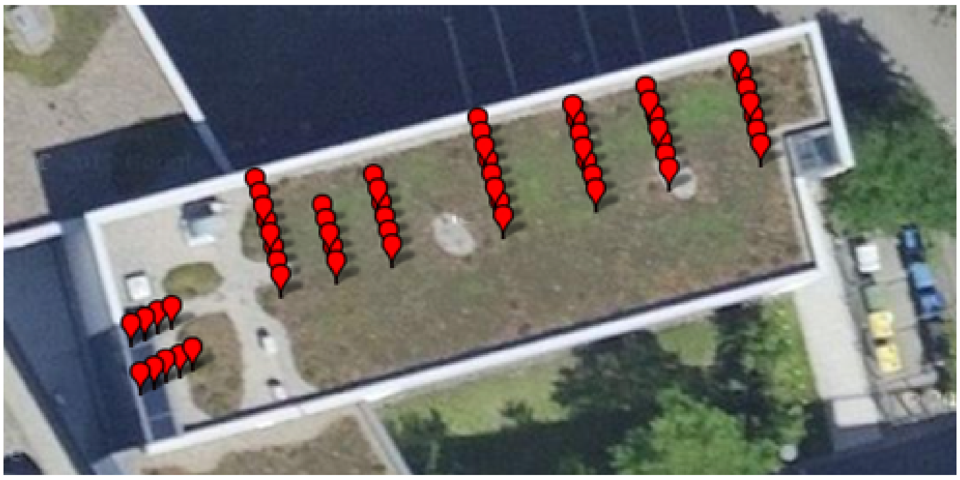


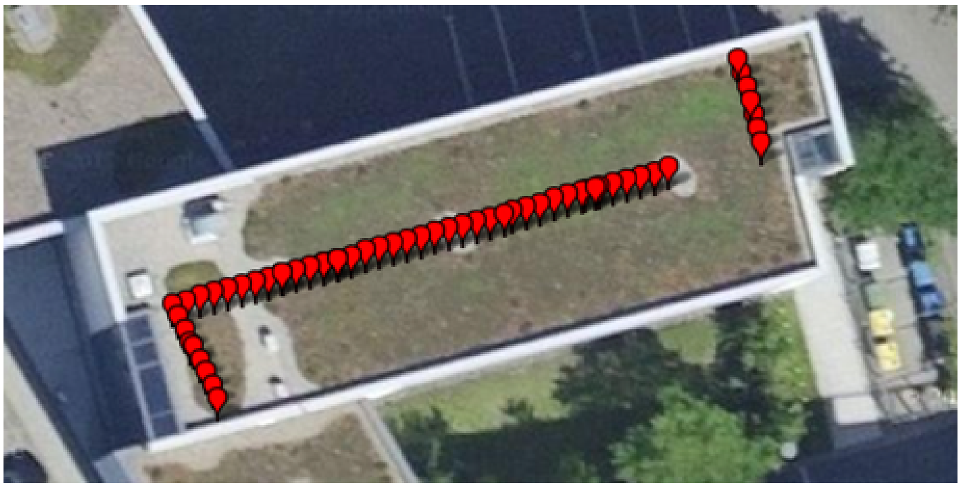
(b) Results of *compass correction* feature.



(c) Results of *compass correction* and *map-matching*/*map-aiding* features.

Figure A.9.: Results of the test configuration SIM-2 (initial heading error of 45°)


(a) Raw positions.


(b) Results of *compass correction* feature.

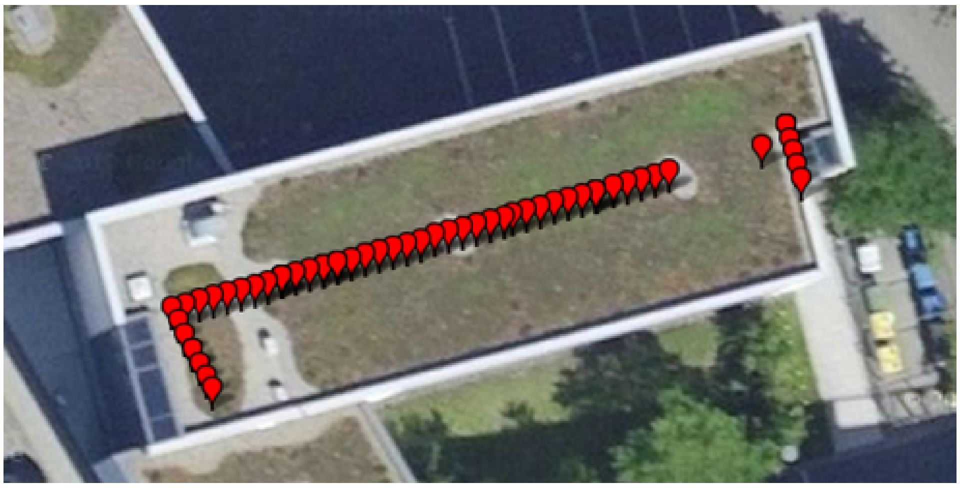

(c) Results of *compass correction* and *map-matching*/*map-aiding* features.

Figure A.10.: Results of the test configuration SIM-3 (initial heading error of 90°)

Declaration of Academic Honesty

I hereby declare to have written the Master Thesis with the topic

Indoor Navigation Based on a Multimodal Positioning System

on my own. All parts of this assignment which are cited literally or in a rough summary from publications or other secondary material are recognizable, and I have clearly defined them with their respective references.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich meine Masterarbeit mit dem Thema

Indoor Navigation Based on a Multimodal Positioning System

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, August 17, 2012

Björn Bittins