Universität der Bundeswehr München

Development and Implementation of a Peer-to-Peer Kalman Filter for Pedestrian and Indoor Navigation

Doktorarbeit
Institut für Raumfahrttechnik und Weltraumnutzung

Isabelle Krämer

der Bundeswehr
Universität München

Development and Implementation of a Peer-to-Peer Kalman Filter for Pedestrian and Indoor Navigation

Doktorarbeit

Von Isabelle Krämer
Institut für Raumfahrttechnik und Weltraumnutzung

7. Dezember 2012

INSTITUTE OF
SPACE TECHNOLOGY & SPACE APPLICATIONS

Acknowledgements

This work could not have been realized without the help of my current and former colleagues at the Institute. I want especially thank Dr. Thomas Pany who had an immense impact on the realization of my, at first, very basic idea of the P2P Kalman filter and Prof. Eissfeller who supported my process and gave me the opportunity to develop my own ideas.

A very special thank goes to my students Paul and Tobias who did a great job implementing parts of the prototype and devoting more time than required for this work.

For the constant encouragement to work on my thesis and to eventually finish it coming from Roland, Thomas and Victoria I was and still am very grateful. I also want to thank all my other colleagues for their help and support during my time at the Institute.

Without the support and encouragement of my family namely my parents, my brother and his family I would not have been able to realize this work. The most important person in my life – Christian - I owe most of my gratitude for his constant support and patience.

Abstract

Smartphones are an integral part of our society by now. They are used for messaging, searching the Internet, working on documents, and of course for navigation. Although smartphones are also used for car navigation their main area of application is pedestrian navigation. Almost all smartphones sold today comprise a GPS L1 receiver which provides position computation with accuracy between 1 and 10 m as long as the environment in beneficial, i.e. the line-of-sight to satellites is not obstructed by trees or high buildings. But this is often the case in areas where smartphones are used primarily for navigation. Users walk in narrow streets with high density, in city centers, enter, and leave buildings and the smartphone is not able to follow their movement because it loses satellite signals.

The approach presented in this thesis addresses the problem to enable seamless navigation for the user independently of the current environment and based on cooperative positioning and inertial navigation. It is intended to realize location-based services in areas and buildings with limited or no access to satellite data and a large amount of users like e.g. shopping malls, city centers, airports, railway stations and similar environments.

The idea of this concept was for a start based on cooperative positioning between users' devices denoted here as peers moving within an area with only limited access to satellite signals at certain places (windows, doors) or no access at all. The devices are therefore not able to provide a position by means of satellite signals. Instead of deploying solutions based on infrastructure, surveying, and centralized computations like range measurements, individual signal strength, and similar approaches a decentralized concept was developed. This concept suggests that the smartphone automatically detects if no satellite signals are available and uses its already integrated inertial sensors like magnetic field sensor, accelerometer, and gyroscope for seamless navigation. Since the quality of those sensors is very low the accuracy of the position estimation decreases with each step of the user. To avoid a continuously growing bias between real position and estimated position an update has to be performed to stabilize the position estimate. This update is either provided by the computation of a position based on satellite signals or if signals are not available by the exchange of position data with another peer in the near vicinity using peer-to-peer ad-hoc networks. The received and the own position are processed in a Kalman Filter algorithm and the result is then used as new position estimate and new start position for further navigation based on inertial sensors. The here presented concept is therefore denoted as Peer-to-Peer Kalman Filter (P2PKF).

One aspect of this thesis is to develop and implement a prototype of the P2PKF for an actual smartphone. But different issues have to be examined before being able to provide a useful prototype implementation. It is therefore first analyzed if there are already similar approaches regarding indoor positioning and cooperative positioning. Although indoor positioning is an important topic for quite a long time (maybe since the first mass-market receivers were sold) a satisfying solution could not be found yet. Only recently some approaches came up which addressed indoor positioning in the context of cooperative positioning strategies. These are examined as well as already existing approaches based on inertial sensors for pedestrian navigation with mass-market smartphones. A dead reckoning algorithm which detect steps, estimates the step length and the heading is introduced based on the work which has already been done in this area of application.

The next step is then to analyze the impact of the filtering algorithm on the position accuracy. Two variations of the standard Kalman Filter are examined and implemented to process simulations. Different scenarios are developed based on the variation of simulation parameters like the amount of peers, sensor accuracy and maximum distance to peers. These scenarios help to obtain the impact of the different parameters and of the filtering algorithms on the position estimation accuracy. Based on the results of these simulations a Kalman Filter algorithm can be implemented for the actual prototype.

The last step to complete the prototype is the choice of a suitable communication standard to enable the P2P communication. Essential aspects are the pervasiveness of this standard in common smartphones

and the ability to autonomously establish temporary connections between peers without relying on available infrastructure like access points. Existing standards are therefore examined on their fulfillment of those characteristics.

A prototype as it is intended for this work is composed of these three parts: navigation by means of inertial sensors, filtering algorithm, and communication. These three parts are implemented in a certain type of Java for Android smartphones. Each part is tested separately on its functionality and in combination with the other modules. At the end the whole prototype is examined in a real-life scenario. The results of these tests are analyzed and discussed in detail regarding issues like performance, functionality, and also power consumption.

Table of Figures

List of Tables

1. INTRODUCTION

The first mobile phone which was equipped with a receiver for satellite navigation for the U.S. NAVSTAR Global Positioning System (GPS) was announced in July 2001 in Germany [1]. Almost exact one year before in May 2000 the selective availability (SA) of GPS was turned off which enabled decimeter-accurate positioning by satellite signals also for civilian users where before the position accuracy was limited to around 100 m. This caused a boom in the production and sell of especially car navigation systems but also the integration of receivers into mobile devices like cell phones started. With SA turned on useful pedestrian or even street navigation was not an option due to the low position estimation accuracy.

However GPS's original deployment for military applications as intended by the Department of Defense (DOD) and the National Aeronautics and Space Administration (NASA) still is obvious especially when trying to use it for positioning in environments with obstructed access to satellite signals like narrow streets with high building density or within buildings. GPS works best in outdoor areas with free view to the sky. Here position estimations with accuracies below 10 m are possible and enable smooth navigation even for civilian users. However the deployment of GPS receivers also in mobile devices changed the consumer needs asking for continuous and seamless navigation regardless of the current environment and a fast access to a position estimate.

Several approaches to meet these needs already have been developed and some of them are addressed in the coming chapters. Especially techniques like assisted GPS (AGPS) [2] enabled the pervasiveness and acceptance of mobile phones as navigation devices and as a provider of location-based services (LBS). Nevertheless the navigation indoors and in urban areas still poses a problem for users. Most of the time a position cannot be estimated at all, the accuracy is too low to enable navigation or it takes much time to compute a position.

Modern smartphones not only comprise a receiver for satellite navigation but also inertial sensors which are able to estimate the heading and the step length of a user. The intention of this thesis is to exploit these sensors for pedestrian navigation and additionally suggest a method to stabilize the position estimation by means of cooperative filtering. Since the position estimation by inertial sensors especially the ones integrated into mobile phones is not reliable the idea is to improve this estimation by the estimation of other users' devices based on mutual filtering. This results in the development, implementation and test of a peer-to-peer Kalman filter (P2PKF) for pedestrian and indoor navigation.

The first section in this chapter gives an overview of the basic idea of this approach. This overview is later in this work addressed in more detail. Based on this overview the requirements for this approach are developed in the second section. The last section comprises the outline of this thesis.

1.1 Introduction to the Peer-to-Peer Kalman Filter

The idea of the P2P Kalman filter is that users help each other to improve their position estimated by inertial sensors in weak-signal areas or indoors by mutual filtering. The users which consecutively are also denoted as peers are equipped with state-of-the-art smartphones comprising a receiver for satellite navigation and inertial sensors enabling step detection, step length estimation, and the estimation of the heading of a walking peer. It is further assumed that on these smartphones an application is installed which automatically switches from position estimation via satellite signals to inertial navigation when signals become too weak to enable a useful navigation and vice versa when the environment changes in a way that enables the reception of satellite signals and supersedes the use of inertial sensors.

While the user resides in areas with limited or no access to satellite data the application computes the user's location based on the last known position estimated with the help of satellite signals, step detection, step length estimation, and heading estimation and provides a seamless navigation. The user must not be aware of the fact how the current position is determined and just continues to walk to the destination. Additionally to the computation of the position the application on the smartphones tries to find other devices in its surroundings via mobile communication links to exchange position data for mutual filtering. The intention behind this mutual filtering is that users have different devices enabling a different quality of position estimation by satellite signals but as well by inertial sensors. Devices providing more accurate position estimation indicate this by a smaller uncertainty regarding their current position estimation. This uncertainty is exchanged with the other peer as well and is integrated into the filtering algorithm and used as a weighting factor. This means that the position estimated with a more sophisticated device is weighted heavier than the position estimation of a device with less accurate position estimation. Based on this for both peers new position estimations are computed.

It is obvious that the user should not be bothered by the data exchange. As well as the application encloses if the position estimation is handled via satellite signals or inertial sensors it should also enclose the establishment of connections to other peers and the exchange of position data. The user only sees a continuous position estimation which can be used for location-based services, map matching and similar applications based on position estimation. It is of course mandatory that the user is currently using the device for navigation and enables the satellite navigation receiver, the inertial sensors and allows the establishment of connections via mobile communication links but the rest should be handled by the application autonomously.

The theory is that the more often such a position exchange is processed the better the overall position accuracy of all peers becomes. There might be cases where the position becomes worse when filtering it with the position of another peer but it is expected that this is only marginal and that the position accuracy becomes better the more peers take part in the cooperative filtering. Therefore this approach is adequate for environments where many users are in the same area with limited or no access to satellite signals but equipped with smartphones of different quality. These types of areas are e.g. large shopping malls, airports or train stations which would then be able to provide floor plans to the user without asking for certain equipment.

It is therefore necessary for the implementation of this approach to first determine the ability of today's smartphones to provide a position based on the measurements of their inertial sensors. If this is possible also the uncertainty regarding this position estimation can be determined. This can be obtained by experiments comparing the position estimated by inertial sensors with e.g. the position estimated by satellite navigation. It has to be tested if the assumption stated above that the position accuracy gets better the more users are available is true and with which filtering methods this can be achieved. Another question is also if a smartphone application is able to handle the navigation and the data exchange completely autonomously or if there are limitations. All these questions have to be addressed to be able to tell if such an approach of mutual assistance is valuable and realizable.

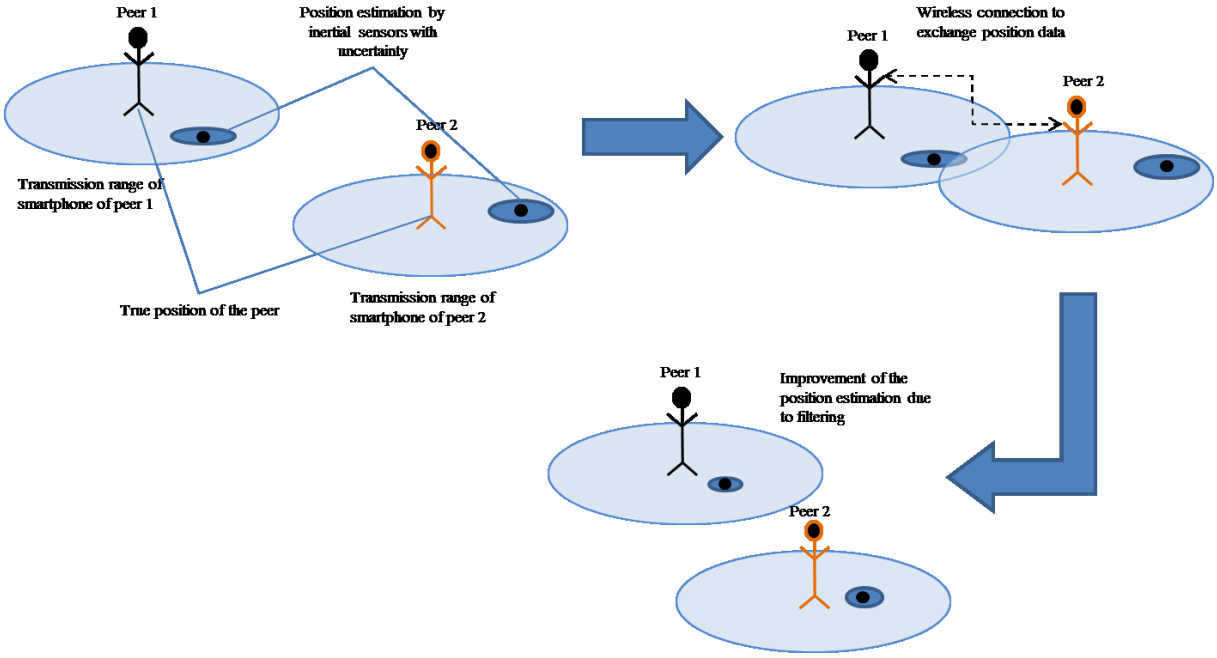


Figure 1-1: Schematic illustration of the idea of the peer-to-peer Kalman filter

In Figure 1-1 the basic idea of the Peer-to-Peer Kalman filter is depicted. By exchanging position data between two or more peers the own position is re-computed based on a filtering algorithm using the received position and its uncertainty as input. This results in a better position for both peers and a smaller uncertainty about the estimated position. It is expected that the results become better the more often the peers exchange position data with each other.

1.2 Requirements of the approach

The idea of this cooperative approach is to use position information which is available in the vicinity of a user to improve the own position accuracy and the position estimation of other users in areas with limited access to satellite signals. In contrast to methods which try to enable positioning in such areas no centralized entity is necessary to compute the position of a device or to transmit assistance data via the infrastructure of a mobile communication network. On the one hand this decreases the costly data links for the user on the other hand network traffic is reduced as well.

Additionally the P2P Kalman filter should be available in areas where no reception of satellite data at all is possible. At the same time position estimation with reasonable accuracy should be provided to enable location-based services in all kinds of different environments.

For the user there should be no visible difference between the navigation with satellite data or with the peer-to-peer Kalman filter. This means also that neither additional devices nor an intentional switch from one positioning strategy to another are necessary. The smartphone used for pedestrian navigation is a common off-the-shelf mobile phone enhanced only with the P2P Kalman filter application, a GPS receiver, and inertial sensors. Of course the device must be able to establish wireless communication links with other smartphones to exchange data.

The avoidance of additional hardware also includes a complex and time-consuming survey of buildings, the implementation of data bases, and the calibration of navigation devices. This does not mean that map data like e.g. for an airport cannot be provided by the owner of a certain building but the position used in such maps should come exclusively from the smartphone and not from hardware or software installed in this building.

In the list below the above described requirements are summarized. At the end of this thesis it is evaluated if all or only parts of these requirements could be addressed with the here introduced approach.

1. Navigation and positioning should also work deep indoors.

2. The user must not pay additional costs for data transmission of navigation assistance.

3. The P2PKF should be independent from centralized entities.

4. The mobile device is equipped with inertial sensors to estimate step length, step and heading and a satellite navigation receiver.

5. The mobile device must be able to communicate with other mobile devices within an ad-hoc network.

6. The P2PKF should enable a continuous positioning when moving from outside into indoors and vice versa. The survey of indoor area should not be necessary to enable positioning.

1.3 Outline

This works starts with an introduction of existing positioning and navigation systems based on different methods and techniques. The intention is an analysis about the adequacy of these systems to enable pedestrian indoor navigation with certain accuracy. It is explained why today existing approaches are still not able to provide seamless navigation from outdoor environment to indoor environment and vice versa. There are different solutions to overcome this problem and those are addressed in this second chapter. Nevertheless the draws of these solutions are depicted and based on those the possibilities of cooperative inertial navigation are developed. An important part in this chapter is also the clarification of the terminology which is kept throughout the whole work. The chapter starts after the explanation of the terminology with a short introduction to global navigation satellite systems (GNSS) in general and GPS in particular and explains the idea behind assisted-GPS. This is followed by a summary of positioning via communication networks and building infrastructure. Part of this chapter is also an introduction to "classical" inertial navigation and modifications necessary to enable inertial navigation for pedestrians based on smartphones equipped with inertial sensors like magnetic field sensor, accelerometer and gyroscope. A possible application enabling pedestrian dead reckoning (PDR) is introduced here as well. The last part of this second chapter is a review about existing cooperative positioning approaches and their performance.

The third chapter is the core of this work. Here the Peer-to-Peer Kalman filter is introduced in more detail than in the section above. Based on this description the filtering algorithm and possible variations are explained mathematically, as well as the dead reckoning algorithm and possible network standards which can be used to enable the communication of the peers among each other. These standards are examined regarding their adequacy for fast connection establishment and their communication range. Based on this section a simulation environment is developed to evaluate if the position accuracy improves when allowing cooperative filtering within an indoor area with limited access to GNSS signals. The different filtering algorithms which are identified in the section before are tested and compared to each other to find the optimal algorithm for the P2P Kalman Filter. This algorithm is then adopted for the implementation of the prototype in Java on an off-the-shelf Android smartphone. Each of the three modules of this prototype application is explained and analyzed in detail to find problems of this application. Also all three modules are tested separately to depict the performance of each part of the application. Different tests are carried out to analyze the dead reckoning algorithm, the filtering algorithm and the communication between the peers. In a last test the complete P2P Kalman Filter is evaluated.

This thesis concludes with a summary in the fourth chapter. The implementation of the P2P Kalman filter as an idea as well as the actual prototype implementation is discussed. Of main interest is the question if this approach is realizable as an application for real world scenarios. This analysis is based on the results from the complete test of the prototype application in the chapter before. Issues that have to be covered when implementing such a cooperative positioning approach especially regarding privacy issues, power consumption, connection establishment, and acceptance from the users in particular will also be part of this chapter. Different solutions to address these issues are examined and discussed. This chapter also gives an outlook on future work regarding the here introduced implementation of the P2P Kalman filter. Possibilities for further improvement of the existing prototype are discussed and analyzed regarding their impact on the approach.

The last chapter comprises the appendix. Here issues are addressed which are essential to some of the topics described in this work but are too complex or not directly linked to the topic addressed in this work. This applies to the description of the different coordinate frames and their transformation into each other. Although this topic is necessary to understand inertial navigation it is not directly related to the P2P Kalman filter. The basics of object-oriented programming are considered in this chapter as well to simplify the understanding of the implementation of the prototype application. Additionally here the list of abbreviations and the literature referenced in this work can be found.

2. PEDESTRIAN INDOOR POSITIONING AND NAVIGATION

The estimation of the own position relative to something or absolute has always been a crucial factor in all sorts of different scenarios ranging from navigation on sea over aviation to vehicle navigation and today even pedestrian navigation. While the computation of a position and the navigation to a certain target today even for non-military applications seldom is a problem it is still not possible with today's methods to provide a position with feasible accuracy within buildings or environments with obstructed view to the sky.

In this chapter different methods for positioning and navigation in general and pedestrian and indoor navigation in particular are presented. It starts with a short introduction of the terms indoor, outdoor, light indoor environment and weak-signal area to avoid misunderstandings. In this section also the expected position accuracy for indoor location-based services is discussed.

This is followed by a short introduction to GNSS positioning based on the example of GPS. In this section it is explained why standard GNSS positioning is not working when there is no direct line-of-sight.

The third section introduces the concept of assisted-GPS (AGPS). Most mobile devices today are equipped with an AGPS receiver enabling the reception of satellite data via mobile communication networks. The idea behind this alternative information channel is reviewed and based on this it is explained under which conditions GPS / GNSS signals can be received in indoor environment.

The current position of a user and her device is not only of interest for location-based services but also to enable wireless communication in mobile networks. Techniques enabling the position estimation of a user based on the infrastructure of the communication network are described in the fourth section. This includes position estimation based on time-ranging and the building infrastructure as well.

In the fifth section inertial navigation in general and inertial navigation based on pedestrian dead reckoning (PDR) are introduced. The differences and similarities between "classical" inertial navigation and PDR are discussed and current smartphones are introduced equipped with inertial sensors. An application enabling pedestrian dead reckoning for an Android smartphone is depicted.

The last section deals with concepts based on cooperative positioning. Since the P2P Kalman Filter belongs to this category as well it is of interest to introduce similar approaches working with P2P networks and cooperative assistance as well.

2.1 Terminology and Requirements

Before going into detail on existing localization methods for indoor environment the terminology has to be defined to clarify what the term indoor means to avoid any misunderstandings. Another important factor is also the accuracy which has to be obtained to enable location-based services also in indoor environments. Based on these requirements the existing indoor positioning methods described in the following sections are evaluated.

2.1.1 Terminology

For this work the signal availability is distinguished in three categories: outdoor, light indoor and deep indoor. Outdoor denotes a perfect view to the open sky with access to more than four satellites. A position can easily be estimated with assistance data or without; the assistance data only decreases the time to first fix (TTFF).

The terms urban or light indoor describe environments with very high building density and a limited view to the sky. Satellites with low elevation cannot be received nevertheless the PRN code of four or less satellites can be acquired while the decoding of the navigation message is not possible anymore. The receiver is dependent on assistance data providing the content of the navigation message. The shortfall of GNSS positioning is possible for short time intervals. The term LI is also used in [3]. Here it denotes an area where signals from three or less satellites can be acquired which means that a position cannot be computed. However in this work the term LI denotes an area where the receiver is not able to compute a position without assistance.

The expression deep indoor means areas where no line-of-sight can be established anymore between the receiver and any amount of satellites. Satellite signals can only be acquired by high-sensitivity receivers and a long incoherent integration time. Common off-the-shelf mobile devices with integrated GPS receiver are not able to provide a position.

The table below also categorizes all other possible terms to describe the signal availability as they are used in this work:

	Outdoor	**Indoor**	
		Light indoor (LI) [3]	**Deep indoor**
Terms	Open Sky (OS) Direct line-of-sight	Urban Canyon Weak-signal environment GNSS-challenged environment	Within building Non-Line-of-Sight (NLOS) GNSS-denied environment
Examples	Highways, fields, rural streets, etc.	Narrow urban streets, within buildings near windows and doors, buildings with glass roof, etc.	Within buildings of concrete away from doors and windows, cellars, underground stations, etc.

Table 2-1: Terms denoting the different categories of signal availability

2.1.2 Accuracy requirements for indoor positioning and navigation

The requirements for position accuracy for indoor location-based services (LBS) can be as diverse as for outdoor LBS. But in general there are limitations by the size of the building. If a service should work within an airport for example the required position accuracy of a LBS should be adjusted to the size of the airport and the distances between points of interest like terminals, check-in counters of different airlines, the ways to the security check and the gates. If only a position accuracy of e.g. 100 m is required by such a service the user has difficulties to navigate through the airport from the check-in to her desired gate. With a low accuracy of 100 m the user might only be able to tell in which terminal she resides currently but not if she is on the right way to the gate. Therefore a universally valid statement on the least necessary position accuracy for indoor navigation is not easily possible.

In the report on the E911 service [4] it is stated that the number of households using wireless communication instead of landline connections increased during the last years. The E911 mandate demands that the current position of a person calling the (U.S.) emergency number 911 can be obtained. This also means that compared to 2003 when only 40% of emergency calls came from indoors by mobile phones this number incremented to 56% today and will most likely continue to grow. An E911 compatible and certificated device has to provide position accuracy for network-based technologies of 100 m in 67% of calls and 300 m for 90% of calls. For handset-based positioning strategies accuracies of 50 m in 67% of cases and 150 m in 90% of calls have to be met. These accuracies can easily be obtained when the call is started outdoors and also for indoor calls the requirements seem not to be too hard. However mobile phones using a common GPS receiver without assistance data will have trouble to provide a position at all. In [4] it is also specified that indoor tests of E911-compliant devices have to be discussed in the near future but up to now no precise accuracy requirements only for indoors have been defined. But from the report it becomes clear that the FCC is aware of this problem especially regarding the today valid accuracy requirements which cover in urban areas several streets and blocks.

Another possibility to define requirements for accuracy of indoor positions is to use the same requirements as for outdoor LBS. Since the common GPS receiver provides accuracy between 1 to 10 m outdoors what seems sufficient for most LBS it makes sense to demand equal accuracy for indoor LBS. Assuming that a user in a shopping mall wants to know where the nearest ATM is a position accuracy of less than 10 m would make no sense. The goal should therefore be to obtain position accuracy indoors the same or even better as outdoors to be able to provide the same location-based services.

2.2 Satellite Navigation - GPS and GNSS

This section will give only a brief introduction to the basics of positioning with GNSS (global navigation satellite system) and in particular of GPS and explains why indoor positioning is not easily possible only with satellite signals. For a comprehensive description of satellite navigation please refer to [5], [6], [7], and [8].

The abbreviation GNSS comprises all global navigation satellite systems some of which are already in use for a long time like NAVSTAR GPS and GLONASS and some of which are still under development like the European Galileo or will reach Full Operability Constellation (FOC) in the near future like COMPASS from China. Although they all operate in different frequency bands the functionality principles are very similar for each of them. Their functionality therefore is explained based on the example of GPS.

2.2.1 Overview of GNSS

The development of NAVSTAR GPS started in the beginning of the 1970s as U.S. military application by the Department of Defense (DOD) and several other U.S. government organizations like the NASA (National Aeronautics and Space Administration) and the Department of Transportation (DOT). But the full constellation comprising 24 satellites was accomplished not until 1988. During the 1980s the use of GPS satellite signals was also made available for civilian applications. The GPS signals and codes are subject of ongoing modernization which started in December 2005 with the launch of the Block IIR-M satellite. It provides a second civil signal on the L2 frequency band additionally to the one on L1 used from the very beginning. Due to the use of two-frequency GPS receivers the largest error source the ionospheric delay is mitigated and enables higher position accuracy. All current GPS satellites sending the civil signal only on L1 will be replaced by this new kind of satellite by 2012. The next step which is planned to be finished in 2015 will be Block IIF signals sending a third civil signal on the L5 frequency band.

The development of GLONASS also started in the early 1980's by the former Soviet Union strictly for military purposes. It was full operational in 1995 but due to the collapse of the Soviet Union satellites which already reached their end of lifetime could not be replaced since founding was not possible and therefore the amount of satellites decreased to only 7 in 2001. The number of operational GLONASS satellites was again increased to 13 in 2006 due to a GLONASS modernization plan. In 2011 the 24th satellite of type GLONASS-M was launched in space completing the constellation again for the first time since 1996. The next generation of satellites – GLONASS-K – have an increased lifetime and provide a third civil signal in the L3 and L5 frequency bands this time using code-division multiple access (CDMA) modulation of the signals like Galileo and GPS. Before frequency-division multiple access (FDMA) was used. The first of this type of satellites was launched beginning of 2011 and more will follow.

In contrast to GPS and GLONASS the European system Galileo is a non-military satellite navigation system developed by members of the European Union and the European Space Agency (ESA). Although it was initiated already in 1999 it took until 2005 before the first test satellite GIOVE-A (Galileo In-orbit Validation Element) could be launched due to financing difficulties. It was planned to fund this project partly by governmental organizations and partly by private companies. Unfortunately this collaboration did not work and in 2007 it was decided to finance this project without private participation. The second test satellite GIOVE-B followed then in 2008. This finalized the planning and definition phase of the Galileo project. At the end of 2011 two satellites for in-orbit validation (IOV) were launched in space starting the installation phase which is the preliminary last phase of the Galileo system. Until 2014 the constellation should comprise 18 satellites providing already services. The full constellation with 30 satellites is expected to be operational in 2020.

At the end of 2011 the Chinese system COMPASS became operational initially as regional system for the Asian-Pacific region (China, parts of Russia, India, Japan, Indonesia, Australia and Antarctica). It is expected that COMPASS will obtain global coverage in 2020. The today existing augmentation system called Beidou consists of five geostationary satellites of which three were launched into space as test satellites (Beidou 1A, Beidou 1B and Beidou 1C) between 2000 and 2003. A fourth satellite was added in 2007. In this year also the first non-geostationary satellite COMPASS-M1 was launched expanding the formerly regional system to one with global coverage. In 2009 another geostationary satellite followed and from 2010 to 2011 several geostationary and MEO (medium earth orbit) satellites were launched into space.

The existence of more than one global satellite navigation system enables higher position accuracy for multi-frequency receivers since more satellites are simultaneously available. The reason for this is explained in 2.2.3. Nevertheless also with more satellites available indoor positioning based solely on satellite signals is not possible when certain position accuracy is needed.

2.2.2 System overview

The basic configuration is similar for all different systems: it consists of space segment, control segment, and user or ground segment. Since GPS currently is the most established global navigation satellite system it is deployed as an example to explain the functionality of the three segments.

The space segment comprises all satellites with respect to their constellation and the features of the satellite occupying each orbital slot. The constellation varies from system to system regarding the amount of satellites and parameters like inclination and distance of the satellites within one orbital plane. The GPS constellation consists of at least 24 satellites moving around Earth in six orbital planes. According to [9] 32 satellites are currently in space for GPS comprising also spare satellites to cover shortfalls. With respect to the equator the planes have an inclination of 55° and the orbital planes are equally spread around the equator with a 60° separation between them. The distance from the center of Earth to the center of mass of one satellite is approximately around 26,600 km. Due to their distance the time a satellite needs to orbit the Earth is approximately 11 hours and 58 minutes.

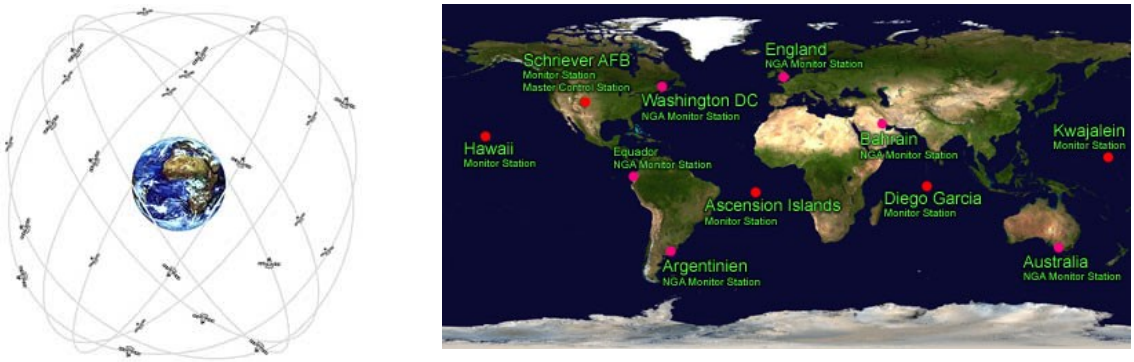


Figure 2-1: The constellation of GPS space segment (left) and the distribution of the control stations in the control segment (right) [10]

The control segment of GPS consists of one master control station (MCS) located in Colorado, USA and several monitor stations located around the world. The MCS controls the health and accuracy of the satellite constellation. It receives navigation information from the monitor stations and computes precise locations for the satellites from this information which is then uploaded to the space vehicles (SV). The function of the monitor stations thereby is to track passing satellites and collect atmospheric data, range/carrier measurements and navigation signals. To be able to provide useful data all monitor stations are equipped with professional GPS receivers providing very high position accuracy.

The user segment comprises all kinds of satellite navigation receivers regardless if they are used for car navigation, military applications or for pedestrian navigation within mobile devices. For GPS the

receivers can coarsely be classified in receivers for civilian use and in receivers for military use. At the moment civilian receivers only track the C/A code on the L1 frequency band but after the modernization of GPS there will be signals on L1C, L2C and L5. Furthermore receivers can be divided into their fields of application or market segments. The range goes from tiny mass market receivers that can be integrated into mobile devices costing only a few dollars to complete flight systems available for several tens of thousands of dollars.

Dependent on the type of application are the requirements of the receiver regarding position accuracy, time to first fix (TTFF) and sensitivity of the receiver. Nevertheless the configurations of a receiver for all these applications resemble each other regarding the basic modules even when they are implemented as software.

2.2.3 Positioning with satellite systems

To determine a three dimensional position using satellite signals the distance to at least four different satellites has to be estimated. The distance to one satellite is computed by calculating the time a signal needs to travel from the satellite to the user derived from the difference of the time the transmission started and the time the signal arrives at the user receiver times the signal propagation speed (speed of light c in space). The time of arrival can be read from the receiver clock and the time of transmission is contained in the navigation message coming from the satellite. As soon as the distance to a satellite is known from this computation the receiver resides somewhere on the surface of a sphere around the satellite with a radius the same length as the derived range. Since satellites are normally equipped with beam antennas pointing towards Earth it even only is a cone on which one end the receiver probably is. The ranges to two more satellites result in intersecting spheres in two points of which only one is likely to be the user position (at the surface of the Earth) [7].

The requirement for this scenario is that the receiver clock and the satellites' clocks are synchronized to each other since a deviation results in range measurements which are either too short or too long depending on the clocks being fast or slow. Since the satellites are equipped with high precision atom clocks mitigating also the relativistic effects their clock error compared to the GPS system time is rather small and additionally it is known by measurements of the control stations. The biases of satellites' clocks are transmitted within the navigation message and the receiver is able to balance those in its computations. What is not known is the clock error of the receiver. Although chip-size atomic clocks already exist [11] it is not common yet to integrate them into the standard mass-market receiver. To keep the time of a receiver a quartz crystal oscillator is used providing an accuracy of only approximately $2{,}5 \cdot 10^{-6}$ [12]. The fourth observation is therefore used to mitigate the receiver clock error. The ranges estimated are hence denoted as *pseudoranges*. To estimate the four unknowns – x_u, y_u, z_u and t_u - a set of equations is set up to estimate the *pseudoranges* to four satellites with *j* denoting the coordinates of one observation in an inertial frame (see 5.1.1).

The position of the satellite is encoded within the navigation message but not in inertial *x-*, *y-,* and *z-*coordinates as implied by equation (2-1) but as Kepler elements from which the receiver is able to compute the current position of the satellite.

$$p_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + ct_u \quad \textbf{(2-1)}$$

With

x_j, y_j, z_j	Position of satellite *j* expressed in *x*-, *y*- and *z* - coordinates
x_u, y_u, z_u	Position of the receiver expressed in *x*-, *y*- and *z* – coordinates
p_j	Pseudorange between receiver an satellite
t_u	Estimated transmission time at the receiver
c	Speed of light

2.2.4 Receiver functions

To understand why the use of GNSS signals for indoor positioning and navigation is not possible some basic issues regarding the signal design but also the processing of the signals within the receiver have to be addressed. Both are handled based on the example of GPS. Although the signal processing is very similar for most GNSS there are some fundamental differences regarding the signal design which makes the use of a concrete example necessary.

2.2.4.1 Signal design

Before addressing the signal processing functions of a receiver the signal design of GNSSs is shortly explained. A more comprehensive description can be found in [2] and [7]. The general built-up of the different satellite navigation systems is similar to a certain degree although all of them use different modulation schemes. But since to date most receivers are GPS L1 receivers it is sufficient to explain the signal processing using the example of GPS L1.

The carrier frequency of GPS C/A code is centered around 1575.42 MHz. In the satellite this frequency is produced as sinusoidal wave by an oscillator driven by the atomic clock. This carrier is then modulated using the satellite's unique C/A code forming a binary-phase shift keyed (BPSK) signal. The modulation with BPSK means that the carrier frequency is either left as it is or its phase is switched by 180° depending on the current value of the C/A code i.e. if a zero or a one is transmitted. This kind of radio interface access is denoted as CDMA (code division multiple access) instead of segmenting the limited radio interface by time (time division multiple access, TDMA), frequency (frequency division multiple access, FDMA), or space (spatial division multiple access, SDMA) each satellite has its own C/A code what makes it possible to distinguish them although they are all sending on the same frequency at the same time.

C/A codes have beneficial characteristics regarding their autocorrelation and cross-correlation properties (this is explained in more detail in the paragraphs below) and are also denoted as pseudo random noise (PRN) code. The C/A code is a code with a sequence length of 1,023 chips and a chipping rate of 1.023 MHz which leads to a repetition rate of 1 ms. The BPSK signal is then further modulated by the digitized content of the navigation message.

The navigation message is broadcast from each satellite every 30 s arranged into frames and further divided into five sub-frames. Sub-frame 1 contains parameters for the satellite clock correction; sub-frame 2 and sub-frame 3 comprise the satellite position in form of Kepler elements called ephemeris and sub-frame 4 and 5 transmit parameters for ionospheric effects correction and the almanc (simplified

orbit parameters for all satellites). At the beginning of each sub-frame is the telemetry word (TLM) used for data synchronization and the handover word (HOW) containing the most important parameter in form of the time of the week (TOW). If the GPS week is known in the receiver the estimation of the pseudorange can start right after decoding the HOW otherwise it has to be obtained from the first sub-frame.

2.2.4.2 Signal processing

At the receiver the signal as described above is received as radio frequency (RF) plus noise by the antenna. Within the front end this signal is down-converted to IF (intermediate frequency, f_{IF}) which is individual for each receiver but typically lies between 2 to 20 MHz. This process initally adds even more noise to the signal. The mixer removes the carrier frequency of the incoming signal by multiplying it with the matching carrier frequency produced by the oscillator of the receiver and leaves the binary sequence comprising PRN code and the navigation data. A complete removal of the carrier is only possible if the carrier frequency of the incoming signal matches the frequency of the receiver produced signal exactly.

The PRN code is removed by the correlation. A replica of the code stored within the receiver is applied on the incoming signal and if those are perfectly aligned a positive value for each multiplication appears. Those values are then summed (integrated) and the typical triangular correlation peak appears. This operation is denoted as coherent integration and results in a correlation peak computed from the integrated values and indicating the perfect code alignment. To keep the code replica aligned with the incoming signal the information from the integration is fed back to the correlator. Also the local oscillator receives feedback from the integration to keep the mixer aligned with the incoming carrier frequency. In Figure 2-2 the signal processing as described here is depicted.

The frequency feedback is necessary because although all satellites emit on the same carrier frequency due to the Doppler Effect caused by receiver and more severe by satellite movement (800 m/s) the carrier frequency (f_D) varies depending on the current constellation and the current setting or rising of the satellites. Also the local oscillator in the receiver normally has an offset compared to the true carrier frequency which has to be mitigated, f_e. This results in 8.4 kHz caused by the satellite's movement, 1.5 Hz for each 1 km/h the receiver moves, and 1.5 kHz of unknown frequency offset which are produced due to the oscillator offset [2].

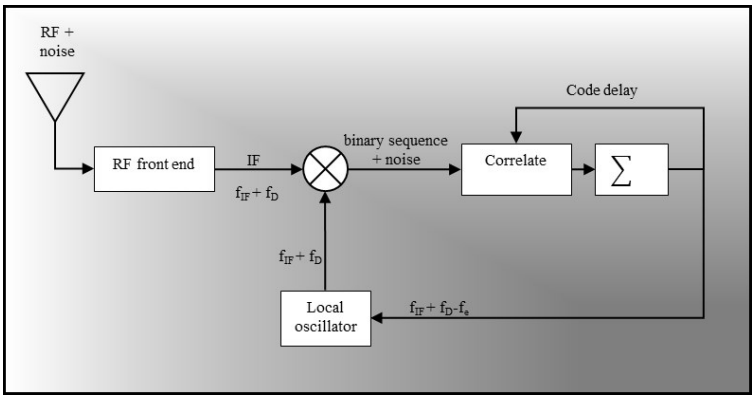


Figure 2-2: Generic signal processing within a GPS receiver [2]: The front end performs a down-conversion of the incoming signal to IF (intermediate frequency). The carrier signal plus the frequency caused by Doppler Effect is removed by the mixer driven by a local oscillator. Correlation aligns the local replica code with the code of the incoming signal and the integration produces the typical correlation peak.

The time delay between the code replica and the incoming signal enables the estimation of the signal propagation time in sub-millisecond part of the *pseudorange*. To compute the delay the replica code is shifted until both codes are aligned with each other. The receiver normally does not compute all possible

delays but rather two delays. One of which is aligned slightly earlier the other later than the received signal.

Before keeping the replica code and the local oscillator aligned with the incoming signal the receiver has to perform the signal acquisition. This means it has to search for the correct combination of code, code delay and Doppler frequency. It was already mentioned that PRN codes are chosen based on their beneficial characteristics regarding cross-correlation and auto-correlation. Cross-correlation means that one PRN code can clearly be distinguished from another PRN code during the correlation process and auto-correlation that one PRN code can be mapped explicitly to its replica. In a standard GPS receiver the frequency and code delay search is mostly implemented by dividing the unknown frequency search space in bins of several Hz and the unknown code delay in bins for early, late, and prompt code delays which are then examined to find the correct frequency and code delay combination of the incoming signal. The code-delay can range from zero to 1,023 (for C/A code) and therefore the dwell time within one frequency bin normally lasts 1 ms according to the code length. The total amount of time the receiver needs to find the right combination depends on the size of the frequency bins. In today's receivers this search can be performed for more than one incoming signal in parallel.

At the moment there are 32 satellites in space only for GPS. Both, the frequency search and the code delay search have to be performed for all satellite signals received by the antenna. Therefore a receiver which has no prior knowledge of satellites in view, code delay or Doppler frequency needs to search all possible combinations of PRN codes and Doppler frequencies. This scenario is denoted as "cold start" and depending on the constellation the time to first fix (TTFF) is at least more than a minute if no data bit errors occur while reading the navigation data.

A "warm start" can be processed when the receiver stored the frequency offset, the almanac and the last position during its last run and is turned on less than six hours later. The last known position is then used as starting point to compute the expected satellite position and Doppler frequencies. This information can then be used as starting point for the frequency/code-delay search. A warm start normally can be performed within 30 s.

The best situation to obtain a fast TTFF is when the receiver is able to perform a "hot start". This means that the frequency offset is well-known and the receiver time could be kept within sub-millisecond accuracy during off-time. Also ephemeris and time of the week were stored during the receiver's last run and it was turned off for less than 2 hours. A hot start typically can be performed in less than a second.

2.2.5 GPS / GNSS for indoor positioning

GPS is a military application designed to provide precise positioning in areas with unlimited access to satellite signals due to an unobstructed sky. The strength of received signals therefore does not need to be very high. According to the GPS Interface Control Document (ICD) [13] the minimum received signal strength using a 3-dBi (3 dB of gain with respect to an isotropic antenna) linearly polarized receiving antenna is around -128.5 dBm (decibel milliwatts). Theoretically this value should be maintained regardless of the satellite's elevation since a beam antenna pointing to Earth is used and the radiated power is stronger on the edges of the cone [2]. But this can only be obtained when assuming a perfect antenna and when the signal is not partially obstructed by obstacles like trees what of course occurs more often when the satellite has a low elevation. In reality the signal strength decreases the lower the elevation of the satellite is. After the signal processing in the front end the signal strength is denoted in signal-to-noise ratio (C/N_0, unit: dB-Hz) which approximately differs by 174 dB compared to the signal strength:

$$C/N_0 = \text{Signal strength (dBm)} + 174 - F_{dB} \quad \textbf{(2-2)}$$

F_{dB} is the front end noise figure in dB of the receiver making the signal-to-noise ratio also dependent of the receiver's characteristics.

While the relatively weak satellite signal can be obtained most of the time in unobstructed outdoor environment the received signal power is seriously attenuated when encountering an obstacle in form of buildings, trees or mountains. According to [14] the GPS signals are 20 to 30 dB weaker then outside. The received signal power behind an obstacle can be computed by:

$$P = P_0 - L_{total} \quad \textbf{(2-3)}$$

$$L_{total} = L_o + L_p + L_t$$

With:

- P Indoor received power
- P_0 Outdoor received power
- L_{total} Total attenuation
- L_o Outdoor attenuation:
  Attenuation caused by the additional transition through the layer (negligible when the layer is thin)
- L_p Permeate attenuation:
  Attenuation of signal amplitude caused by transition through the medium (dependent of dielectric constant of the medium, wavelength and thickness of medium)
- L_t Transmission attenuation:
  Attenuation caused by changes at the border layer of the media (also dependent of the dielectric constant of the media)

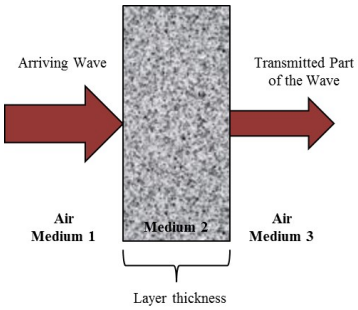


Figure 2-3: Transition of an electromagnetic wave through a dielectric medium [14]

In the table below the total attenuation caused by different kinds of material is listed [14]:

Material	Attenuation in L1	
	[dB]	Factor
Drywall	1	0.8
Plywood	1 – 3	0.8 – 0.5
Glass	1 – 4	0.8 – 0.4
Toned glass	10	0.1
Timber	2 – 9	0.6 – 0.1
Iron	2 – 11	0.6 – 0.08
Brick	5 – 31	0.3 – 0.001
Concrete	12 -43	0.06- 0.00005
Armored concrete	29 – 33	0.001 – 0.0005

Table 2-2: Total attenuation of different materials in L1 frequency band

This means that the statistical attenuation for residential houses is around 5 to 15 dB, for office buildings it is around 30 dB and for underground garages it is more than 30 dB according to [14]. Using the relation depicted in equation (2-2) the C/N_0 outdoors is ca. 44 dB-Hz depending on the receiver. Therefore the more the received signal power decreases the worse the signal-to-noise ratio gets which makes it hard to distinguish the received signal from the noise and to align it with the code replica. Even if the code and the frequency can be obtained the decoding of the whole navigation message which takes more than 30 s may not be possible due to losing the signal during tracking or already during acquisition.

Satellite signals are not only obstructed by material but they can also be reflected or diffracted at the surface of certain objects resulting in multipath. If the delay of the multipath signal relative to the direct path is large (i.e. two times greater than the spreading code symbol for BSPK modulation) the multipath is resolvable by the receiver [7]. As long as the receiver is able to track the direct path the effect of multipath is little on the receiver performance. More critical are multipath signals with very short delays caused by reflecting objects nearby which would be a common phenomenon within buildings or urban canyons. Such signals affect the correlation function and distort the phase of the received signals resulting in *pseudorange* and carrier phase errors different for each satellite.

Even more severe is the situation when the direct signal is partially blocked by an obstacle and the reflected signal is stronger at the receiver than the direct signal. Actually this is often the case indoors and also in areas with high density: the direct signal is partially or totally blocked by obstacles (walls, buildings etc.) while the multipath signal is reflected at the surface of those obstacles and reaches the receiver through a window. The error induced by multipath depends on the delay, the power and the carrier phase relative to the signals of the direct path. Since those errors vary from satellite to satellite (satellites with lower elevation normally suffer stronger from multipath than satellites with higher elevation) and are also time dependent due to receiver, satellite and possibly obstacle movement it is not easily possible to predict the influence on the receiver performance. In [7] more information can be found on this topic providing also a mathematical model for multipath. For the understanding of this work it is enough to know that the signal attenuation and multipath within buildings and also in areas with high density make it difficult or almost impossible to acquire enough satellite signals to provide a position.

2.3 Assisted GPS (AGPS)

The development of AGPS is directly linked to the E911 mandate which requires that a person calling the U.S. emergency number 911 is located with certain accuracy. While the location of the caller via a landline is relatively easy to determine this meant for the mobile communication providers they had to find a solution how to provide the position of a person calling from a mobile device with certain accuracy and within a certain time (see [4]). The embedding of GPS receivers in mobile phones was for a long time not common since receivers were still relatively large and heavy and also drained a lot power. During this time the infrastructure of mobile communication networks was used to determine the position of a caller (see 2.4). Due to progress regarding the chip manufacturing technology GPS receiver chips became small and power saving enough to be able to get integrated into mobile phones. Nevertheless the use of GPS only to obtain a position still asked for complex computation and was very energy consuming especially when performing a cold start. This pioneered the way for the development of AGPS.

Again GPS is deployed as example for Assisted-GNSS since it is up to now the system which is used in most smartphones for satellite navigation. Similar services for other GNSS are conceivable in the near future as soon as they have a greater pervasiveness in end user products.

2.3.1 Overview

The main idea of AGPS is to obtain the content of the navigation message and other data not directly from the satellite but via alternative mobile communication channels. In Figure 2-4 this principle is depicted:

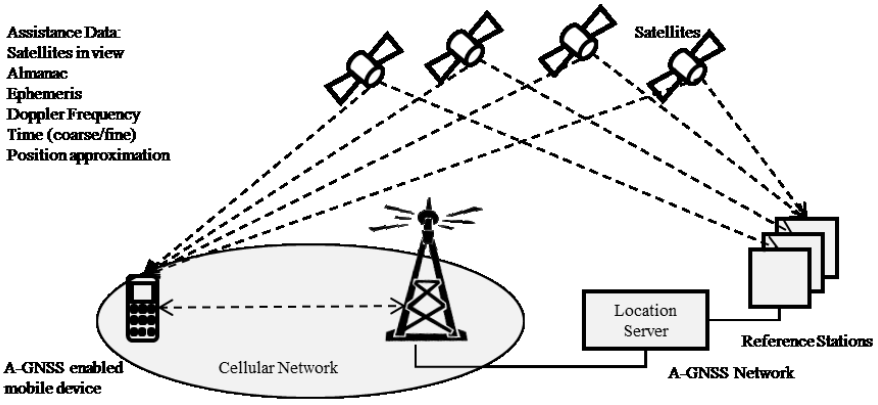


Figure 2-4: Overview of the functionality of AGPS: The mobile device is equipped with an AGPS enabled receiver and is capable of acquiring satellite data. To decrease the TTFF (time to first fix) or to enable positioning in weak-signal environments the mobile device receives additional information from the network provider via mobile communication links [2].

An AGPS enabled mobile phone receives navigation information in form of current satellites in view, the almanac, ephemeris of relevant satellites, the expected Doppler frequency, a coarse or fine time, and a position approximation from the provider of a cellular network. This data is collected by several reference stations equipped with GPS receivers with unobstructed view to the sky and sent from the location server of the network provider via the mobile network to the user. According to [15] the distance between the serving reference stations and the mobile phones must not exceed 100 km otherwise the information contained within the assistance data is not necessarily valid regarding the satellites in view and their Doppler frequency.

The assistance data affects the receiver performance in two ways:

- With free view to the sky the receiver is able to perform a hot start each time it is turned on since it has not to search all possible combinations of code delay and Doppler frequency but only those of the satellites in view. Also the decoding of the whole navigation message is not necessary because the ephemeris to compute the satellites' positions can be obtained via the mobile communication network on demand. Provided additionally with a fine time (maximum accuracy several µs) the time to first fix can be decreased severely.

- If the TTFF is not the critical issue but merely the positioning in environments with very weak signal power due to obstacles like walls or buildings the reduction of the search space allows a longer dwell time within the search space and increases therefore the receiver's sensitivity.

AGPS receivers normally can be used in two modes: MS-assisted and MS-based (MS is the abbreviation of mobile station and denotes the mobile phone or smartphone). In MS-assisted mode the position is calculated by the server while in MS-based mode the receiver computes its own position with help from a server. In Figure 2-5 the difference between those two modes is depicted. Nowadays most AGPS receivers are able to perform in both modes and what mode is used depends on the service for which the position is needed.

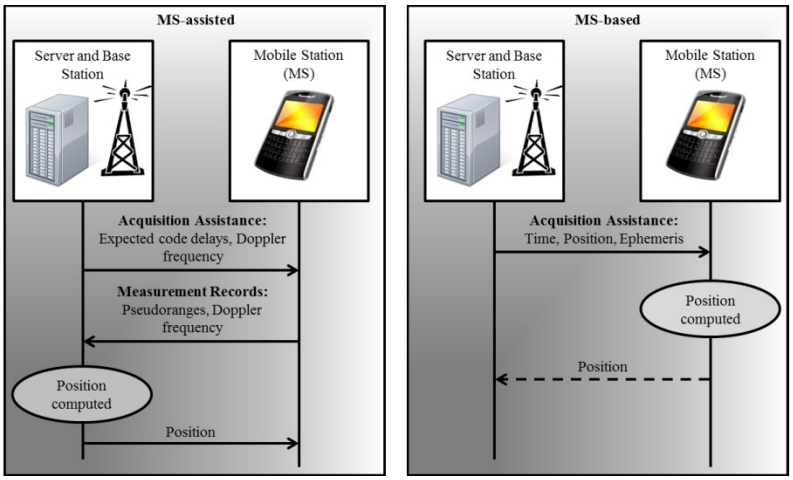


Figure 2-5: The differences between the two modes of AGPS devices

Errors in the assistance data have different impact on the performance of the receiver based on its current mode. These errors comprise frequency, time, and position errors and their effect is concisely explained in the following paragraphs. A detailed description can be found in [2].

For a MS-based receiver frequency assistance comprises time, reference frequency, position, almanac, and / or ephemeris. In MS-assisted mode the receiver obtains a reference time, reference frequency, and expected Doppler and Doppler rate. For frequency assistance the time accuracy only has to be in the order of 1 s according to [2]. For each second of time error the Doppler is wrong by 0.8 Hz (magnitude of change rate of Doppler frequency is 0.8 Hz/s depending on the satellite's velocity).

For mobile devices the frequency assistance is normally computed by the voltage controlled oscillator (VCO) of the device locked to the signal from the base station of the mobile network. The frequency of the base station is known with an accuracy of ±50 ppb (parts per billion) and within the mobile device ±100 ppb. Also the movement of the receiver adds up to the reference frequency since the VCO is driven by the base station: due to the movement of the receiver towards or away from the base station Doppler is generated as well and results in 1ppb for each 1 km/h [2].

Errors in the position assistance result in errors of the Doppler frequency since it is a function of time and position of the receiver. For a position error of 1 km a satellite Doppler error of approximately 0.19 m/s = 1 Hz (in the L1 band) is generated.

Errors due to almanac or ephemeris normally can only be caused by using almanac data instead of ephemeris since those provide velocity accuracy on the order of 1 mm/s which is equivalent to a Doppler accuracy of millihertz [2]. Although the use of almanac can cause in the worst case an error of 163.4 Hz (in L1) and 31.12 m/s respectively it is still possible to use this data for assistance since the error resulting from the less accurate almanac is uncorrelated across the different satellites.

Time assistance can also be used in the code delay search but it has to be accurate to microseconds. Since the C/A code repeats after 1 ms the receiver time has to be more accurate than 1ms otherwise the code delay cannot be estimated in advance. If the receiver time is not known better than 1 ms the position must be known to better than 150 km. A position error of more than ±150 km (300 km) results in more than 1ms code delay. In mobile CDMA networks which are currently common only in North America and Asia [3] a time accuracy between 10 µs and 1 ms (fine time) can be provided but in common European GSM and UMTS networks the time accuracy is only 2 to 3 s (coarse time) [2]. For fine-time assistance the MS-based receiver is provided with a time accurate to several microseconds, a position, and almanac and ephemeris; for the MS-assisted the data comprises fine time and the expected *pseudoranges* and rates.

An error in the fine time directly results in a code delay error (1 ms is equivalent to 1,023 chips for the C/A code). A wrong position leads to an error in the expected range and with that also in the expected code delay. In the worst case the upper bound caused by errors in the almanac or ephemeris is 0.8556 ms code delay error [2].

2.3.2 Indoor Positioning with AGPS

The deployment of AGPS enables positioning in light indoor areas where it is still possible to obtain weak satellite signals. Due to the transmission of the navigation message via terrestrial radio communication networks the message must not completely be received. It is sufficient if the PRN code can be identified all other necessary information can then be obtained from the location server. To enable the acquisition of weak signals different strategies can be used. Both are introduced in the following two sections and are based on extending the integration time (compare 2.2.4.2). As mentioned before if the TTFF is not the critical issue but instead the received signal power the search time can be prolonged.

2.3.2.1 Extending coherent integration time

The assistance information coming from the network can also be used to acquire weak signals. From [2] and [7] the following relation between the number of signal samples M_c and the ideal coherent gain is known:

$$\text{ideal coherent gain} = 10 \log_{10}(M_c)\ \text{dB} \tag{2-4}$$

Due to band limiting effects on signal and noise the actual coherent gain is defined by the following equation where $|\Delta|$ denotes the sum of band limiting effects caused by frequency mismatch, IF filtering loss, quantization loss and code alignment loss.

$$\text{actual coherent gain} = \text{ideal coherent gain} - |\Delta| \tag{2-5}$$

Nevertheless the more samples are used during the correlation the better the actual coherent gain becomes according to (2-4). The standard GPS receiver uses two samples per chip which means that for the correlation 2,046 samples in 1ms which is the standard coherent integration time are available. With assistance data this integration time could be easily extended. As described earlier the assistance data effects the positioning in two ways. On the one hand a faster acquisition of satellite signals can be performed and on the other hand the information can be used to search more intensively also for weak

signals in the search cells. The coherent integration time is therefore simply extended to process more samples within a longer time interval to obtain a larger correlation peak or a correlation peak at all for weak signals.

However the extension of the coherent integration time is only beneficial if the navigation message is known what is not a problem when using AGPS and fine time assistance which is the case in CDMA networks. Otherwise the results of several milliseconds of integration time cancel each other out due to phase changes. Those phase changes not only occur due to frequency errors and the un-modeled receiver velocity but especially due to data bit transitions. As mentioned in 2.2.4 the signal is modulated by BPSK resulting in a phase change each time the transition from a 0 to a 1 and vice versa is performed based on the content of the navigation message and the PRN code. If the navigation message is known and fine time assistance is available a so called data wipe-off can be performed and longer integration times are feasible. Other phase changing effects like frequency errors can then be mitigated by choosing certain time intervals for the coherent integration. In [2] the selection of a suitable time interval for the coherent integration time is described in detail. If the receiver velocity and heading can be modeled as well also this phase changing effect can be mitigated.

2.3.2.2 Non-coherent integration

In contrast to standard GPS / GNSS receivers so called high-sensitivity receivers (HS-receivers) have I and Q channels for each PRN code. In 2.2.4.2 the receiver modules are depicted in Figure 2-2; a HS-receiver looks almost the same only that the IF signal is "split" in its I (in-phase signal) and Q (quadrature signal) parts which are individually processed by their own correlators. Below the definition of the I and Q part of a signal is depicted [2]:

$$\mathrm{I}=d_k(t)\cos(\omega t) \tag{2-6}$$

$$\mathrm{Q}=d_k(t)\sin(\omega t) \tag{2-7}$$

With:

d_k	Digital part of the signal (PRN code and navigation message)
ω	Residual frequency error
t	Time interval of one chip

In HS-receivers the signal energy is therefore not lost due to phase changes but wanders back and forth between I and Q. The results of the the I- and Q-correlators are squared and added to obtain a correlation peak. The integration which is then performed after squaring is called non-coherent since the phase information is removed by the squaring. The squaring operation induces a non-linear squaring loss which causes correlation magnitude changes, a variation of the noise's zero-mean to non-zero and a change in the standard deviation of the noise. However the non-coherent gain is similar to the before mentioned ideal coherent gain [2]:

$$\text{non-coherent gain}=10\log_{10}(\mathrm{M_{nc}})\ \mathrm{dB} \tag{2-8}$$

$\mathrm{M_{nc}}$ is the number of non-coherent intervals. The non-coherent gain is a function of those non-coherent intervals instead of coherent samples as for the coherent gain. After the squaring operation the noise level is higher than when using coherent integration but after the non-coherent integration a correlation peak should be visible even in the presence of un-modeled frequency errors and long integration time.

This enables the acquisition and tracking of signals with a power of -150 dBm and even lower. Comparing this to -128.5 dBm (2.2.5) which has to be received by a standard GPS receiver the HS-receiver is able to obtain signals also in GNSS-challenged environments.

A more comprehensive description of acquiring weak signals with non-coherent integration can be found in [2]. Nevertheless signals have to be available with a certain power to perform non-coherent integration otherwise positioning is not possible. It is also very unlikely that the standard L1 receiver chip is exchanged by HS-receivers in the next generation of smartphones. Only the conversion of today's terrestrial radio communication networks to CDMA networks would enable the use of AGPS for indoor navigation.

2.4 Positioning based on mobile communication networks

In 2005 the U.S. Federal Communications Commission (FCC) instructed per E911 mandate that 95% of a network operator's in-service phones in the U.S. have to be E911 compliant [4], meaning that the position of the person calling via mobile phone can be obtained within a certain accuracy. A similar service (E112) is also planned for the European Community (EC). With the introduction of the 112 as universal emergency number Europe-wide in 2003 by a Directive [16] it was also demanded to provide the location of the calling user as soon as an emergency call is placed via a mobile device. At this time only few mobile phones had an integrated GPS receiver and let alone other possibilities to locate themselves indoors. Therefore the positioning was mainly realized by radiolocation techniques like Time Difference of Arrival (TDOA) or Time of Arrival (TOA) or by simply exploiting the cellular structure of mobile communication networks.

Which of these location techniques is used is dependent on the type of network structure. Still in use today is the GSM (Global System for Mobile Communications) network which belongs to the second generation (2G) of mobile radio systems and was the first system using digital voice transmission. In contrast to the today common UMTS (Universal Mobile Telecommunications System) the connection of a device to more than one base station is not allowed and therefore positioning methods like TDOA, AOA and TOA are not possible using this system. UMTS is part of the third generation (3G) of mobile radio systems and provides not only a larger bandwidth but also data services. The mobile device can be connected simultaneously to up to three base stations; a two-dimensional computation of the current position is therefore possible [17]. The modernization of mobile radio systems is continuously on-going and the systems of the fourth generation (4G) are already developed. In 2013 the first LTE-Advanced (Long-Term-Evolution-Advanced) enabled devices will be sold in Germany [18]. Despite the pervasiveness of CDMA networks in North America and Asia LTE-Advanced is based on FDMA and fine time synchronization between base station and mobile station which would be necessary for TOA is therefore not necessary.

2.4.1 Positioning based on network structure

To enable wireless communication in GSM and UMTS networks for each user network providers combine two radio access strategies. On the one hand they divide their whole network spatially in cells (SDMA) and on the other hand they use time division strategies (TDMA, in UMTS networks a combination of TDMA and FDMA) within those cells. The spatial division already provides the possibility to locate a mobile device with accuracy dependent on the size of the area, see Figure 2-6.

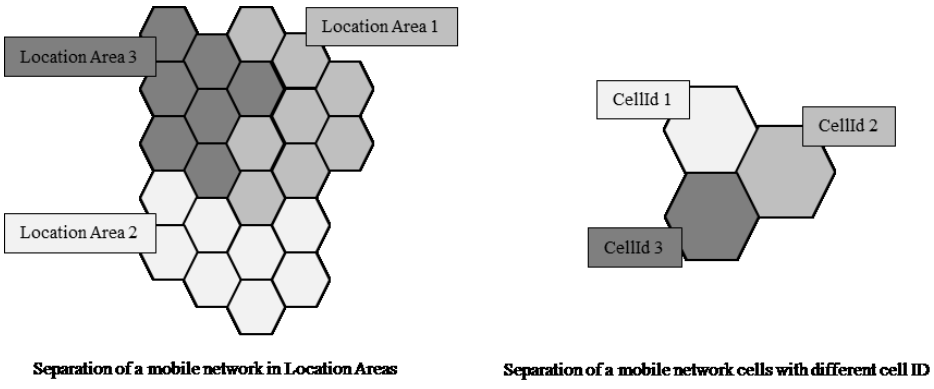


Figure 2-6: Separation of the mobile communication network of a provider in Location Areas (LA) and cells.

The network of a mobile communication provider is separated into Location Areas (LA) each of them having a unique identifier (LAI). One LA consists of several adjacent radio cells. Each cell has its own identifier, the CellId. Since the size of those cells is very small at least in urban areas not the current CellId of a mobile phone but its current LAI is stored within a database of the network provider to avoid

permanent database updates due to mobile station movement. When a call arrives, the mobile phone is paged within the Location Area, connects to the serving base station of its current cell and is now able to receive the call. Therefore it is relatively easy for the provider to estimate in which cell the mobile phone resides even if there is no incoming call. A phone which did not register for a certain time at a base station is periodically paged. The area a mobile phone resides in is therefore limited to either the size of a location area or the CellId is currently known [17]. Since a location area normally comprises several kilometers depending on the size and amount of cells only this information cannot be used to estimate the position of a user. But the information about the current location area can be used to provide a coarse position estimate with accuracy of better than 150 km for an AGPS receiver.

2.4.2 Positioning based on time-ranging information

Today's mobile communication networks like GSM and UMTS use a combination of TDMA, FDMA and SDMA to organize data transmission. To enable the exchange of messages within a cell between the mobile device and the serving base station have to be synchronized to a certain degree to each other. The messages are sent within a time slot assigned to a certain device. The farer the device is away from the base station the earlier it has to start its data transmission to be able to catch the right time slot. This method is called Timing Advance (TA) [17]. Based on this information the distance of the mobile device to its serving base station can be estimated. If additionally a directional antenna or an antenna array is used at the base station also the direction of the mobile device relative to the base station can be defined. Since the synchronization in this type of networks is only accurate to seconds which is enough for communication the estimation of the position cannot be used for LBS but as coarse position estimation for AGPS. The idea of TA used for positioning is depicted in Figure 2-7.

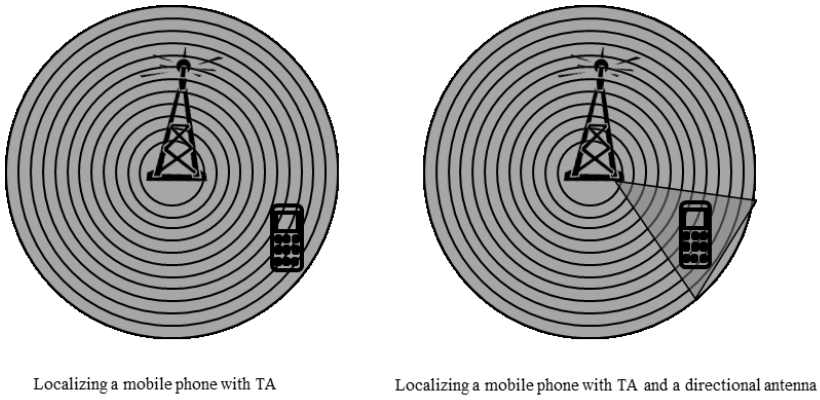


Figure 2-7: Localization of a mobile device exploiting timing advance (TA) and additionally a directional antenna

A directional antenna or an antenna field uses multiple sensors spaced by fractions of a few wavelengths and deploys a specific geometrical arrangement [19]. Based on the marginal time and phase delays on the different sensors the angle of an incoming signal can be obtained. This angle of arrival (AOA) is used to determine the position of a mobile device if its emitted signal can be received by at least two or better more base stations. Technically speaking this method does not deploy time-ranging information but is described here for the sake of completeness. In Figure 2-8 this positioning principle is depicted.

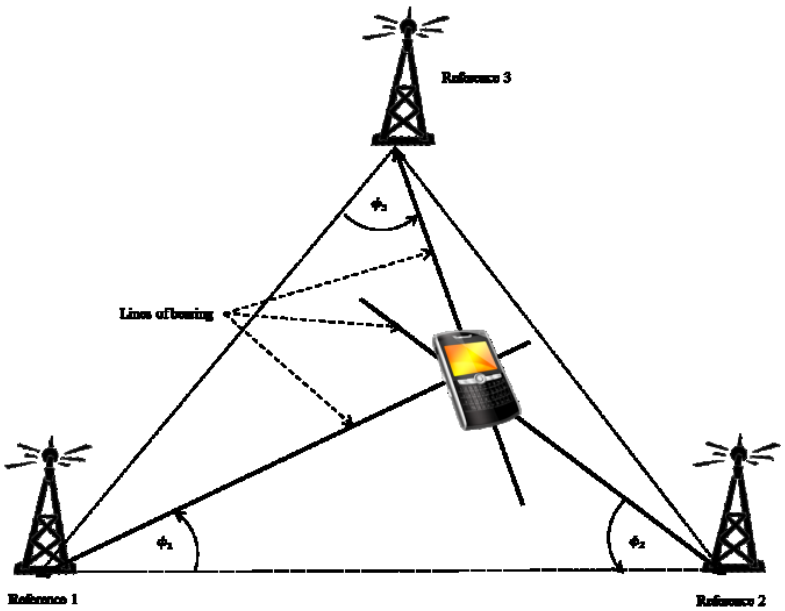


Figure 2-8: Principle of angle of arrival (AOA) [19]

The two positioning methods which are based on time information are time of arrival (TOA) and time difference of arrival (TDOA). The idea of TOA has already been mentioned in 2.2.3 when the positioning with satellite systems was described. As a matter of fact GNSS in general use the information about the time a signal travels from at least four emitters (satellites) to a receiver to compute the position of the receiver. The same principle can be employed for mobile communication networks. Since the altitude often is not crucial for land based applications three base stations are enough to estimate the position of a mobile device. As for GNSS the synchronization of base stations among each other and with the mobile device is necessary to compute correct ranges. As described above in GSM and UMTS networks the devices are synchronized to the base stations' time slots only to a certain degree (2 to 3 s) therefore more than three stations are necessary to obtain a more accurate position. It is very likely that in future networks the medium access will be organized as CDMA enabling synchronization accurate to µs. In Figure 2-9 the idea of TOA is depicted.

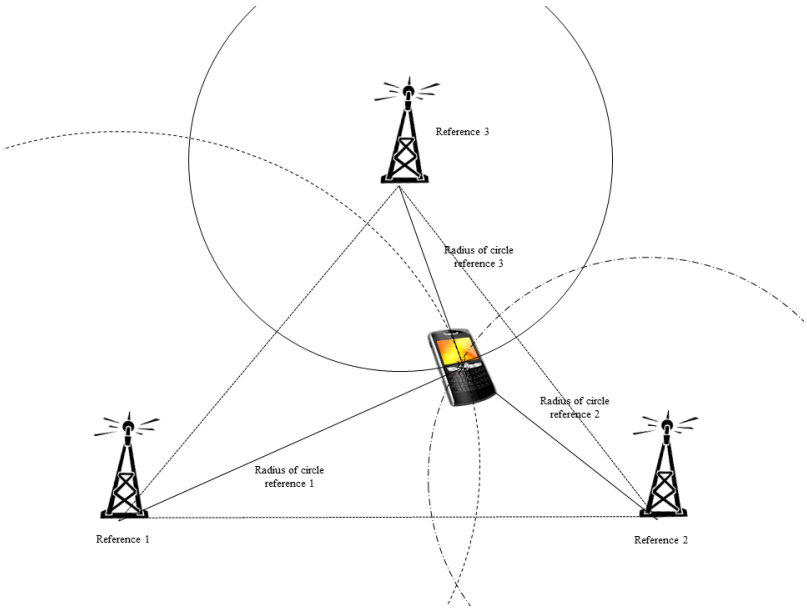


Figure 2-9: Principle of time of arrival (TOA)

To become independent of the exact synchronization between base stations and the mobile device time difference of arrival (TDOA) can be used to obtain a position. For this method only the base stations have to be synchronized to each other. TDOA is therefore proposed as mobile positioning standard by 3GPP which is a standardization organization for topics related to mobile communication. The position of the mobile device is then in contrast to TOA computed on the intersection of three hyperbolas and not on three circles. A device emits a signal to at least three reference stations simultaneously. A centralized unit of the network registers the difference in time the signal arrives at the base stations and computes in each case a hyperbola with the base station as center. The time when the transmission started is then not necessary anymore to compute the distance from one base station to the mobile device.

In Table 2-3 [19] below existing technologies are enlisted and the position accuracy which can be obtained. AFLT is the abbreviation for advanced forward link trilateration and measures time or phase delay from the CDMA pilot signal of base stations to compute the mobile device's position. The abbreviation EOTD means enhanced observed time difference and is the standard TDOA method used in UMTS networks. As can be seen from the table the accuracies of all location methods exploiting mobile communication networks are very low and are therefore not suitable for indoor location-based services. Additionally AOA asks for an unobstructed line of sight (LOS) to enable the computation of a position and TDOA and TOA have problems when dealing with multipath. Both situations are present in urban areas and indoors.

Method	Technology	Accuracy
CellID	All	100 m to 3 km
CellID with TA	GSM	500 m
AFLT	CDMA	50 – 200 m
EOTD	GSM	50 – 200 m
TDOA	All	100 – 200 m
AOA	All	100 – 200 m

Table 2-3: Positioning methods in mobile communication networks [19]

2.4.3 Positioning based on building infrastructure

These types of positioning methods rely on wireless communication links installed permanently within one building. WLAN fingerprint belongs to this kind of positioning as well as RF-ID tags.

WLAN (wireless local area network) fingerprint uses the individuality of signal strength of several access points at certain locations. A grid with accurately surveyed intersection points is created for the building. Several access points providing wireless network links based on the IEEE 802.11 standard [20] have to be installed. On each intersection point the signal strength of all receivable access points is measured and the signal characteristics (fingerprint) are stored within a database. The position of a mobile device is then computed based on the database. In [21] several methods for the position estimation by WLAN fingerprint are examined and their performance evaluated. Although the use of this approach provides usable results when database and intersection points are carefully surveyed and maintained these are also the most critical issues: The building has to be surveyed in advance and the calibration of the access points and the content of the database have to be constantly monitored. Additionally the position accuracy is dependent of the grid size. The larger the distance between the intersecting points the less accurate the position can be determined. But the nearer the intersecting points are to each other the less distinguishable are the signal characteristics from one point to another. A seamless navigation coming from outside to the inside of a building is also not possible to realize. Outside the building the user most likely uses GNSS navigation and has then to switch to another positioning method independent from the computed positions before. An existing example of this technology is Microsoft's RADAR with an accuracy of 3 m [19].

There are also several indoor positioning systems using signals coming from WLAN access points or Bluetooth reference points which can be received by a mobile device wirelessly to realize TOA or TDOA measurements estimating the position of the device. But the same draws as mentioned above remain: Only an individual solution for one building is possible. The building has to be carefully surveyed and seamless navigation from outdoor to indoor is not possible. Also those radio signals are refracted and shadowed by walls and surfaces as well. One technical realization for this type of localization is Aeroscout which uses the radio signal strength (RSS) of WLAN signals and TDOA [19].

Another way of positioning based infrastructure within a building is the use of RFID (radio frequency identification) tags. In contrast to all of the aforementioned positioning systems this is an active system. While systems like GNSS and TOA / TDOA with network infrastructure only provide signals containing the position and the transmission time of the sender and leave it to the user and the receiver respectively to determine their position, the positioning with RFID tags is active. The user carries a device with a unique code. As soon as the devices approach or pass a compatible reader fixed to a wall a door or a certain point of a building this code can be read via radio communication. Assuming that the readers are all connected with a server and the code of the device is uniquely allocated to a certain person the movement of a person within a building can be stored and queried from this server. Also this provides only an individual solution for a certain building and seamless navigation from outdoor to indoor is not possible as well. Additionally the user has to carry an extra device although many mobile devices already integrate such RFID modules. More severe is the fact that the movement of a person is stored within a central unit and could therefore be accessed not only from this person but also from third parties. In contrast to the aforementioned positioning solutions this might thread privacy issues.

As can be seen from the three sections above indoor positioning and navigation in GNSS-challenged environments is still an issue which is not satisfyingly solved. Often only individual solutions for certain buildings are available but do not provide seamless navigation from outdoors to indoors and vice versa. Since outdoor positioning is and mostly likely will be based on GNSS positioning a solution has to be found that supports GNSS positioning in areas with sufficient satellite signal coverage but is able to substitute the GNSS receiver when no signals are available anymore.

2.5 Inertial Navigation

Inertial navigation describes a form of dead reckoning (DR) navigation based on inertial sensors like magnetic field sensor, accelerometer, gyroscope, barometer, etc. DR uses the last known position to obtain the current position based on sensor measurements providing the heading and the velocity of an object. In contrast to the aforementioned navigation systems an inertial navigation system (INS) is completely independent of infrastructure like satellites or access points. The measurements of the sensors within one inertial measurement unit (IMU) are expressed in coordinates of the body-frame (see 5.1) which have to be transformed into a position of the inertial frame to enable navigation e.g. on maps. Today inertial navigation is often used in combination with GNSS to bridge situations where satellite signals are not available. Especially in safety relevant applications like aviation both systems profit from each other since GNSS provides long-term stability while INS provides short-term stability during outages. For these types of applications the combination of INS/GNSS is standard. But INS and DR can also be used for pedestrian navigation exploiting the functionality of state-of-the-art smartphones and their integrated sensors for navigation in GNSS-challenged and GNSS-denied environments. In the following paragraph the "classical" inertial navigation is described. More information on inertial navigation systems can be found in [5] and [22]. Based on this the differences to Pedestrian Dead Reckoning (PDR) and pedestrian inertial navigation are explained and a PDR for smartphones is introduced.

2.5.1 Overview

Using inertial navigation the heading and velocity estimation is accomplished by sensors: a gyroscope to measure the moving direction and an accelerometer to estimate the velocity. For both sensor types three to each other orthogonal sensors are arranged to measure the orientation and acceleration respectively in x-, y- and z-direction. A unit which consists of these six sensor axes and is connected to a body (e.g. a vehicle) is called an IMU (Inertial Measurement Unit).

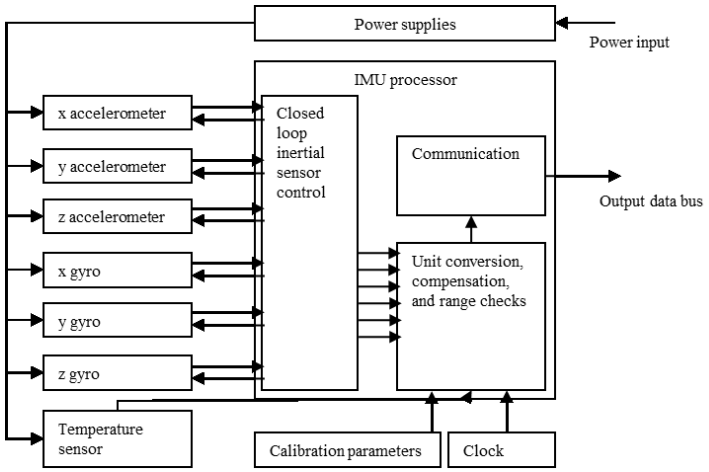


Figure 2-10: Schematic illustration of the elements of an IMU (Inertial Measurement Unit), see [22] and [5].

In Figure 2-10 a generic IMU is depicted. The six sensor axes are connected with a part of the IMU processor controlling the sensors and querying sensor measurements. The output of this controlling module is handed to another module performing unit conversion and range checks. The output of this module is then provided to the entity that handles the communication with parts outside the IMU.

Based on these measurements of accelerometer and gyroscope a new position can be determined. Since the output of the orientation sensor and the acceleration sensor are based on the coordinate system of the body, to which the sensors are attached to, those measurements have to be transformed into the inertial coordinate system to describe the location of the body uniquely (see 5.1 for information on coordinate frames). The algorithm to realize this transformation is not trivial and several possibilities

which are depending on the type exist. The today most conventional way to perform the transformation is a so called *strapdown* algorithm which is an analytical computation. Other algorithms use half-analytical systems (platform system) or geometric systems (deprecated). Since the *strapdown* algorithm is the most common today only the functionality of its computation and the attached processes are explained here.

2.5.1.1 *Strapdown* algorithm

The *strapdown* computation is an algorithm which defines the calculation of a position at the current point in time using the measured accelerations and orientations from the navigation solution of the point in time before. Thereby the computation consists of three steps: 1) propagation of the attitude by integrating the orientation rates, 2) propagation of the velocity by integrating the acceleration rates and 3) propagation of the position by integrating the velocity [22].

The basic equations of a *strapdown* algorithm are depicted below:

$$\frac{d^2\vec{r}^{i}}{dt^2} = \mathbf{C}_b^i \vec{f}^{b} + \vec{g}^{i} \tag{2-9}$$

$$\frac{d\mathbf{C}_b^i}{dt} = \mathbf{C}_b^i \boldsymbol{\Omega}_{ib}^b \tag{2-10}$$

With:

$\vec{r}^{i}$	Position vector in inertial frame coordinates
$\mathbf{C}_b^i$	Transformation matrix from body frame to inertial frame
$\vec{f}^{b}$	Specific forces at the body frame axes
$\vec{g}^{i}$	Gravity vector in inertial frame coordinates
$\boldsymbol{\Omega}_{ib}^b$	Matrix of angular rates in body frame axes

The difficulty of these two equations lies in the estimation of the transformation matrix and the current gravity vector. In equation (2-10) the position vector is computed using a transformation matrix based on the computation of the second equation. The matrix of angular rates is estimated using the measurements of the gyroscope. Those measurements are only available in the body frame and relative to the inertial frame. Therefore already here a transformation has to be performed to express the change of attitude of the body frame relative to the inertial frame in time. The new transformation matrix is then used for equation (2-9) computing the position vector by double integrating over the velocity and acceleration.

Vector $\vec{f}^{b}$ denotes the acceleration forces measured at the three axes of the accelerometer in body frame coordinates. The transformation matrix from equation (2-10) is used to transform them into the inertial system. Additionally the gravity vector has to be integrated into the computation. Since the accelerometer not only measures the linear acceleration of the body but also the gravitational acceleration affecting the body, this force has to be eliminated from this equation to be able to estimate the position of the body in the inertial frame.

The definition of the correct gravitational vector and the initial transformation matrix is a sensible issue and influences the total positioning process. If the gravitational acceleration is not completely eliminated

or too much is eliminated the error remains during the whole navigation process and produces serious errors regarding the position accuracy. The gravitational vector is a function of the position and has to be carefully processed to obtain useful position estimations.

Important is also the time synchronization between the different units of the IMU. Since sensor measurements are necessary to compute a new position they have to be available at this point in time. The result of the *strapdown* algorithm also is dependent on an accurate estimation of the time interval between two measurement updates to enable the correct evolution of the position estimation. A highly accurate clock is therefore necessary to obtain useful time intervals for measurement updates.

Another issue which has to be carefully handled is the initialization. Since inertial navigation is based on the computation of a new position using the position of a certain time before the starting position has to be determined as accurately as possible. Without this determination neither the first transformation matrix nor the current gravitational vector can be defined. The initialization therefore has to be performed using an absolute positioning system like for example GNSS measurements. However not only the position but also the initial attitude of the body relative to the inertial frame has to be determined before being able to navigate by inertial sensors. Every error that is employed during the initialization phase is present during the whole navigation process and intensified due to the integration. But not only the erroneous computation of the transformation matrix and the gravitational vector induce errors on the position solution, also biases of the sensors influence the position accuracy.

2.5.1.2 Sensors

Depending on the area of application, the costs and the space sensors can provide precise measurements which are necessary for example for flight systems, or low-quality measurements for non-critical applications. The sensors integrated into mobile phones are so called MEMS (Micro Electro-Mechanical System) which are produced very economically but also provide a very low measurement quality since they are sensible to noise. With MEMS micron-sized devices are denoted which are interacting with the physical world [23]. The manufacturing is inherited from the processing steps of basic integrated circuit techniques. Today it is possible to produce these devices for the mass-market: cheap enough and small enough for mobile phones or game controllers.

The first MEMS were already produced in the 1960s. Since then the number and functionality of those sensors are continuously growing. Today MEMS are integrated almost anywhere and used every day: gyroscopes used for image stabilization in cameras, pressure sensors in vehicles for releasing airbags, micro-mirrors for display projectors etc. Although the use of diverse materials for example glass or gold is possible silicon normally is the material of choice. It is stronger than steel and only has a third of its weight. It is brittle and not subject to plastic deformations what makes it ideal for micromachining technology. ”*Once combined with integrated circuits, electrical signals generated by the moving structures give perception and control capabilities to create sensors for a large variety of applications*” [23].

At the moment there are two different ways to combine sensing element and interface with each other: monolithic (single-chip, single-package) and hybrid (two-chips, single package). According to [23] the multi-chip single package solution provides better modularity and flexibility and is in addition more cost effective than single-chip single package solutions. These so called System-in-Package (SiP) solutions are also very common for inertial sensors. For the accelerometer for example the acceleration is translated into a differential capacity change and then converted by an interface integrated circuit (IC) chip into a digital or analog output. Another advantage of the SiP solution is that different sensors can easily be clustered with other standard chips like memories or micro-controllers.

To enable inertial navigation attitude and velocity have to be estimated and fed into the *strapdown* algorithm as described in the section before. The estimation of the angular rate is provided by a

gyroscope and / or a magnetic field sensor and the acceleration by an accelerometer. The following sections give an overview of the functionality of those sensors emphasizing especially MEMS.

2.5.1.2.1 Gyroscope

There are different ways to estimate the rotation of a body. The three most common ones are by spinning masses, optical gyroscopes and vibratory gyroscopes. The measurements of gyroscope MEMS are mostly based on the Coriolis acceleration to measure rotation which means that they belong to the class of vibratory gyroscopes. The Coriolis acceleration occurs additionally to the centrifugal force when a mass is able to move free within a rotating body frame. To illustrate the Coriolis Effect one pictures a rotating disk with a free movable sphere in the center of the disk. Due to the centrifugal force when rotating the disk the sphere moves to the rim of the disk. When watching this from outside of the disk the sphere rolls on a straight line to the rim but when watching this while standing on the disk the sphere's track seems curved. This is due to the fact that the observer resides within the rotating body system.

A gyroscope using the Coriolis acceleration is a vibratory gyroscope. A part of this type of sensor starts to oscillate as soon as the body is affected by rotation. This part could be a string, beam, tuning fork, ring, cylinder or hemisphere consisting of quartz, glass, plastic, and ceramic [23]. These strings vibrate at a very high frequency. When the body frame is rotated the Coriolis Effect occurs and adds an additional vibration in the perpendicular direction. The magnitude of this vibration is direct proportional to the angular rate and can be interpreted in rotation direction and angle.

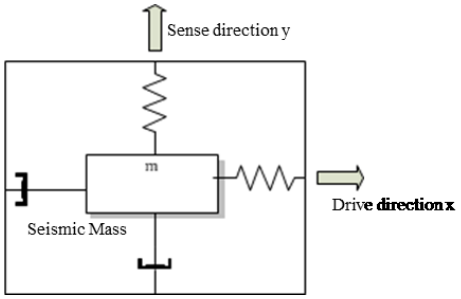


Figure 2-11: Vibratory gyroscope with a proof mass free to oscillate in two perpendicular directions [24].

Almost all MEMS gyroscopes have the same configuration: The proof mass is able to oscillate in two orthogonal directions, the drive direction (x-axis) and the sense direction (y-axis). The oscillation in the drive direction is induced by an external sinusoidal electrostatic or electromagnetic force. When the frame is subject to angular rotation the Coriolis force causes another oscillation in the proof mass orthogonal to the one in the drive direction; the sense direction.

2.5.1.2.2 Accelerometers

To estimate acceleration the basic idea is to measure the displacement of a proof mass which is able to move freely along the sensitive axis. There are two types of accelerometers: Open-loop accelerometer where the displacement is measured directly and Closed-Loop accelerometers where the force necessary to keep the proof mass in its original place is measured. Closed-loop accelerometers are more accurate than the ones using an open-loop approach especially when the proof mass is exposed to strong accelerations since the relation between displacement and acceleration then is not necessarily linear. Below is a schematic illustration of the principle of acceleration measurement.

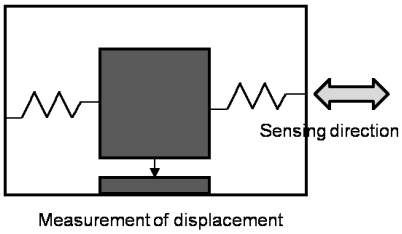


Figure 2-12: Basic principle to measure accelerations.

A simple MEMS pendulum accelerometer is built according to this principle. An elastically hung mass is etched from a silicon weaver and metalized on the upper and lower surface thus a dual capacity structure is built (see Figure 2-13, left). When the accelerometer is not subject to acceleration the capacitors have the same capacity. As soon as acceleration is introduced to the accelerometer the capacities change and the particular capacitor plate is charged to keep the proof mass in idle state. The magnitude of the acceleration can then be derived from the charging capacity.

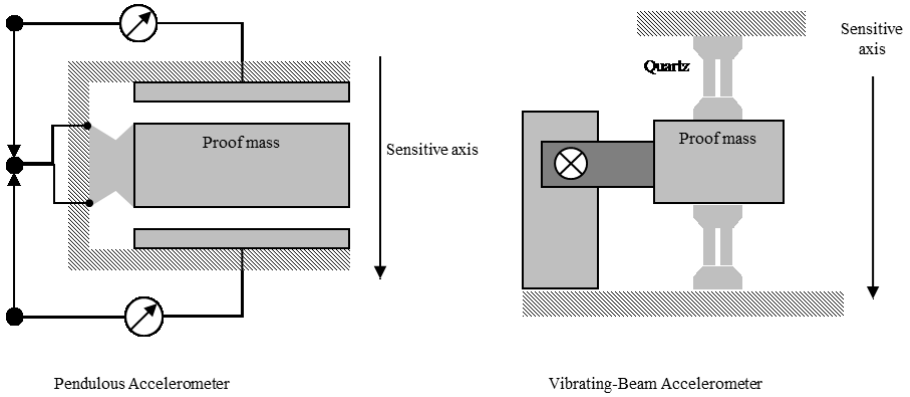


Figure 2-13: The schematic build-up of a pendulous accelerometer (left) and a vibrating-beam accelerometer [22].

Another way to measure acceleration is a vibrating-beam accelerometer. Here two quartz crystals oscillate with the same frequency when no acceleration is induced. When the quartz crystals are exposed to acceleration one of them is compressed by the proof mass while the other is stretched. According to this the frequencies of the quartz crystals is changed and the magnitude of acceleration can be derived from the beat frequency.

2.5.1.2.3 Magnetometer

To measure magnetic heading the Earth's magnetic field at the current position has to be determined. The geomagnetic field points from the magnetic North to the magnetic South through the Earth and takes the opposite path through the upper atmosphere. However the position of the poles is not fixed and they change their position slowly over time. The magnetic field is described by the magnetic flux density vector. The force per unit length due to magnetic inductance is the vector product of the flux density and current vectors [5]. The magnetic flux density is expressed in Tesla (T). Three parameters are determined. One is the magnitude of the flux density which varies from 30 μT at the equator to 60 μT at the poles. The second parameter is the inclination or dip angle which is the magnetic latitude within about 10° of the geodetic latitude. The third parameter – the declination angle - denotes the bearing of the magnetic field from True North and is needed to estimate the heading of a body from magnetic field measurements. The declination angle is a function of time and position and can be computed using global models like the Geomagnetic Reference Field (IGRF) or the U.S. / U.K. World Magnetic Model (WMM) [5].

A magnetometer measures the total flux density according to its sensitive axes (three or two axes arranged perpendicular to each other). There exist different types of magnetometers. The simplest and oldest one of them is the floating-needle magnetic compass where the heading has to be estimated by the user. An electronic compass measures the magnetic field and derives the heading of the body frame from its measurements. Magnetoinductive and magnetoresistive sensors are very small and can also be realized as MEMS providing an accuracy of 0.05 µT which is good enough for most navigation approaches according to [5]. Further magnetometer types include Fluxgate sensors or hall-effect sensors which are larger and more expensive.

There are many more types of sensors which can be useful for navigation applications like for example barometers measuring the air pressure and deriving the height. The sections above give only a concise overview to this topic. Especially in the last years the development of MEMS could benefit from the knowledge in chip fabrication and their size decreased as well as their fabrication price. This made it possible to integrate them also in small mobile devices like phones or gaming consoles. In 2.5.2.2 some of these devices are introduced.

2.5.1.3 Error characteristic

The accuracy of the estimated position by an INS is highly dependent on the initial position and attitude estimation of a movable object and the quality of the sensor measurement. Measurement errors, biases and inaccurate sensor alignment cause position errors. In contrast to measurement errors position deviations caused by misalignment of the sensor axes can be mitigated to some extent.

Errors caused by misalignment are based on sensor axes which are not perfectly perpendicular aligned to each other. Since the misalignment becomes already obvious during the manufacturing of the IMU they can be compensated during the operation by the processing algorithms. The misalignment of a sensor is denoted in a misalignment matrix comprising the scale factors on its main diagonal. The deviation of these scale factors from the ideal value 1 are defined as scale factor errors expressed in parts per million (ppm) and consisting of a constant scale factor and a non-linear part [22].

Also sensor biases comprise a constant part (ca. 90%) and a time-variant part (ca. 10%). The constant part of a bias is different each time the IMU is in use but stays constant during its use while the time-variant part (bias drift, bias variation) is variable during the use of the IMU. Causes for this drift are e.g. changes in the temperature [22].

In Figure 2-14 the effects of the constant scale factor error, the non-linear scale factor error and the bias is depicted. The scale factor errors have different magnitudes for the different types of gyros and accelerometers. For MEMS the total error can be as high as 10^{-2} [5].

Additionally the sensor measurements are affected by white noise caused by electric noise. The white noise is assumed to be normal distributed and zero-mean. To express the magnitude of white noise the power spectral density (PSD) or its root is used since the time-discrete variance of inertial sensors is dependent of the sampling interval. Therefore the units for the accelerometer are $\frac{\mu g}{\sqrt{Hz}}$ with $1\frac{\mu g}{\sqrt{Hz}} = \frac{1}{9.80665 \cdot 10^{-5}\frac{m}{s^2}}$ and the units for the gyroscope are $\frac{^\circ}{\sqrt{hr}}$ or $^\circ \cdot \frac{1}{hr} \cdot \sqrt{Hz}$ with $1\frac{^\circ}{\sqrt{hr}} = 2.909 \cdot 10^{-4}\frac{rad}{\sqrt{s}}$. It is not possible to calibrate or compensate white noise. Using the square root of the PSD makes it possible to obtain the expected rotation or velocity error within a certain time interval. Its standard deviation can be obtained by multiplying the root of the PSD with the root of the monitored time interval. Compared to the other types of angular rate sensors MEMS have the highest error caused by random noise with $1\frac{^\circ}{\sqrt{hr}}$. MEMS integrated in small electronic devices additionally suffer from white noise caused by the small space and the density of electro-magnetic sources in their direct vicinity caused by other units integrated into the mobile device.

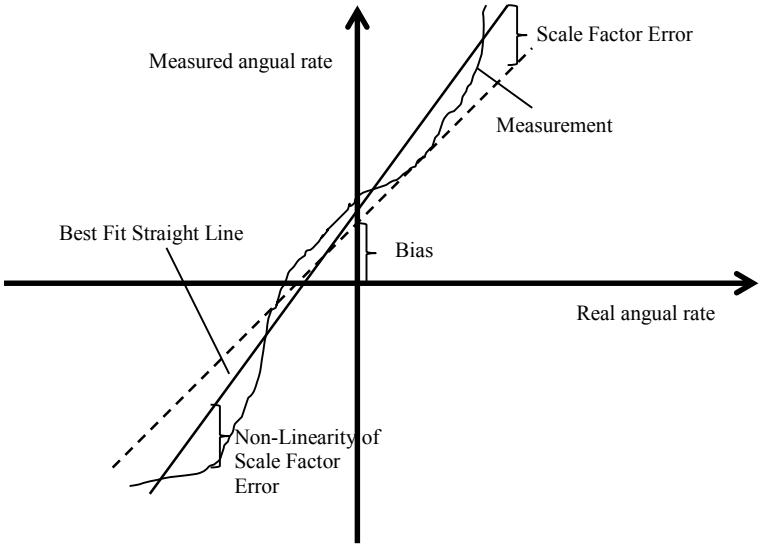


Figure 2-14: Illustration of bias, scale factor error and non-linearity of scale factor error [22].

Those errors make the use of INS as standalone navigation solution even for high-end applications difficult. However inertial sensors can act as completion for GNSS. While GNSS positioning provides long-term stability sensors can help out if no or only weak GNSS signals are available to provide continuous navigation to the user. Nevertheless without any correction of the position obtained by inertial navigation only the estimated position soon diverges strongly from the true position.

2.5.1.4 GNSS/INS integration

Since the different possibilities of the integration of GNSS/INS are beyond the scope of this work this section only gives a short overview of this topic for the sake of completeness. The integration systems here introduced are mainly used for continuous inertial navigation systems necessary for cars or aviation. For more information on this topic refer to [5] and [22]. The integration of inertial sensor measurements into a GNSS solution for pedestrian navigation is considered at the end of the next section.

To increase the accuracy, availability and the reliability of a navigation system it is common to combine two navigation systems which complete each other. As can be seen from the section above inertial navigation has draws when attempting to navigate in the long-term while GNSS provides well results in the long-term but has problems with short-term stability when no satellite signals are available due to shadowing. The combination of these navigation systems seems like a good match. Actually it is already common to use for car navigation GPS/GNSS positioning together with sensor measurements from the tires or the steering wheel to provide a better positioning and a position solution at all within tunnels or urban canyons.

The sensor measurement or the position solution of an INS is combined with GNSS measurements using a filter algorithm. In most of the cases this is a Kalman filter. An overview of the Kalman filter is given later in section 3.2.2. In general there are three different ways to combine INS and GNSS measurements: loosely coupled systems, tightly coupled systems, and ultra-tight or deep integrated systems.

Loosely coupled systems are the most common way to integrate GNSS measurements into INS since the integration is relatively easy and the development effort is lower than for the other two integration strategies [22]. The GNSS position and velocity solution of the receiver is used to support the INS measurements in the integration algorithm. Problems occur when no satellite information is available due to shadowing or signal interference and when the GNSS receiver uses internally another Kalman filter since the position and velocity solution might be affected by time correlations.

Tightly coupled systems integrate pseudoranges and *deltarange* measurements for the navigation filter. *Deltarange* measurement is the integration of *pseudorange rates* over an interval defined by the time expired since the last measurement. The *pseudorange rate* is a transformation from the Doppler shift measurements of a satellite to estimate the velocity of the receiver. In the literature the term tightly coupled also means that the GNSS receiver is supported and one talks of a closely coupled system when this support is omitted. The support of the GNSS receivers can take place during the signal acquisition but also during tracking. A tightly coupled system has the advantage that the support of the INS is also possible when the GNSS receiver cannot provide a position solution (i.e. direct line to less than four satellites). However the integration is more complicated since the navigation data has to be decoded to compute the satellites' position. The receiver is able to provide *pseudorange rate* and *deltarange* measurements as output but only based on its update rate. The transmission time is then not known and has to be iteratively obtained.

Systems using ultra-tight coupling or deep coupling have integration filters processing the I and Q samples of the signal and the tracking loops of the GNSS receiver are closed by the filter. The receiver and the INS are not able anymore to work independently from each other. The navigation solution provided by such systems is highly precise and is also able to tolerate a worse SNR of the GNSS signals. However the integration effort is very high and this type of system asks for deep changes in the receiver architecture.

2.5.2 Pedestrian dead reckoning (PDR) with inertial sensors

To realize pedestrian dead reckoning (PDR) there exist in general two ways to estimate a new position based on sensor measurement. One way would be to implement a PDR based on the classical inertial navigation by estimating acceleration (velocity) and heading of the pedestrian at certain points in time. Better than the classical double integration of the acceleration measurement as described in 2.5.1.1 is the exploitation of the human kinematics during walk [25]. This approach estimates the position stepwise. Therefore the sensors have to detect steps, the step length and the heading of the pedestrian. The dead reckoning algorithm for the second approach looks different than for common inertial navigation since the computation of the velocity is not necessary. Instead of that a step has to be recognized by the sensors and the step length has to be estimated. The estimation of the heading is similar to the common dead reckoning algorithm. Since it is measured in the body-frame it has to be transformed to the inertial frame to compute the direction of the pedestrian.

In the first part of this section the idea of pedestrian dead reckoning and inertial navigation based on step detection is introduced. The second part evaluates currently available smartphones and devices comprising inertial sensors.

2.5.2.1 Position estimation with smartphones

Although the development of a dead reckoning algorithm based on step detection is at the first glance easier than classical inertial navigation, two challenges remain: 1) while in planes, cars and ships the sensors are normally fixed at a certain and known position in the vehicle the position of a mobile device can constantly be changed. It can be carried in the hand, in the pocket, in a bag or in a backpack. Some approaches for pedestrian navigation work with sensors mounted at certain body parts like foot or head to estimate the heading or a step as described for example in [26]. At these locations on the body it is easier to estimate heading and step but it is very unlikely that this will become a common sight on the streets. And 2) the freedom of movement for pedestrians is much higher than for example for cars. While those can only drive in one direction on a street and turn on certain intersections, a pedestrian can almost walk wherever she likes. They can make sudden turns, stop or even walk the way back they came from. This makes is difficult to predict the track of a pedestrian and also the sensors have to adapt to such sudden movements.

Vehicles normally comprise measurement units which can act additionally as sensors for dead reckoning like the odometer, distance information from wheel ticks, a turn rate sensor, or a steering linkage angular sensor. The company u-blox for example produces GPS/INS systems for car navigation which can be integrated during the manufacturing phase of a new car or even afterwards [27]. Also the dimensions, weight, power consumption, and costs for in-vehicle sensors are not as critical as for handheld devices. Already available measurement units can be used and completed with more sensors. Due to the limited freedom of movement and well-established map-matching algorithms at least for car navigation inertial navigation for vehicles is more common to implement and able to produce better results.

At the University of Calgary several approaches enabling pedestrian dead reckoning (PDR) have been developed and analyzed. As mentioned before it is reasonable to exploit the kinematics of human walk and veer away from the classical *strapdown* algorithm dependent on the update time of the sensors. Rather than that the estimation of the new position has to be processed step-wise. In [25] it is suggested to use the accelerometer to detect a step, to estimate the step length and to propagate the position using a gyroscope, a magnetic compass, or a combination of both. Since a magnetic compass is very prone to any type of magnetic disturbances (steel, computer, etc.) which is especially indoors a huge drawback and the measured magnitude of the earth's magnetic field is very small it is a challenge to estimate the heading using only a compass. Therefore a gyroscope can additionally be used to detect disturbances using a filtering algorithm. But more promising is a highly accurate initialization. It is very likely that the navigation within a building already begins on the outside. This is also assumed for the Peer-to-Peer Kalman Filter. In [25] a position determined by Differential-GPS which uses correction data from reference stations to provide centimeter level accuracy is required for the initialization of the indoor navigation. The user's step length is either estimated by DGPS measurements or a specific RF foot-to-foot range measuring technique [28].

According to [26] three requirements are identified to estimate a new position for a walker:

1) Step detection

2) Step length estimation

3) Heading estimation

2.5.2.1.1 Step detection and step length estimation

Two general approaches exist when estimating the step length of a walking person: by biomechanical models or by algorithms based on empirical relationships [29]. In [29] four different approaches are implemented and compared with each other. Three methods are based on empirical data and the fourth on a biomechanical model of a knee-less biped. Here it is assumed that the mobile device is carried in the pocket of a trouser. The vertical displacement of the center of mass (COM) of the user is computed by double integrating the vertical acceleration during the ascending of the COM. Assuming that the leg can be considered as pendulum (knee-less) the step length can be computed from the deflection of the COM and the known length of the leg.

The empirical methods are mainly based on the average acceleration during one step divided by the amount of steps and using additionally a tuning parameter to calibrate the estimator. The different algorithms are compared with each other theoretically and with real data. In the experiments it becomes obvious that the main error sources are the accelerometer bias and the tuning parameter. The best overall performance is achieved with one of the empirical methods where the vertical acceleration at the foot during one step is correlated with the length of the step.

In [26] it is stated that the step length is correlated to the step period. Therefore an estimate for the step length can be drawn from the current frequency of steps. An accelerometer to detect a step is placed at the torso for this kind of measurement since the acceleration of the torso during one step is also an

indicator for the step length. According to [26] the individual step length of a person is subject to small variances around an otherwise very stable value. With sensors attached to the torso the step length can only be indirectly measured either by a mathematical model or by empirical methods as stated in the paragraph above. Generally speaking are the empirical methods superior to the mathematical models when the user's walking conditions are foreseeable. For measuring the stride length directly only a foot-mounted sensor can be considered. The stride length can be computed from the integrated acceleration measured at the foot. For this method it is necessary that the orientation of the sensor at the body is known. Additionally a gyroscope is necessary to estimate the angular velocity of the foot during one step since it is constantly changing during the gait cycle. The output of the gyroscope is used to define the attitude of the foot and combined with the accelerometer measurements the displacement can be computed. Another way is the use of parallel offset accelerometers measuring the angular acceleration of the foot and integrated twice to get the foot angle. According to [26] this measurement method can be used for all kinds of movement velocities from slow walk to running. The advantage of the direct measurement methods compared to the indirect ones is that those work without further assumptions of body height, gait or walking environment.

In [30] it is proposed to detect a step by using a peak detection algorithm applied to the filtered magnitude signal of the pedestrian's acceleration. The algorithm is based on activity detection (walking, standing still, and irregular movements) as introduced in [31] and attempts to localize maxima within a fixed interval of the signal. A minimum period between two steps is assumed based on the gait frequency of the pedestrian. An amount of samples of acceleration measurements is fed into the algorithm. A window with the size of the minimum period between two steps is slid along the samples; the median value is then computed within each window. A step is detected if one of the following conditions apply: when one median value is greater than the others, if the elapsed time since the last detected peak is greater than the minimum distance between two steps, or if the standard deviation of all samples in one window is greater than a defined threshold value.

According to [25] the errors in step length estimation are the largest error source for pedestrian dead reckoning. In [29] it is stated that all estimators tend to overestimate the step length.

2.5.2.1.2 Heading estimation

The estimation of the heading is a crucial issue in inertial navigation and especially for PDR. Although the step length estimation induces the largest errors the overall performance of the accuracy of the position estimation of PDR is limited by the heading estimation which causes a position error growth of third order. This is also one of the reasons PDR still is not a standard on mobile devices. The other reason as mentioned before is the fact that the sensors and the mobile respectively are not fixed on the body of the pedestrian. This makes it impossible to estimate the position with a three-axis gyroscope all alone. The use of a three-axis magnetometer measuring the earth's magnetic field is therefore necessary. Another issue is also the way the heading is defined. In [32] three different definitions for the heading are evaluated. The first one defines the heading as the angle between a chosen body axis and the earth fixed north axis. The second proposes the angle between the velocity vector in the l-frame (local level plane) and the north axis. The third proposition uses the angle between the vehicle frame's y-axis and the n-frame's north axis.

Since the mobile device can nearly be carried everywhere at the human body (waist, hands, bags, etc.) an additional transformation matrix to relate the b-frame of the device to the frame of the pedestrian is necessary. As long as this relation does not change during the navigation process the transformation matrix is constant but has to be re-computed when this relation changes. Also the human body is able to change its attitude compared to the earth-fixed coordinate system and with it the attitude of the sensor system is modified as well. If the sensor frame is fixed to a certain part at the body like the foot [26] or the waist [32] the attitude of the mobile device in relation to the human body is fixed and the transformation matrix between the b-frame of the sensors and the e-frame is well defined and the heading

can be computed according to the common equations of the *strapdown* algorithm (2.5.1.1) or by using a three-axes compass and accelerometer.

In [26] a PDR is suggested with foot-mounted sensors consisting of a three-axis accelerometer and three-axis magnetometer. To estimate the heading the direction of the gravitational field $\vec{g}$ and the magnetic field $\vec{h}$ has to be measured during the stance phase of the human gait. The heading is defined here as the pedestrian's direction in the horizontal plane and related to the axes of the n-frame (North, East, and Up). During stance the vector measured by the sensor estimating the gravitational field points down perpendicular to the local tangent plane of the earth. The output of the magnetometer is a three-element vector where the magnitude of the horizontal magnetic field H is defined by the sum of the magnetic field components B_x and B_y provided that the sensor is aligned with those axes in the horizontal plane and the alignment of the sensor is known respectively. The relation of the magnetic North to the True North (see 5.1.3) is denoted by angle D which can be computed from magnetic field models like for example the International Geomagnetic Reference Field [33]. The final heading angle ψ with respect to the True North can then be derived from:

$$\psi = \tan^{-1}\left(\frac{B_y}{B_x}\right) \pm D \tag{2-11}$$

The value of B holds the total magnetometer measurement and I the inclination angle between B and horizontal magnetic field H. Based on the measurements the coordinate frame X, Y and Z can be defined (see Figure 2-15).

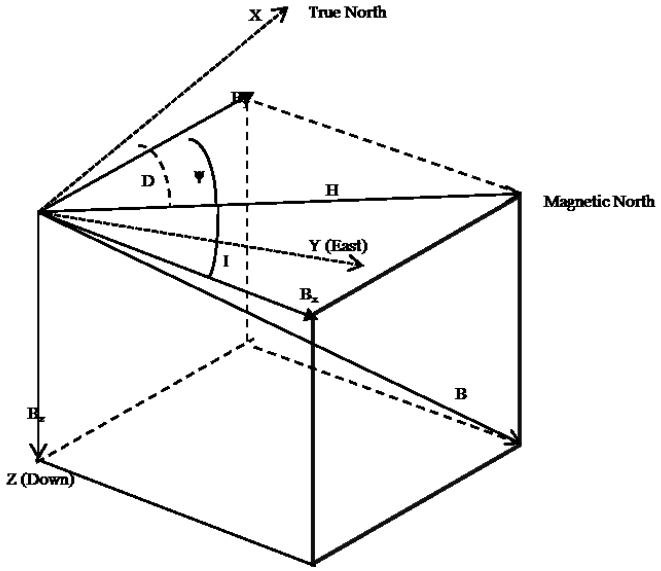


Figure 2-15: The axes of the vectors measured of the magnetometer and the relation to the true north [33]

As mentioned before the output of the magnetometer measurement is dependent on the known attitude of the sensors and the quality of the observations. As mentioned in 2.5.1.2.3 a magnetometer is subject to several disturbances caused by artificial magnetic fields induced by electrical power sources which are especially within buildings a crucial issue. In [33] an algorithm is introduced to detect erroneous magnetic field measurements nevertheless the use of magnetometers indoors is a problem.

2.5.2.2 Current smartphones

Although the navigation with integrated MEMS for pedestrian dead reckoning is still in its infancy the development is ongoing and is a promising approach for pedestrian navigation in GNSS-denied areas.

Also the accuracy of MEMS is increasing which makes them a more and more popular feature for all kinds of applications. The section below introduces some devices which are already on the market featuring gyroscopes, pedometers, accelerometers and even magnetometers.

2.5.2.2.1 MEMS in mobile devices

In 2006 Nike and Apple introduced together a Nike + iPod sport kit for running [34]. The idea of this kit is a sensor (piezoelectronic accelerometer) placed within the shoe (Nike) measuring the time the foot stays on the ground. This contact time is directly related to the velocity of the runner. The measurements of this sensor are then transmitted over a 2.4 GHz radio frequency protocol to a receiver in the iPod and analyzed. After training this data can be accessed and the runner knows the velocity and the length of track which was covered during the run. According to [35] the kit has an accuracy of over 90% without a special calibration. Nevertheless a calibration process to increase accuracy is possible. With this kit step estimation and step length estimation is possible but also here the sensor has to be mounted at a certain body location and it is designed to be applied only to Nike shoes together with the iPod. This limits the usage to a certain group and to a certain usage.

Figure 2-16: The iPod Nano with attached receiver (right) and the accelerometer (left) which is placed in the left training shoe (photo from wikipedia.de)

Despite being a portable gaming device the Nintendo 3DS introduced in March 2011 also comprises a pedometer. Every 100 steps virtual coins can be collected and used in games or in the Nintendo eShop. Although nowhere specified it is most likely that an accelerometer is used to define if a step was made or not. The accelerometer measures vibrations which occur when taking a step. This can also be measured when the device is not turned on and resides within a bag or a backpack [36].

Figure 2-17: The Nintendo 3DS (photo from wikipedia.de)

The Playstation (PS) Vita is a portable gaming device as well which is developed by Sony and was released at the beginning of 2012 [37]. According to [37] the PS Vita is equipped with a GPS receiver

and a sensor system consisting of a triaxial gyroscope, a triaxial accelerometer and a triaxial electronic compass.

Figure 2-18: Illustration of the PlayStation Vita (from wikipedia.de)

Not only gaming devices comprise nowadays sensor systems to enhance the gaming experience but also mobile phones are equipped with a GPS receiver and gyroscope, accelerometer and even magnetometers. The sensor system is not necessarily used for positioning but rather for the user to activate certain functionality by hand movement like rotating the screen or taking a call by shaking the phone. In contrast to most gaming devices the sensor data of smartphones is accessible by well-defined standard APIs (Application Programming Interfaces) in a well-known programming language like Java or C/C++.

2.5.2.2.2 Operating systems (OS) of smartphones

Nokia and Siemens were for a long time the only companies providing a way to write applications for their mobile phones. They made development kits available based on a special type of Java for mobile devices together with an emulator to test the developed software on a conventional PC. Siemens bailed from the mobile phone business in 2005 and sold its branch to the Taiwanese company BenQ which crashed one year later in 2006 [38]. Also Nokia was recently in trouble regarding its market share in mobile phones. The company had a market share of 36.4% in 2009 which dropped to 28.4% in 2010. Nokia always used their self-developed operating system (Symbian) for their mobile phones but due to the bad figures recently cooperated with Microsoft and introduced in 2011 a mobile phone equipped with Windows Phone 7 OS [39]. Also Microsoft has its own history of operating systems for mobile devices. In contrast to Nokia and Siemens Windows Mobile OS was never dedicated to the mobile phones of a certain company. The company HTC focuses mainly on Windows Phone OS but there are also Motorola and Samsung devices using this OS. The first version of Windows Mobile dedicated only to smartphones was introduced in 2003 and named Windows Mobile 2003 for Smartphones. In 2010 Windows Mobile 7 was introduced followed at the end of 2012 by Windows 8 which can be deployed on smartphones and tablets but also on desktop PCs. Compared to Nokia and Siemens Windows Phone OS can be regarded up to now as niche product since the market share is only at 1.5% in the third quarter of 2011. Also Microsoft provides a development kit for the implementation of software for mobile phones; it is based on the programming language C# [40].

Nokia and Siemens were global players in the past of the mobile phone market and Microsoft is currently still irrelevant in this market. Today the global players are Apple with their iPhone and the iOS and Google with their Android OS.

The first iPhone was introduced in 2007 and provided for the first time a touchscreen with multi-touch-functionality. In 2011 the fifth version of this phone (iPhone 4 S) was released [41] and since September 2012 the preliminary last version the iPhone 5 is available. According to [41] 27 Million iPhones were sold in 2012. The iPhone and different other mobile devices of the Apple company use iOS as operating system. The functionality of the mobile device is enhanced by so called Apps. App is short for application and denotes software which provides certain functionality. Some of these apps are pre-installed and part of the iOS like the contact list, email application, calendar etc. and some apps can later be purchased and installed from the iTunes store providing more than 700,000 (2012, [42]) additional

applications implemented by companies or by users. The user is able to choose the applications she wants and is able to adapt the functionality of the device to her needs. For the user it is possible to develop new apps based on the programming language Objective-C [43]. Objective-C is an enhancement of the programming language C and every program written in some form of C can also be compiled by an Objective-C compiler. Apple provides an API for the development of new apps; the necessary library and the development environment can be downloaded from the Apple Homepage.

Figure 2-19: The iPhone 4 S (photo from wikipedia.de)

In contrast to the iOS the Android OS developed by Google is not dedicated to the mobile devices of a certain company. Different manufacturers like Samsung, LG, Motorola, etc. provide phones using the operating system of Google. Android was introduced by the Open Handset Alliance whose main member is Google. It is based on a Linux-Kernel which makes Android a free software and open-source as well. The first device comprising this OS was the HTC Dream introduced in 2008. As well as iOS Android uses a touchscreen to receive input from the user. In October 2011 Android version 4 "Ice Cream Sandwich" was introduced and in June and November 2012 with Android 4.1 and Android 4.2 named "Jelly Bean" an updated version was published. The market share of mobile devices using Android OS was worldwide at 28% in 2010 and is 75% in the third quarter of 2012 [44]. As well as iOS Android provides the user with standard apps pre-installed on the mobile device and with additional apps in the Android Market. About 700,000 apps (Nov. 2012 according to [45]) are available to enhance the functionality of the mobile phone and each month about 40,000 new apps are published. Google provides a development kit for the implementation of new software on their homepage [46]. The necessary API is based on Java and provides several classes to access the basic functionality of the mobile phone like sensor data.

Figure 2-20: The Samsung Galaxy Nexus with the Android OS (photo from wikipedia.de)

The handling of both operating systems – iOs and Android – is based on the touch screen but also on sensing rotation and acceleration. Due to the rotation of the mobile device the screen is turned from portrait view to landscape view and vice versa. Several other apps take advantage of the built-in sensors be it the ShakeCall-Application from the Android Market or the iBeer app for the iPhone where the user can drink a virtual beer. Of course those sensors can also be used for positioning. To develop a prototype for the P2P Kalman filter three Android smartphones have been purchased and the quality of their sensors is tested. Android is chosen as the operating system of choice since it is an open-source

application and all elements can therefore be easily accessed. Since Android is not only available on one certain device the choice and therefore also the price range is larger than for the Apple product.

2.5.2.2.3 Sensor accuracy

It is extremely difficult to work out the sensors integrated into certain smartphones. Those details normally cannot be found in the specification of the mobile phone nor are the manufacturers willing to reveal the MEMS manufacturer or the model which is integrated. On inquiry all smartphone manufacturer declined to answer this question due to company policy. Therefore to be able to get an idea about the measurement accuracy of MEMS integrated into mobile devices it is easier to retrieve datasheets directly from the MEMS manufacturers. Going this way it is not possible to assign the different MEMS models to certain mobile phones but at least a statement about the average measurement accuracy of such kind of MEMS could be made. It has also to be taken into account that the measurement accuracy specified by the datasheets of different MEMS is not the measurement accuracy which can truly be achieved during operation. The measurements are additionally distorted by electronic sources within the mobile devices. It is very likely that most of the measurement noise is produced by the device itself and not necessarily dependent on the quality of the MEMS. It is therefore impossible to establish an error model as described in 2.5.1.3 for the sensor measurements to balance errors caused by misalignment of the axis, scale factor errors, and biases. Normally these figures have to be integrated into the filtering algorithm to weigh the measurements correctly. Since this information is not available for smartphones the error of the sensors has to be obtained indirectly by evaluating the quality of the dead reckoning algorithm (see 3.5.1).

2.5.2.2.4 GNSS/INS integration in smartphones

In Android devices the measurements of inertial sensors can be used to enhance the positioning via GPS. Since the operation of sensors is able to draw the power very fast this is an option which can be turned on or off by the user as necessary. How and which measurements are used to enhance the position accuracy is not communicated by the phone manufacturers. Also on inquiry it is not possible to retrieve this information. It is very likely that a Kalman filter is used to integrate the measurements of the magnetometer and accelerometer into the estimation of heading and velocity but detailed information about the algorithm could not be obtained. Since the use of sensors for positioning is optional it can only be a matter of loosely coupling as described in 2.5.1.4.

For the implementation of the prototype the integration of sensor measurements into the GPS position solution is not considered. As long as GPS is available it is used to estimate the position and as soon as no satellite data can be obtained the device switches completely to dead reckoning. It is of course possible to switch the sensors on as described above to enhance the position solution but this is not realized by the prototype application itself but has to be done by the user.

It is also possible to not use the sensor measurements directly but to integrate the position solution computed by dead reckoning into a filtering algorithm with the GPS position and use this output as position solution. Since the accuracy of the GPS position and the one of the sensors is known both position solutions can be weighted in this filtering algorithm. However normally the GPS solution is always much better than the dead reckoning solution due to the accuracy of sensor measurements therefore this additional filtering algorithms was not implemented into the prototype.

2.5.3 Dead reckoning application for mobile phones

In the scope of this work a bachelor thesis was prepared which main topic was the implementation of a dead reckoning algorithm for areas where the reception of GNSS signals is interfered due to so called in-car jammers and to locate those jammers. In-car jammers are devices which can be plugged into the cigarette lighter of a car and disturb the reception of GNSS signals within a certain area dependent of the strength of the emitted signal. They are mostly used to avoid fees on highways charged depending

on the track length of a car or truck, to disguise car theft and similar illegal acts. The situation is similar to indoor scenarios: The C/N_0 of the received signal is too low if present at all to be able to estimate a position at least with the standard receiver integrated into a mobile phone.

In [47] the step detection is based as described already above on the characteristics of the human walk and exploits the sensors already integrated into mobile devices. To detect a step and to estimate the step length measurements taken from the accelerometer are used while the heading of the person is computed from the magnetometer. As described already in 2.5.2.1.2 the measurements of magnetometers near electric sources which is especially indoors a problem strongly differ from the true heading and have therefore a severe impact on the quality of the position estimation. To overcome this problem an improvement of the algorithm of [47] is currently in process using the gyroscope measurements for heading estimation.

The basic scenario of [47] is a person using a mobile phone for GPS navigation approaching a static jammer. The jammer emits an interfering signal affecting an approximately spherical area with the jammer as center. The intensity of the interfering signal increases with each step of the person towards the jammer and in the same extent the received signal strength within the phone's receiver decreases. As soon as the C/N_0 of the received satellites degrades due to the jammer the phone switches to step detection and step length estimation to enable seamless navigation. To locate the jammer exactly the person carrying the phone is now asked to continue to walk in a straight line which is ensured by the magnetometer measurements. During this time the C/N_0 of the received signals is continuously measured and the position with the lowest measurement is computed. The user is then asked to change her heading to 90° relative to the original heading and walk in a straight line. Also here the C/N_0 is measured and the position with the lowest measurement is computed. Based on the two position computations the location of the jammer can be estimated.

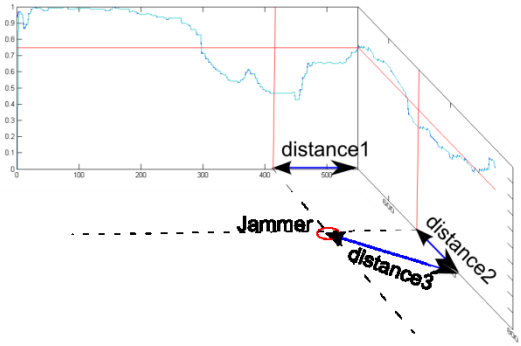


Figure 2-21: Illustration of the principle of jammer detection and localization based on pedestrian dead reckoning as introduced in [47].

The implementation of this dead reckoning algorithm is used with small modifications for the implementation of the prototype in this thesis. The step detection and step length estimation of [47] is mainly based on the work of [48] assuming that a step can be recognized by certain patterns in the measured acceleration. According to [48] a step consists of three phases: the support phase, the swing phase and the heel-touch-down phase, which can be distinguished by different acceleration measurements.

In contrast to [48] where the device is fixed and foot-mounted it is assumed that the user carries the device in the hands to be able to navigate. It is therefore sufficient to measure only the vertical acceleration instead of the vertical **and** the acceleration along the longitudinal side of the smartphone as proposed by [48].

The support phase is part of the calibration and processed each time the mobile phone is used for navigation by dead reckoning. In this phase 100 measurements of the vertical acceleration are taken and

the average computed while the user is standing still. This average vertical acceleration is used to detect a step in swing and heel-touch-down phase.

As soon as the user starts walking after the support phase one feet swings forward causing an increase of the vertical acceleration above the before computed average. It reaches a certain maximum and then again decreases until it has the same value as the average vertical acceleration. At this point the swing phase ends and the heel-touch-down phase starts.

The user hits the ground with the foot resulting in further decrease of the vertical acceleration until a minimum is reached and the vertical acceleration rises again until it has the same value as the average vertical acceleration. This indicates that a step has been detected.

Swing phase and heel-touch-down phase take place within certain time intervals which have been empirically defined in [48]. As soon as all conditions - 1) Recognition of the swing phase, 2) Swing phase within a certain time interval (150 to 300 ms), 3) Recognition of the heel-touch-down phase, and 4) Completion of both phases within a certain time interval (300 to 800 ms) – are fulfilled a step has been detected. In Figure 2-22 the pattern of the measured acceleration during one step and the support phase are generically depicted.

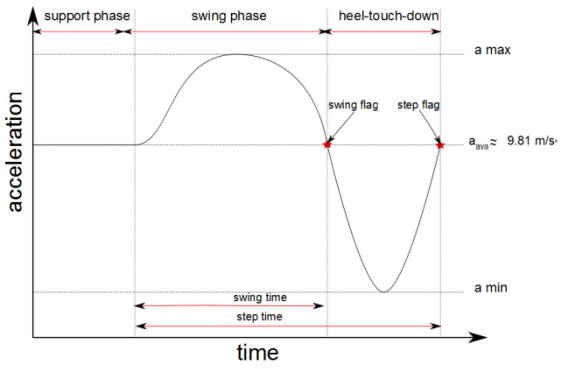


Figure 2-22: Measured vertical acceleration during one step [47]

As stated already in 2.5.2.1.1 the step length is dependent on the frequency of the steps and also on the measured acceleration. Based on the measured maximum vertical acceleration and the minimum vertical acceleration the step length can be estimated using this equation:

$$l = k * \sqrt[4]{a_{max} - a_{min}} \quad \textbf{(2-12)}$$

With:

l	Step length
k	Individual *k*-factor
a_{max}	Measured maximum vertical acceleration during one step
a_{min}	Measured minimum vertical acceleration during one step

This equation to determine the length of one step is based on the work of [49]. The estimation of the step length is based on empirical measurements instead of models (compare 2.5.2.1.1). The maximum vertical acceleration and the minimum vertical acceleration can be easily measured but the *k*-factor has to be determined within a separate calibration process for each user individually. Therefore the user has

to walk a track of a known length. During the calibration a *k*-factor of 1 is assumed and based on this the step length and the complete length of the walked track is computed. The *k*-factor is then determined based on the relation between real track length and computed track length:

$$k = \frac{d_{est}}{d_{real}} \quad \textbf{(2-13)}$$

With:

d_{est} Distance estimated by the step detection and step length estimation algorithm

d_{real} Distance walked during calibration by the user

This *k*-factor has to be determined only once but individually for each user, is stored within the mobile device, and used each time pedestrian dead reckoning is necessary.

To determine the heading of the user in the work of [47] the measurements of the magnetometer are used together with the acceleration of the mobile phone. The Android-API provides functionality to compute the current angle between the geographic North (see 5.1.3) and the y-axis (azimuth) of the mobile phone in degree using magnetometer and acceleration measurements. In 2.5.1.2.3 it is already mentioned that the quality of the magnetometer measurements is strongly influenced by electro-magnetic sources, which is especially within buildings but also in urban environments a problem. Although this problem is known the magnetometer measurements are used in this prototype to estimate the heading of a walking person. Currently in an on-going practical study the implementation of a dead reckoning algorithm using gyroscope measurements is tested. In 3.5.1 the results of the dead reckoning algorithm using magnetometer measurements are reviewed.

2.6 Cooperative Positioning

Although probably nobody would refuse to help another person asking for directions there are only few approaches in the field of positioning that deal with the sharing of information. One of the reasons might be that in this case one cannot control what happens with the information revealed to another device and therefore to another person. And secondly one does not have a direct benefit from providing her knowledge to someone else. Nevertheless some approaches rely on the willingness to cooperate with other mobile devices to improve the overall position accuracy.

2.6.1 AGPS based on P2P networks

On behalf of ESA the Politecnico de Torino started research related to cooperative positioning in 2010 based on GNSS data only and on a combination of GNSS data and terrestrial ranging information. The aiding strategies for GNSS data only are similar to those of AGPS described in 2.3 except that assistance data is coming from another peer nearby who has access to satellite data instead of a location server.

From the papers [50] and [3] the term light indoor (LI) environment which is used in this thesis as well (see 2.1.1) originates. It describes an environment where the receiver is able to receive weak signals but the computation of a position is not possible due to satellite signals coming from less than four satellites. Users or peers are therefore classified in LI peers and OS (open sky) peers; the latter having access to enough satellite data to compute a position and are able to assist other peers. In contrast to AGPS the peers are not time synchronized to each other neither coarsely nor finely and the diameter of the area where aiding and assisted peers reside should not exceed 1km.

In [3] three peer-to-peer assistance strategies are described. For the first method Doppler frequency of satellites in view are shared between the peers. The effect is similar to AGPS: Due to the reduced search space either TTFF can be decreased or a longer integration time can be enabled. The quality of the received Doppler frequency, however, is dependent on the accuracy of the local oscillator of the aiding peer. Since smartphones mostly comprise quartz oscillators with a very limited accuracy the idea is to collect the Doppler frequency of several peers and use a weighting strategy based on the distance to the peers or the quality of the oscillator.

The second method is described in detail in [50] and is based on certain signal characteristics of Galileo E1 frequency band. The assisted peer uses information about the secondary code delay of the Galileo E1 pilot channel to perform a wipe-off of the secondary code. The Galileo signal has another modulation strategy based on a primary and a secondary code. This makes the signal more robust compared to GPS C/A code. If a wipe-off of the secondary code can be performed the acquisition has to be performed only on the primary code which is shorter than the secondary one. But as described in 2.3.2 to enable such a wipe-off the peers would have to be synchronized with an accuracy according to the code length of the secondary code of Galileo E1.

In the third method C/N_0 data is exchanged between the peers. Although this is already identified as an aiding strategy for the AGNSS standard [3] it is not exactly specified how this parameter can help the receiver to compute a position if it comes from a location server which normally has unobstructed LOS. According to [3] the exchange of C/N_0 data of satellites in view between peers is more beneficial than receiving it from some server since they currently reside within a similar environment if they are able to establish P2P networks. Again the incoming data of several peers is used in a weighting algorithm to obtain the most likely C/N_0 value for a peer. The received value is used to adapt the receiver to weak signals by extending the integration time and choosing an adequate number of coherent and non-coherent integrations.

The combination of GNSS data and terrestrial ranging information uses radio signal strength (RSS) or the estimation of round trip time (RTT) of P2P networks (e.g. Bluetooth, WiFi direct) to compute distances between peers. According to [51] *pseudoranges* of less than four satellites and range

measurements to other peers are used to compute a position. A coarse knowledge of one's own position is therefore necessary and needs to be shared with other peers. With position estimations and range measurements coming from other peers and receivable *pseudoranges* from satellites equation (2-1) again is over-determined and can be solved iteratively by each peer on its own.

Although these approaches enabling cooperative positioning sound promising not all issues that might occur are addressed by the publications. For all papers and articles concerning the work in this area by the Politecnico de Torino it is assumed that the peers have smartphones equipped with a GNSS receiver and access to radio communication units. The same assumption is used as a requirement for the P2PKF (see 1.2) but this also means that the access to receiver data can be very restricted. In 3.4 where the implementation of the prototype is described it will become obvious that the access to data like *pseudoranges* is not possible for today available smartphones and the standard receiver independently of the fact that at the moment no low-end receivers exist exploiting Galileo signals. Although the access of such data might become possible in the near future it is very unlikely that signal processing algorithms in a hardware receiver can be accessed from applications running on the smartphone as it would be necessary to exploit the expected C/N_0 value. Also the problems of data exchange and privacy issues which occur for such an approach are not addressed as well. In 3.4 and 3.5 the problem of establishing autonomous connections between peers to exchange data is discussed. As can be observed in 3.5 where the prototype evaluation is described this is not an easy task. However in [52] it is assumed that this might only be a matter of time since for car-to-car (C2C) communication such limitations are already abolished.

2.6.2 Cooperative positioning in Car2Car (C2C) networks

The C2C communication standard was developed based on the WLAN standard IEEE 802.11 and upon the suggestions of the C2C-Communication Consortium (C2C-CC). The C2C-Communication Consortium was founded as non-profit industrial driven organization by vehicle manufacturers like among others BMW Group, Audi, Fiat Group and Honda. While the purpose of inter vehicle communication mainly focuses on the emission of messages containing security warnings and directives to improve the traffic efficiency there are also efforts to extend the functionality of the on-board unit (OBU) which should normally be equipped with a GNSS receiver by cooperative positioning algorithms.

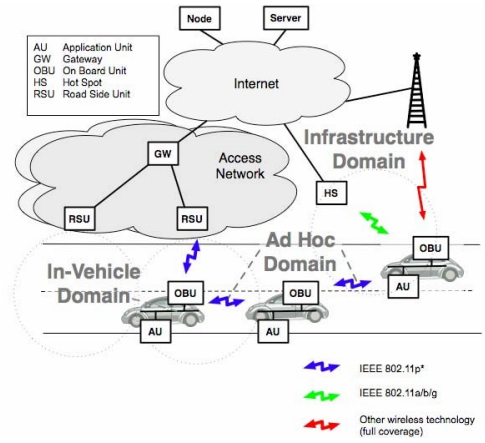


Figure 2-23: The three domains of the C2C communication system. The Ad-Hoc domain contains all transmission between the vehicles and the Road -side Units (RSU). In the In-Vehicle domain the data exchange between On-Board Unit (OBU) and application unit (AU) takes place. The Infrastructure domain is identified by the direct communication from an OBU directly to a wireless network without using a RSU (figure taken from [53]).

The manifest [53] which was published in 2007 is the first official publication of the C2C-CC and describes the general functionality of the communication system, several use cases, and the standards it is based on. Until July 2010 this manifest also was the only official document regarding the description of the physical network layer (PHY), the Logical Link Layer (LLC), and the Medium Access Control

(MAC) layer respectively. Those lower network layers are based on the IEEE 802.11 standard [54] which is completed by [20] published in July 2010. The latter standard – IEEE 802.11p or IEEE 802.11p* adapted for European traffic – is still a draft version and therefore not all topics are completely finalized. The upper protocol layers are described by the standards IEEE 1609.1 to IEEE 1609.4 ([55], [56], [57], [58]).

Based on the proposed message formats there exist different suggestions to extend the existing standards by cooperative positioning methods. In [59] it is assumed that there are vehicles which have access to GNSS signals and vehicles which are not able to compute their position based on satellite signals. The un-equipped vehicle sends PREQ (Position Request) messages to its next neighbors (one-hop) which are assumed to have access to GNSS signals. If the requesting vehicle receives more than three PREP (Position Reply) messages it is able to compute a two-dimensional position. This approach also works with less than three PREP messages when the distance travelled between the position where PREQ messages are sent the first time and the position where the second PREQ messages are emitted is taken into account.

A very similar approach is chosen in [60]. In contrast to the approach proposed above it is assumed that all vehicles have access to GNSS signals and are able to compute their position. The idea here is to use the measured distance to other peers to eliminate errors based on GNSS range measurements. The communication is based on the extension of the DSRC (Dedicated Short Range Communication) messages.

Since the C2C standard is very young there are still several possibilities to extend the current functionality. The position estimation is an essential part of the C2C communication and vehicle networks are very vulnerable if GNSS is not available no matter if this is caused by the environment (tunnels, urban canyons) or by interfering sources like radio networks or jamming devices. It is therefore possible that the research will focus more on cooperative positioning approaches.

2.6.3 Other P2P positioning strategies

Most cooperative positioning methods are based on range measurements as introduced in 2.4. Signals of surrounding peers can be used to determine the position by TDOA, AOA or TOA. The distance to the emitting peer can either be defined by measuring the RSS or the RTT. In contrast to infrastructure networks where the nodes are synchronized with certain accuracy to each other the time synchronization in ad-hoc networks is a problem complicating the estimation of a position. Although within one cell the peers are synchronized within a certain degree to each other not necessarily all peers used to compute a position reside within one ad-hoc network cell. Additionally the range measurement by RSS can be erroneous when walls or other people are between the requesting peer and the peer which answers due to signal attenuation. Estimating the range by RTT is often more accurate but also asks for synchronized clocks within the peers. In each case the responding peers have to be aware of their current position. This is possible if they are residing in light indoor environment and have access to assistance data. In [61] a very comprehensive description of cooperative indoor positioning can be found especially with regard to algorithms and obtainable accuracies.

Another approach for peer-to-peer positioning describes [62]. The authors introduce a method to estimate the relative position of a fixed target to a moving user based on distance and direction. Actually the movement of the device is exploited to estimate the target's position. In contrast to the approaches described above the user needs sensors measuring travelled distance and direction changes. Additionally so called helper nodes (peers) are used for the position estimation nevertheless a surveyed infrastructure is not necessary.

The pervasiveness of peer-to-peer networks and also their simplicity seem to be a driver for researching their dynamics not only for positioning but also in other fields of applications. Since most people nowadays use mobile devices which integrate GNSS receivers it is logical to use the information

acquired by surrounding peers before requesting data from a remote server via a costly network. A diverse approach also is more robust to incidents and errors than a centralized approach. And as proven by [3] to [63] the performance is just as well or sometimes even better.

3. SIMULATION AND IMPLEMENTATION

The focus of this work is on the development and implementation of an algorithm for mutual filtering. This chapter is therefore dedicated to address all details regarding the technical realization of the P2P Kalman filter. To evaluate if mutual filtering is beneficial regarding the position accuracy of the peers simulations are performed before diving into the actual implementation development of a prototype. Both topics, the simulation and the implementation, are addressed in this chapter. The first section starts with a more detailed description compared to the introduction in 1.1 of the P2P Kalman filter approach. This definition is used as base for the simulation and implementation.

The following section introduces the different techniques and tools which are combined in the P2P Kalman filter approach. The principle of dead reckoning is explained mathematically as well as the filtering algorithms based on the discrete Kalman filter. Since these concepts are well known those parts are very concise and references are made at this point to other sources in literature. The last part handles different possibilities to establish connections between peers without involving infrastructure and based solely on so called ad-hoc networks.

In the third section the P2P filtering approach is evaluated by simulations implemented in Matlab. Since the core of this approach is the participation of as many users as possible it is easier to test if the position accuracy is improved by the data exchange based on simulations. Therefore an indoor environment is emulated where the access to GNSS signals is limited to certain spots and the peers are dependent on navigation based on dead reckoning. Different scenarios based on various parameters like the amount of peers are described. In the second part the implementation of the different types of Kalman Filters is addressed. At the end the results of the different simulation scenarios are reviewed and interpreted. Based on these results ideas for the implementation of the prototype are developed.

The implementation of the prototype is explained in detail in the fourth section. First a short introduction to the Android application programming interface (API) and Android in general is given. Also here references to more detailed sources are made. The structure of the prototype application is explained and the implementation of different modules dead reckoning, filtering and communication is described in its own sub-section respectively.

The fifth section evaluates the modules separately. First the position accuracy of the pure dead reckoning algorithm is tested separated into evaluation of the step detection and step length estimation, the heading estimation and then the overall position accuracy. Then the position accuracy for dead reckoning when employing additionally a Kalman filter based on the measurements of the sensors is tested as well. In the last part the communication between the devices is evaluated and here also the whole prototype is tested on the one hand for its general functionality and stability but also if the position accuracy of the different peers can be improved.

3.1 Introduction to the concept

It is assumed that the user has a smartphone equipped with a GPS or AGPS receiver, a triaxial gyroscope, a triaxial magnetic field sensor, and a triaxial accelerometer. On the mobile device an application like the one described in 2.5.3 is installed providing step detection, step length estimation and the heading relative to the geographic North. The output of the dead reckoning application is based on the measurements of the inertial sensors. This output could also be integrated into the position estimated by GPS as long as satellite signals are available but the possible coupling of INS and GPS is not addressed in this thesis.

As soon as the user enters a narrow street or a building and the smartphone loses GPS signals it is not able anymore to provide a position solution based on satellite signals. The device switches then completely to dead reckoning and continues to navigate based on the last position estimated by GPS signals as starting point and step detection, step length estimation, and heading estimation. The position which is computed by measurements coming from the inertial sensors is two-dimensional. A three-dimensional position is also conceivable but asks for sensors determining the height of the user by e.g. a barometer. Since this sensor is not yet a standard in current mobile devices this was not considered in this work. It is supposed that since the user is currently navigating the mobile device is hold in one hand at eye level or little below.

Due to the short-term stability characteristics of inertial navigation the position accuracy most likely degrades over time. To avoid this, the user's device searches constantly for other devices also currently navigating by either GPS positioning or inertial navigation or both. As soon as another device resides within a certain transmission range the devices exchange their current estimated position and their uncertainness of this position in form of a variance. The received and the estimated positions are then processed within a filtering algorithm using the variance as weighting strategy. Each user's device does this on its own and uses the newly computed position solution as new starting point for dead reckoning. Directly after the data exchange the connection between the users is closed and a new connection with other users can be established.

Ideally the position accuracy is improved by the filtering because users which have for example a more sensitive receiver are able to provide better positions with a smaller variance than users with less advanced receivers. It is assumed that one estimated user position is more accurate than the other one and this is expressed by the weighting of the position solution. If possible the user should not be aware of the fact that the navigation process is triggered by signal availability. The position should be received seamlessly coming from outdoors to indoors and vice versa without losing the possibility to navigate.

Three issues have to be addressed when implementing such a positioning algorithm:

1) An economical but effective dead reckoning and filtering algorithm has to be developed since the resources of a mobile device in terms of processing capacity and power supply are limited.

2) The communication links between the user devices have to be established and closed quickly without bothering network infrastructure and the user.

3) The filtering algorithms should provide a better position accuracy overall but also individually for one user; the filtering algorithm therefore has to be carefully examined and important parameters defined.

3.2 Tools

As mentioned in the list above, these three issues have to be addressed to provide a prototype. An algorithm to determine the heading, the step length, and to detect steps already was introduced in 2.5.3. Based on these measurements a position has to be computed. The method to do this is explained in the first part of this section. It follows a short introduction to the mathematical deviations of the Kalman filter in general and two variations of the Kalman filter, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). On these filtering algorithms the simulations and the implementation is based. The last part of this section addresses existing communication standards which are suitable for the realization of the P2P Kalman filter. Three standards are introduced which enable P2P ad-hoc networks within small transmission ranges.

3.2.1 Dead Reckoning

As explained in 2.5 dead reckoning is the computation of a position based on a position before. As depicted in Figure 3-1 the user starts from a certain start position which has to be estimated as accurately as possible since the rest of the navigation depends on the accuracy of this first estimation. Using classical inertial navigation this would be done during the initialization phase using e.g. GPS positioning or DGPS (Differential GPS) which is able to provide centimeter-accurate positions with the help of correction parameters coming from a network of reference stations. This effort of course is necessary and important when initializing an IMU for safety relevant services. For pedestrian navigation the start position is estimated by using the integrated GPS receiver which results in accuracy between 1 and 10 m depending on the signal availability and the receiver quality. The last position which can be derived by satellite signals denotes the start position for dead reckoning. From here the current position is estimated step-wise based on the last position. To compute a new position comprising x and y for the Matlab simulations the following two equations are used.

$$x_n = x_{n-1} + s_{n-1} * \sin(\theta_{n-1}) \tag{3-1}$$

$$y_n = y_{n-1} + s_{n-1} * \cos(\theta_{n-1}) \tag{3-2}$$

The x- and y-coordinate of the position are computed based on the coordinates from the step before (x_{n-1} and y_{n-1}), the step length s_{n-1}, and the sine or cosine of the heading in degree θ_{n-1}. A new position is only estimated when the user actually makes a step. The step detector as described in 2.5.3 indicates if the user made a step; simultaneously the length of this step and the heading of the user have to be determined. Since the position computation is not based on integration about acceleration as for classical inertial navigation the complicated synchronization of the sensor measurements and the update rate of the IMU are omitted and the algorithm to estimate the next position is driven by the walking frequency of the user.

For the implementation of the prototype described in 3.4 another algorithm is used to compute a new position after the user made a step. Since a GPS receiver normally provides ellipsoidal coordinates based on the WGS84 ellipsoid the use of other equations becomes necessary. These equations are derived from the Haversine formula which enables the computation of great circle distances between two points on a sphere based on their longitudes and latitudes. Rearranging the Haversine formula enables the computation of new coordinates in latitude and longitude when the starting coordinates, the heading and the distance are known.

$$\varphi_{new} = \sin^{-1}\left(\sin(\varphi_{old}) * \cos\left(\frac{d}{R}\right) + \cos(\varphi_{old}) * \sin\left(\frac{d}{R}\right) * \cos(\theta)\right) \tag{3-3}$$

$$\lambda_{new} = \lambda_{old} + \tan^{-1}\left(\sin(\theta) * \sin\left(\frac{d}{R}\right) * \cos(\varphi_{old}), \cos\left(\frac{d}{R}\right) - \sin(\varphi_{old}) * \sin(\varphi_{new})\right) \tag{3-4}$$

With

$\varphi_{new}, \varphi_{old}$	New and old latitude
$\lambda_{new}, \lambda_{old}$	New and old longitude
d	Distance between new and old position (i.e. step length)
θ	Bearing between new and old position (i.e. heading as computed by accelerometer and magnetometer)
R	Earth radius

For this implementation the simpler variation assuming a spherical Earth is used. Since the distances between the coordinates are very small (below 1 m according to the step length of a user) it was decided to implement the dead reckoning application based on this assumption.

As illustrated in Figure 3-1 the position variance denoted by the circles around the estimated position increases due to erroneous sensor measurements, wrongly detected steps and the uncertainty about the start position. The rate of the growing error is dependent on the quality of the sensor measurements and the accuracy of the start position (compare 2.5.1.3). This progress can only be mitigated by external updates to correct the position estimation of an INS. As mentioned in 2.5.1.4 there exist several strategies combining GPS and INS measurements. While GPS signals are available the output of the IMU can be used to stabilize the position estimation of GPS and if GPS signals are not available the IMU provides the position estimation by itself and is updated by GPS measurements as soon as signals are again available. The variance of the position estimation decreases when integrating GPS measurements with a better accuracy than the IMU measurements.

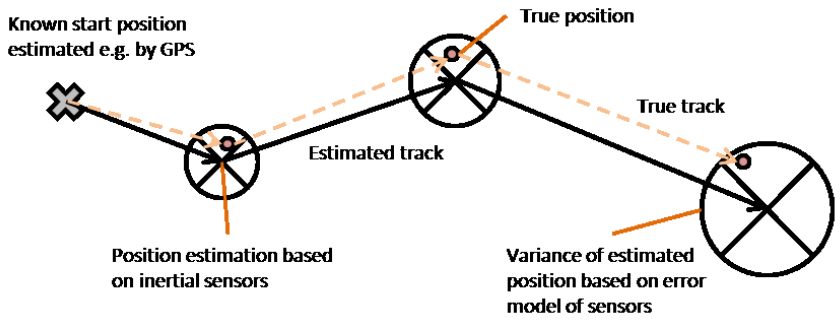


Figure 3-1: Illustration of navigation by dead reckoning [5]

The P2P Kalman filter as implemented here is able to additionally use the measurements of other peers to update the position estimated only by inertial sensors. Also here the variance of the position increases when relying only on the measurements of inertial sensors and decreases when updating the position with external measurements from another peer. This reduction is based on the equations of the Kalman filter explained in the following section.

3.2.2 Kalman Filter

A Kalman filter is a commonly used mechanism to draw conclusions from erroneous measurements about the state of a system. R. Kalman introduced this concept [64] in 1960. Since then the Kalman filter has been exploited in various fields of applications not only for navigation and positioning. In [65] pp. 3 a very basic version of the functionality of the Kalman Filter is introduced and in [66] amongst others an easy understandable example is given. The mathematical derivation of the filter equations are explained in detail in [64] and of course in [65]. As a Kalman filter is a very common concept today the

functionality is presented here only briefly and therefore it is foregone to analyze mathematical proofs and the derivation of the Kalman Filter equations. For more information on the Kalman filter refer to [64] and [65]. The following descriptions of the different Kalman filter types are a summarization of [64], [65] and [66]. Of more interest in this work is the implementation of the Kalman filter for the P2P positioning which is explained in detail in 3.4 and its variations like the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). The idea of the Kalman filter is illustrated in Figure 3-2:

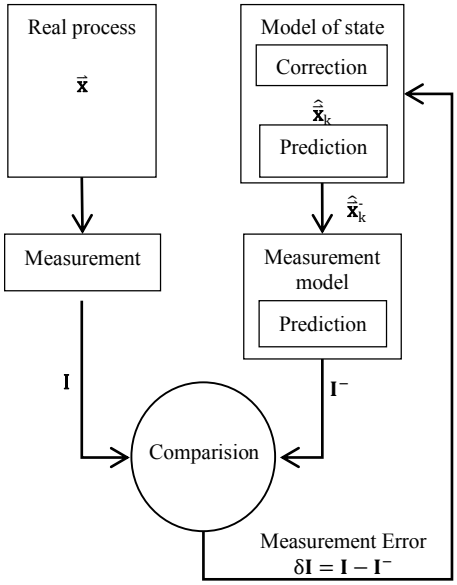


Figure 3-2: The principle of the Kalman filter: A process in real life is modeled by mathematic equations.

3.2.2.1 Discrete Kalman Filter Equations

Expressed mathematically a Kalman Filter is a mechanism to estimate a state $x \epsilon \mathbb{R}^n$ of a discrete time controlled process generated by a linear stochastic difference equation:

$$\vec{\mathbf{x}}_k = \vec{\mathbf{x}}_{k-1}\mathbf{A} + \mathbf{B}\vec{\mathbf{u}}_{k-1} + \vec{\mathbf{w}}_{k-1} \quad \textbf{(3-5)}$$

And the measurement $\vec{z}_k \in \mathbb{R}^m$:

$$\vec{\mathbf{z}}_k = \vec{\mathbf{x}}_k\mathbf{H}\vec{\mathbf{v}}_k \quad \textbf{(3-6)}$$

Where

$\vec{\mathbf{x}}_k$	The state of the system at time step k
$\mathbf{A}$	The n-by-n transformation matrix which relates the current state $\vec{x}_k$ to the state before $\vec{x}_{k-1}$
$\mathbf{B}$	The n-by-l matrix which relates the optional control input $\vec{u}_{k-1}$ to the current state $\vec{x}_k$
$\vec{\mathbf{u}}_{k-1}$	Optional control input

$\vec{\mathbf{w}}_{k-1}$ Process noise

$\vec{\mathbf{z}}_k$ Measurement

$\mathbf{H}$ The m-by-n matrix relates the current state to measurement $\vec{z}_k$.

$\vec{\mathbf{v}}_k$ Measurement noise

Both $\vec{w}_{k-1}$ and $\vec{v}_k$ are independent from each other, white and with normal probability distributions:

$$p(\mathrm{w}) \sim N(0, \mathbf{Q}) \tag{3-7}$$

$$p(\mathrm{v}) \sim N(0, \mathbf{R}) \tag{3-8}$$

Where $\mathbf{Q}$ is the process noise covariance matrix and $\mathbf{R}$ is the measurement noise covariance matrix.

As the true state $\vec{x}_k$ of the system is not known an estimate is used for the following equations. The state estimate $\hat{\vec{x}}_k^- \in \mathbb{R}$ at time step k is the so called *a priori* state estimate which is given knowledge of the process prior to k denoted by the superscript – and $\hat{\vec{x}}_k$ is the *a posteriori* state estimate at time step k given measurement $\vec{z}_k$. The *a priori* and the *a posteriori* estimate errors are defined by

$$\mathrm{e}_\mathrm{k}^- = \vec{\mathbf{x}}_\mathrm{k} - \hat{\vec{\mathbf{x}}}_\mathrm{k}^- \tag{3-9}$$

$$\mathrm{e}_\mathrm{k} = \vec{\mathbf{x}}_\mathrm{k} - \hat{\vec{\mathbf{x}}}_\mathrm{k} \tag{3-10}$$

Therefore the error covariance of the *a priori* and the *a posteriori* error are

$$\mathbf{P}_k^- = E[e_k^- e_k^{-T}] \tag{3-11}$$

$$\mathbf{P}_k = E[e_k e_k^T] \tag{3-12}$$

With E denoting the expected value.

The equations of the Kalman filter can be divided in two groups: The *time update* equations and the *measurement update* equations. The *time update* equations project the current state estimate and the error covariance estimate forward in time to obtain the *a priori* estimates for the next time step. The measurement update equations are used for the feedback of the filter. They integrate a measurement into the *a priori* estimates and create an improved *a posteriori* estimate. The *time update* equations can be referred to as *predictor* equations while the *measurement update* equations serve as *corrector* equations. The final algorithm resembles that of a recursive predictor-corrector algorithm for numerical problems and the Kalman filter alternates between the equations of the predictor and the ones of the corrector.

The equations for the *time update* are:

$$\hat{\vec{\mathbf{x}}}_k^- = \mathbf{A}\hat{\vec{\mathbf{x}}}_{k-1} + \mathbf{B}\vec{\mathbf{u}}_{k-1} \quad \textbf{(3-13)}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \quad \textbf{(3-14)}$$

Equation (3-13) computes the estimated state of the system model based on the *a posteriori* state from the time step before and the matrices **A** and **B** as explained in (3-5). Equation (3-14) calculates the error covariance of the estimate before any measurements have been taken. The error covariance of time step *k* is based on the error covariance of the time step before when the measurement is already taken into account, the matrix **A** and the process noise covariance **Q**.

The equations for the measurement update are:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \quad \textbf{(3-15)}$$

$$\hat{\vec{\mathbf{x}}}_k = \hat{\vec{\mathbf{x}}}_k^- + \mathbf{K}_k(\vec{\mathbf{z}}_k - \mathbf{H}\hat{\vec{\mathbf{x}}}_k^-) \quad \textbf{(3-16)}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}\mathbf{P}_k^-) \quad \textbf{(3-17)}$$

Equation (3-15) computes the so called Kalman gain **K** which minimizes the *a posteriori* error covariance. The second equation of the *measurement update* (3-16) corrects the estimate of the state by using the Kalman gain as weighting matrix and the difference between the actual measurement $\vec{\mathbf{z}}_k$ and the predicted measurement $\mathbf{H}\hat{\vec{\mathbf{x}}}_k^-$. This is indicated by omitting the superscript "-". This difference is called the measurement *innovation* or *residual*. The more the residual equals zero the higher the accordance between predicted measurement and actual measurement is. Additionally one can infer from (3-15) that the smaller the measurement error covariance **R** gets the more the residuals are weighted by the Kalman gain and vice versa: The smaller the error covariance of the estimate gets the less the residuals are weighted. This means that the actual measurement $\vec{\mathbf{z}}_k$ is "trusted" more when the measurement error covariance approaches zero while the predicted measurement becomes less important. But as soon as the *a priori* error covariance of the estimated state reaches zero the actual measurement $\vec{\mathbf{z}}_k$ is trusted less and the predicted measurement $\mathbf{H}\hat{\vec{\mathbf{x}}}_k^-$ gains importance. The last equation (3-17) updates the estimated error covariance after taking the actual measurements. There are different ways to compute $\mathbf{P}_k$ in this case the identity matrix was used. For more information of the derivation of $\mathbf{P}_k$ refer to [64] and [65].

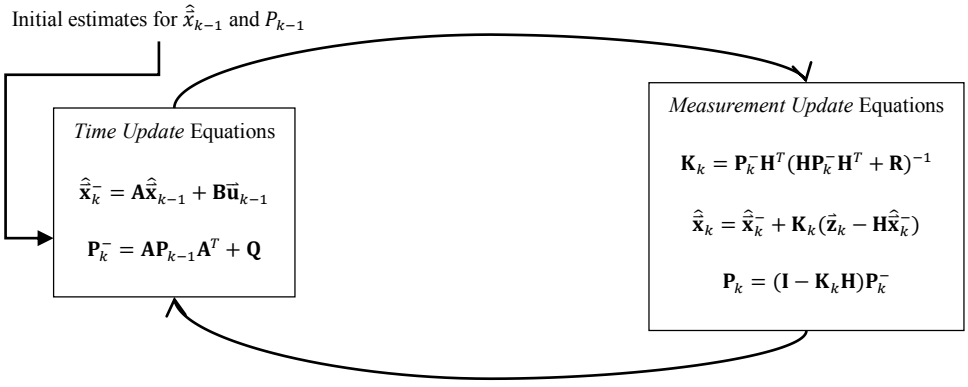


Figure 3-3: The equations of the Kalman Filter taken from [66]

As mentioned before the Kalman filter algorithm alternates between *time update* and *measurement update*. This means that the estimates of state and error covariance which are corrected by actual measurements during the *measurement update* equations serve as *a priori* input for the next *time update*

equations. Due to this property it is possible to implement the Kalman filter recursively and to avoid to operate on all data directly for each estimate as the Wiener Filter suggest it. Although the Kalman filter does not use all data available due to its recursive nature the information of all past estimates and measurements are included within the current estimate of state and the current error covariance. The difficulty can be the selection of the initial estimates of state and the error covariance. Before processing the filter it is also preferable to estimate the measurement noise covariance **R**. Estimating the process noise covariance **Q** normally is a far more difficult process as it is mostly impossible to identify the quality of the system model which is observed.

3.2.2.2 Extended Kalman Filter (EKF)

As mentioned in the section above the Kalman filter equations are used for the problem of estimating a state of a discrete time-controlled process which is adjusted by a **linear** stochastic difference equation. It is possible to modify the Kalman filter in a way that also the state of non-linear processes and a non-linear measurement relationship can be estimated. A Kalman filter linearizing about the current mean and covariance is called an Extended Kalman Filter (EKF).

The actual state vector $x \epsilon \mathbb{R}^n$ is computed by a non-linear stochastic difference equation:

$$\vec{\mathbf{x}}_k = f(\vec{\mathbf{x}}_{k-1}, \vec{\mathbf{u}}_{k-1}, \vec{\mathbf{w}}_{k-1}) \tag{3-18}$$

And its measurement $\vec{z}_k \in \mathbb{R}^m$:

$$\vec{\mathbf{z}}_k = h(\vec{\mathbf{x}}_{k-1}^{-}, \vec{\mathbf{v}}_{k-1}) \tag{3-19}$$

Where $\vec{\mathbf{w}}_k$ and $\vec{\mathbf{v}}_k$ represent the zero-mean process and measurement noise as defined by (3-7) and (3-8) respectively as input of the non-linear function f and the non-linear function h. Function f relates the state of the previous time step $k-1$ to the state of the current time step k driven by optional control input $\vec{\mathbf{u}}_{k-1}$. Function h relates the current state $\vec{\mathbf{x}}_k$ to the current measurement $\vec{\mathbf{z}}_k$.

To estimate the covariance matrices and the Kalman gain matrix a transfer matrix and a design matrix is used. Those matrices are obtained by computing the Jacobian of transfer function and measurement function. The matrices contain the partial derivatives of function f and function h with respect to $\vec{\mathbf{x}}_k$:

$$\mathbf{A}_k = \frac{\partial f_k}{\partial \mathbf{x}_k} \quad \text{Transfer matrix} \tag{3-20}$$

$$\mathbf{H}_k = \frac{\partial h_k}{\partial \mathbf{x}_k} \quad \text{Design matrix} \tag{3-21}$$

$$\mathbf{W}_k = \frac{\partial f_k}{\partial \mathbf{w}_k} \quad \text{Jacobian matrix of } f_k \text{ w.r.t. } w_k \text{ process noise} \tag{3-22}$$

$$\mathbf{V}_k = \frac{\partial h_k}{\partial \mathbf{v}_k} \quad \text{Jacobian matrix of } h_k \text{ w.r.t. } v_k \text{ of the measurement noise} \tag{3-23}$$

The Jacobian matrices are obtainable by means of numerical or analytical computation. If the functions do not change with each time step it is sufficient to retrieve the Jacobian matrices once analytically and use the current values of the state vector. If the Jacobian is built numerically the derivatives have to be obtained in each iteration step. The equations for the EKF are depicted below:

$$\hat{\vec{\mathbf{x}}}_k^- = f(\hat{\vec{\mathbf{x}}}_{k-1}, \vec{\mathbf{u}}_{k-1}, 0) \tag{3-24}$$

$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^{\mathrm{T}} + \mathbf{W}_k\mathbf{Q}_{k-1}\mathbf{W}_k^{\mathrm{T}} \tag{3-25}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k{}^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^{\mathrm{T}} + \mathbf{V}_k\mathbf{R}_k\mathbf{V}_k^{\mathrm{T}})^{-1} \tag{3-26}$$

$$\hat{\vec{\mathbf{x}}}_k = \hat{\vec{\mathbf{x}}}_k^- + \mathbf{K}_k(\vec{\mathbf{z}}_k - h(\hat{\vec{\mathbf{x}}}_k^-, 0)) \tag{3-27}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{3-28}$$

The first two equations denote the *time update*. In (3-24) the state of $\hat{\vec{\mathbf{x}}}_k^-$ is propagated ahead in time based on the state vector $\hat{\vec{\mathbf{x}}}_{k-1}$ of the time step before and the possible control input $\vec{\mathbf{u}}_{k-1}$. Since the process noise is zero-mean it is denoted with 0. The computation of the *a priori* error covariance $\mathbf{P}_k^-$ is depicted in (3-25) and is based on the transfer matrix $\mathbf{A}_{k-1}$, the process noise matrix $\mathbf{Q}_{k-1}$, and the Jacobian matrix $\mathbf{W}_k$ as defined by (3-20) and (3-22). Equations (3-26) to (3-28) define the computations of the *measurement update*. First the Kalman gain has to be obtained using the *a priori* error covariance matrix $\mathbf{P}_k^-$, the design matrix $\mathbf{H}_k$ as defined by (3-21), the measurement noise matrix $\mathbf{R}_k$, and the Jacobian matrix $\mathbf{V}_k$ as defined by (3-23). The next step is the computation of the *a posteriori* state vector. As for the discrete Kalman filter the difference of the real measurement $\vec{\mathbf{z}}_k$ and the predicted measurement $h(\hat{\vec{\mathbf{x}}}_k^-, 0)$ results in a residual which is weighted by the Kalman gain $\mathbf{K}_k$ and added to the *a priori* state vector $\hat{\vec{\mathbf{x}}}_k^-$. Also here the measurement noise $\vec{\mathbf{v}}_{k-1}$ is denoted with zero since it is zero-mean. In the last step the *a posteriori* error covariance $\mathbf{P}_k$ is computed based on the identity matrix $\mathbf{I}$, the Kalman gain $\mathbf{K}_k$, the design matrix $\mathbf{H}_k$ and the *a priori* error covariance matrix $\mathbf{P}_k^-$.

In Figure 3-4 all equations of the EKF are summarized:

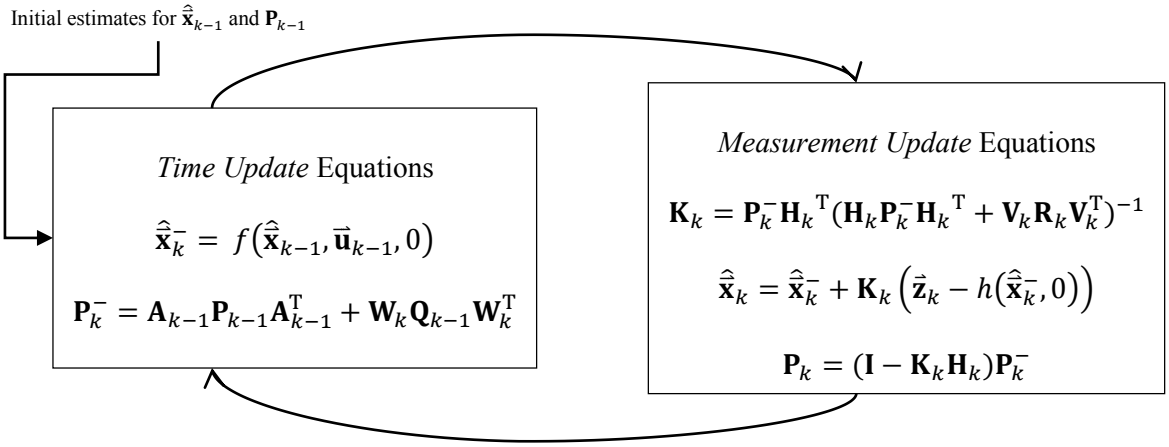


Figure 3-4: Equations of the Extended Kalman Filter [66]

As for the common Kalman filter an initial guess for the state vector and its covariance has to be existent.

Implementing an EKF seems a reasonable possibility to handle non-linear random processes. Unfortunately the EKF is only capable of dealing with a small degree of non-linearity otherwise it does not produce the desired results. Another problem is the computation of the Jacobian matrices. Even today these calculations have a strong impact on processor performance especially if they have to be

computed for each time step. An alternative the Unscented Kalman Filter (UKF) is described in the section below.

3.2.2.3 Unscented Kalman Filter (UKF)

The UKF is an occurrence of the so called Sigma-Point-Kalman-Filter. For the Sigma-Point-Kalman-Filter the mean and the covariance of a Gaussian random variable are defined by deterministically chosen sigma points. The sigma point itself again is a random variable. The state vector as well as the covariance matrix is extended by system noise and measurement noise:

$$\hat{\vec{\mathbf{x}}}_k^a = E[\vec{\mathbf{x}}_k^a] = (\vec{\mathbf{x}}_k^T, \vec{\mathbf{w}}_k^T, \vec{\mathbf{v}}_k^T)^T \tag{3-29}$$

$$\mathbf{P}_k^a = E[(\hat{\vec{\mathbf{x}}}_k^a - \vec{\mathbf{x}}_k^a)(\hat{\vec{\mathbf{x}}}_k^a - \vec{\mathbf{x}}_k^a)^T] = \begin{pmatrix} \mathbf{P}_k & 0 & 0 \\ 0 & \mathbf{Q}_k & 0 \\ 0 & 0 & \mathbf{R}_k \end{pmatrix} \tag{3-30}$$

The amount of elements of the extended state vector is stored in variable L. To represent mean and covariance $2L + 1$ sigma points are computed:

$$\vec{\boldsymbol{\mathcal{X}}}_{0,k}^a = \hat{\vec{\mathbf{x}}}_k^a \tag{3-31}$$

$$\vec{\boldsymbol{\mathcal{X}}}_{i,k}^a = \hat{\vec{\mathbf{x}}}_k^a + \zeta\sqrt{\mathbf{P}_{ki}^a} \text{ for } i = [1, \dots, L] \tag{3-32}$$

$$\vec{\boldsymbol{\mathcal{X}}}_{i,k}^a = \hat{\vec{\mathbf{x}}}_k^a - \zeta\sqrt{\mathbf{P}_{ki}^a} \text{ for } i = [L+1, \dots, 2L] \tag{3-33}$$

A sigma point therefore has the dimension L-by-1 and scaling parameter ζ defines the distance of the sigma points to the mean. To compute the square root of the extended covariance matrix different ways are possible. The Cholesky decomposition is based on this relation:

$$\sqrt{\mathbf{A}} = \mathbf{B} \Rightarrow \mathbf{A} = \mathbf{B}\mathbf{B}^T \tag{3-34}$$

Another possibility to define the square root of a matrix is:

$$\sqrt{\mathbf{A}} = \mathbf{B} \Rightarrow \mathbf{A} = \mathbf{B}^2 \tag{3-35}$$

In [22] the Cholesky decomposition is proposed for the UKF as well as in [67]. Therefore this is also the approach when implementing the P2P Kalman filter.

Within the *time update* the sigma points are propagated by the non-linear system model:

$$\vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-} = f(\vec{\boldsymbol{\mathcal{X}}}_{i,k-1}^a, \vec{\mathbf{u}}_{k-1}) \tag{3-36}$$

Using these transformed sigma points the mean and the covariance matrix are calculated by:

$$\hat{\hat{\mathbf{x}}}_k^{a-} = \sum_{i=0}^{2L} w_i^m \, \vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-} \tag{3-37}$$

$$\mathbf{P}_k^{a-} = \sum_{i=0}^{2L} \sum_{j=0}^{2L} w_{ij}^c (\vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-} - \hat{\hat{\mathbf{x}}}_k^{a-})(\vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-} - \hat{\hat{\mathbf{x}}}_k^{a-})^T \tag{3-38}$$

The computation of the weighting factors of w_i^m and w_{ij}^c can be found in (3-47) to (3-49). During the *measurement update* the sigma-points are propagated by the non-linear measurement model:

$$\vec{\mathbf{Y}}_{i,k} = h(\vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-}) \tag{3-39}$$

Based on these measurements the mean, the covariance matrix and the cross-correlation can be defined:

$$\hat{\hat{\mathbf{z}}}_k = \sum_{i=0}^{2L} w_i^m \, \vec{\mathbf{Y}}_{i,k} \tag{3-40}$$

$$\mathbf{P}_{zz,k} = \sum_{i=0}^{2L} \sum_{j=0}^{2L} w_{ij}^c \, (\vec{\mathbf{Y}}_{i,k} - \hat{\hat{\mathbf{z}}}_k) \, (\vec{\mathbf{Y}}_{i,k} - \hat{\hat{\mathbf{z}}}_k)^T \tag{3-41}$$

$$\mathbf{P}_{xz,k} = \sum_{i=0}^{2L} \sum_{j=0}^{2L} w_{ij}^c \, (\vec{\boldsymbol{\mathcal{X}}}_{i,k}^{a-} - \hat{\hat{\mathbf{x}}}_k^{a-}) \, (\vec{\mathbf{Y}}_{i,k} - \hat{\hat{\mathbf{z}}}_k)^T \tag{3-42}$$

When these values are defined the Kalman gain, the current state estimate, and the covariance matrix can be computed by:

$$\mathbf{K}_k = \mathbf{P}_{xz,k} (\mathbf{P}_{zz,k})^{-1} \tag{3-43}$$

$$\hat{\hat{\mathbf{x}}}_k^{a} = \hat{\hat{\mathbf{x}}}_k^{a-} + \mathbf{K}_k (\tilde{\hat{\mathbf{z}}}_k - \hat{\hat{\mathbf{z}}}_k) \tag{3-44}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{zz,k} \mathbf{K}_k^T \tag{3-45}$$

The computation of the weighting factors w_i^m and w_{ij}^c and scaling factor ζ define the type of the Sigma-Point-Kalman-Filter. As stated before the UKF is an occurrence of the Sigma-Point-Kalman-Filter whose sigma points are calculated by a certain algorithm as described in (3-46) to (3-49). Regardless which type of Sigma-Point-Kalman-Filter is implemented the weighting factors have to fulfill the following equation otherwise a reasonable mean cannot be computed:

$$\sum_{i=0}^{2L} w_i^m = 1 \tag{3-46}$$

An example for the computation of the weighting factors is depicted in the equations below:

$$w_i^m = \begin{cases} w_0 < 1 \; i = 0 \\ \dfrac{1 - w_0}{2L} \; i > 0 \end{cases} \tag{3-47}$$

$$w_{ij}^c = \begin{cases} 0 \; i \neq j \\ \dfrac{1 - w_0}{2L} \; i = j \end{cases} \tag{3-48}$$

$$\zeta = \sqrt{\frac{L}{1 - w_0}} \tag{3-49}$$

The larger the value of $w_0 < 1$ is the larger is the distance of the sigma points to the mean. As for the EKF the equations of the UKF are summarized in Figure 3-5 separated into *time update* and *measurement update* equations and additionally the computation of the sigma points is depicted.

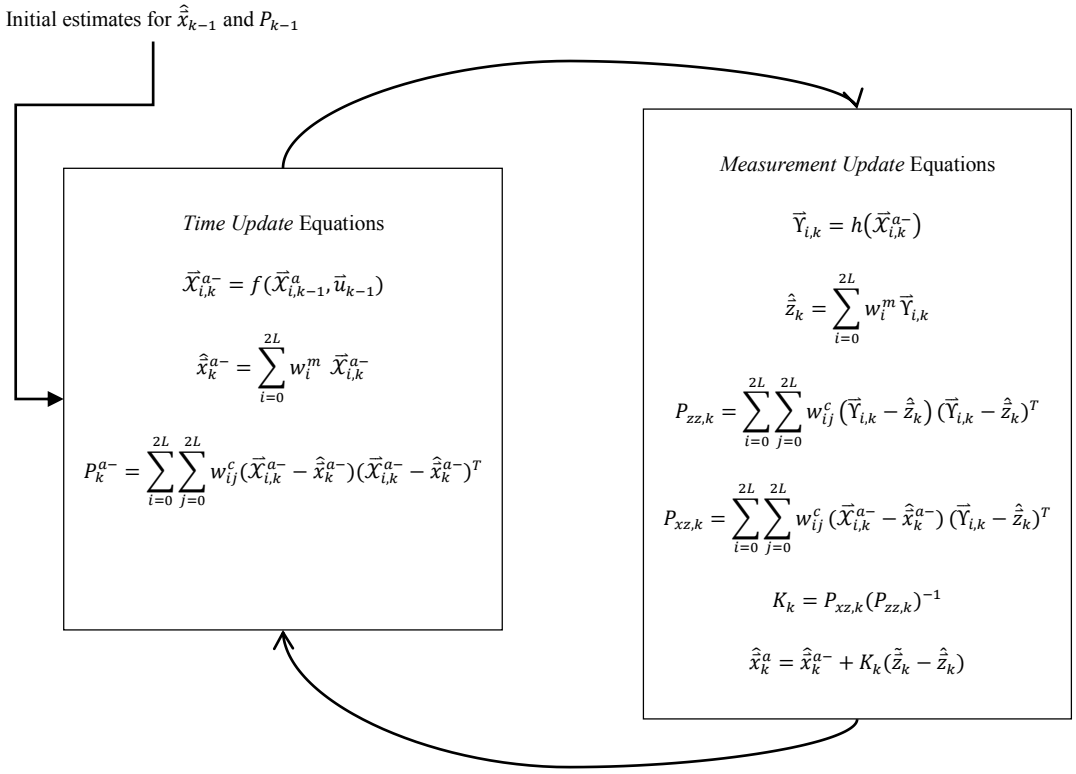


Figure 3-5: The equations of the Unscented Kalman Filter including the computation of the Sigma-Points and their propagation by the measurement model

3.2.3 Ad-Hoc Networks

The term ad-hoc network describes a certain type of network which is not depending on the infrastructure of a provider like e.g. access points or base stations providing access to the Internet but rather temporary communication networks directly established between two or more peers to exchange data from one device to another. This type of networks is quickly created and closed limiting the use of authentication or acknowledgment messages to a minimum. There exist several types of communication standards enabling ad-hoc functionality but only few of them are widely established. Since the functionality of the P2P Kalman filter depends on its use by many peers only standards are regarded in this section which are already integrated into most mobile devices or soon will be integrated.

The two most common standards for ad-hoc networks are the Bluetooth standard (IEEE 802.15) and the wireless local area network (WLAN) standard (IEEE 802.11). Additionally two less familiar standards

are presented as well since they provide interesting characteristics for the P2P Kalman filter communication. Those standards are ZigBee and the relatively new WiFi direct.

3.2.3.1 Bluetooth

In 1998 the Bluetooth Special Interest Group (SIG) was formed by mainly five companies (Ericsson, Intel, IBM, Nokia and Toshiba) [68], [17]. Bluetooth is an industry standard specified by the IEEE 802.15.1 standard [69], [70] and [71] which is used for short-range peer-to-peer connections. Its technology is based on the work of Prof. Jaap Haartsen and Dr. Sven Mattisson for Ericsson [72]. The main intention of Bluetooth was the replacement of cable connections between devices. The name Bluetooth is related to the epithet of the tenth-century Danish king Harald I who united the at this time dissonant Danish tribes to one kingdom. Since also the Bluetooth standard attempts to connect different kinds of devices to communicate with each other this name was chosen [73].

Almost at the same time the Bluetooth SIG was formed a new IEEE working group (IEEE 802.15) named WPAN (Wireless Personal Area Network) was started whose intention is the examination of networks regarding especially market potential, compatibility, autonomy, technical feasibility, and economical feasibility. This WPAN group and Bluetooth SIG worked from the beginning on very tightly together. The lower protocol layers of Bluetooth are specified by the IEEE 802.15.1 sub-standard [69] which was created in cooperation with the Bluetooth SIG. The upper protocol layers are described in [70] and [71]. Several other sub-working groups (802.15.2 to 802.15.4) deal with other issues related to WPANs. The fourth sub-working group is tightly related to the ZigBee Consortium another standard which covers WPANs with low data rate. The ZigBee standard will be addressed in 3.2.3.2.

Bluetooth works in the 2.4 GHz ISM (Industrial, Scientific, and Medical) frequency band separated into 79 channels with a distance of 1 MHz. To distinguish different networks from each other Bluetooth uses frequency hopping spread spectrum (FHSS) with a pseudo random hopping sequence of 1600 hops per second. This hopping sequence is defined by the device ID of the master station establishing a piconet and the current time. A device creating a piconet automatically becomes the master station while all other stations in this net are slaves. To be able to follow the hopping sequence the slaves have to synchronize to the master station therefore the master sends its current system time and the slaves align their clock to the time sent. Within one piconet only one master and seven active slaves can actively communicate with each other whereat the communication takes place only via the master station.

To enlarge the transmission area so called scatternets from overlapping piconets can be established. Since the device ID of a master station is used for defining the hopping sequence it is obvious that a master of one piconet has to be slave within an overlapping one. Scatternets are realized by devices alternating between two piconets. Therefore Bluetooth uses FH-CDMA (frequency hopping – code division multiple access) to distinguish piconets.

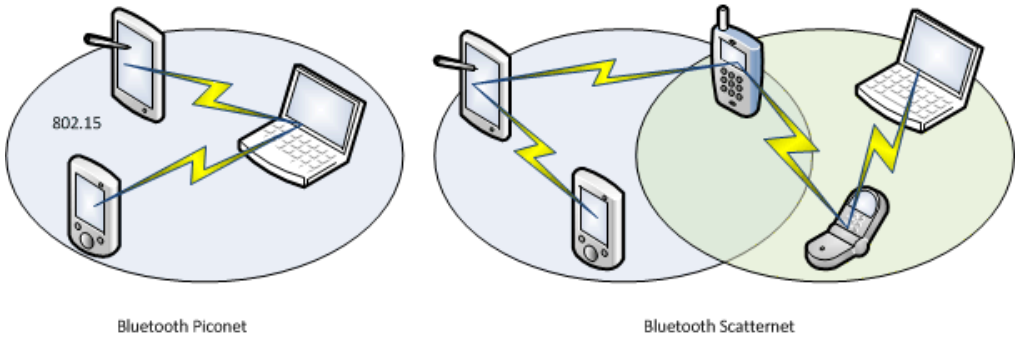


Figure 3-6: The Bluetooth network configuration: on the left a single piconet and on the right two overlapping piconets forming a scatternet

The core specification [70] covers the two lowest layers, the physical layer (PHY) and the medium access layer (MAC). The profile specifications deal with additional protocols and functions adapting Bluetooth to certain applications. The data transmission is realized as combination of FH and TDD (Time Division Duplex) with timeslot duration of 625 µs. Each timeslot has its own frequency. To modulate the carrier Gaussian frequency shift keying (GFSK) is used.

Bluetooth devices are available in three power classes. Class 1 emits with minimum 1 mW to maximum 100 mW and covers a range of 100 to 150 m. Most devices are class 2 devices. Those have an emission power of minimum 0.25 mW to maximum 2.5 mW and typically use 1 mW. With direct line of sight these devices cover 10 m. Class 3 devices only have an emission power of maximum 1 mW and are merely used for devices directly adjacent to each other.

Bluetooth offers a security mechanism to avoid the exchange of data between devices which have not been connected with each other before. To authorize the establishment of a connection a challenge-response protocol is implemented based on a secret PIN chosen by the user. Although this security concept is often bypassed by manufacturers by using fixed PINs it is almost impossible to force a connection with another device which has not been connected before. Unfortunately this security mechanism complicates the use of Bluetooth for the Peer-to-Peer Kalman filter since user interaction should be avoided. All other properties of Bluetooth make this standard an ideal candidate for this approach. It is widely accepted in today's smartphones, has a very limited transmission range (less than 10 m) and offers fast establishment of a local network between two or more peers. Nevertheless the security protocol affects seriously the idea of this approach. Therefore other WPANs have to be regarded.

3.2.3.2 ZigBee

The ZigBee alliance was formed in 2002 as an open non-profit association [74]. The alliance works tightly together with the accordant sub-working group 802.15.4 of IEEE. The main focus of this working group is the development of a protocol for data exchange with very low energy consumption. Possible applications are sensor linkage, remote control and similar applications which ask only for low data rates and small transmission ranges.

As depicted in Figure 3-7 the lower two protocols (MAC and PHY) are covered by the IEEE standard [75] while the upper levels are handled by the specification of the ZigBee Alliance [76]. Since the functionality of the layers of the protocol stack is not of much relevance here they will not be explained in detail. For more information on network protocol layers in general and specifically on the ZigBee protocol please refer to [17], [75], [76] and [77].

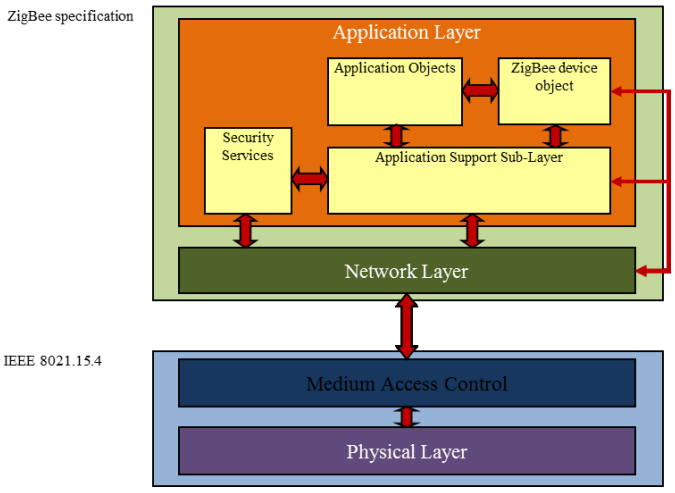


Figure 3-7: The ZigBee protocol stack

The ad-hoc network which is formed by ZigBee enabled devices is called a PAN (Personal Area Network) and comprises one PAN coordinator (PANC) which defines the network configuration. For ZigBee there are two types of devices. One type is the full-function device (FFD) and the other one is the reduced-function device (RFD). Only FFDs can act as PANC and are capable of establishing a network between several other FFDs and RFDs. In contrast to the RFDs a FFD implements the whole protocol stack while RFDs get along with a minimal implementation and are therefore only able to connect to other FFDs. The ZigBee specification supports two network topologies: the star-topology and the peer-to-peer topology.

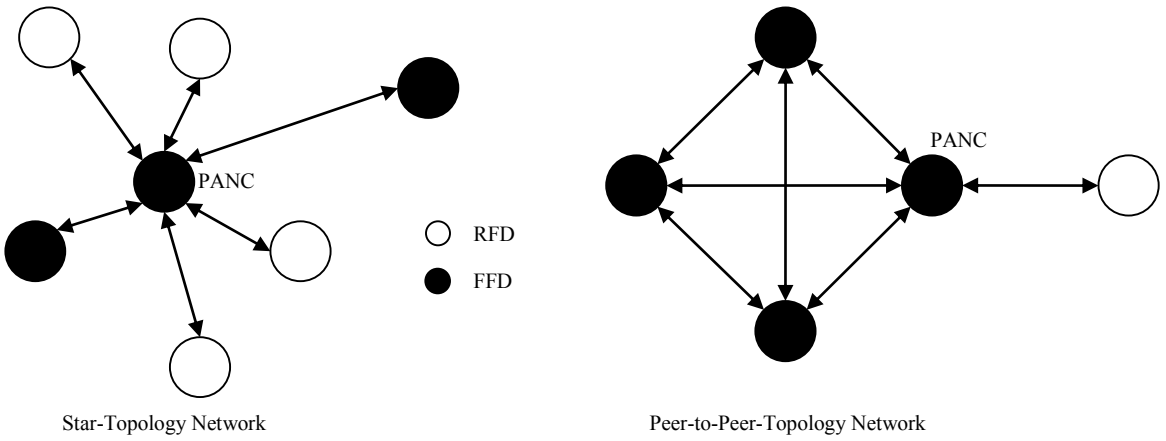


Figure 3-8: Possible network topologies of the ZigBee PAN. On the left side the Star-Topology and on the right side the Peer-to-Peer-Topology [77]

In the star-topology the communication takes place only via the PANC similar as in a Bluetooth piconet. The PANC is the initiation point of the network and other FFDs and RFDs connect to it. The PANC manages the star-topology. The peer-to-peer-topology allows the configuration or more complex networks enabling ad-hoc functionality and self-configuring networks. Each FFD is allowed to directly communicate with other FFDs in the same PAN without involving the PANC. Only RFDs have to be connected directly to the PANC.

The specification of IEEE 802.15.4 allows two frequency bands. One is the low band at 868 to 915 MHz, providing one channel with a raw data rate of 20 kbps which is modulated by binary phase shift keying (BPSK). The other one is in the ISM frequency band (2.4 GHz), providing 16 channels with a spacing of 5 MHz and a raw data rate of 250 kbps which is modulated by offset-quadrature phase shift keying (O-QPSK). Due to the operation in the ISM-band ZigBee is suspected to cause interferences with Bluetooth. However, the interference caused by ZigBee for WLAN is assumed to be much more critical. To access the radio interface CSMA/CA (carrier sense multiple access with collision avoidance) technique is used. This also means that the devices have to be synchronized to each other. The typical emission power of a ZigBee device is about 1 mW (similar to Bluetooth) and enables a transmission range of 8 m.

The MAC layer offers security functionality which can be harnessed by upper layers to achieve desired security level. In general three security levels are provided: no security, access monitoring and symmetric encryption. In contrast to Bluetooth this lack of security mechanism makes ZigBee an ideal candidate for the Peer-to-Peer Kalman filter since authentication mechanisms are optional and help to establish fast and anonymous networks between the peers. Also the peer-to-peer topology of the networks and the short transmission range fits the requirements of the approach. Problematic with ZigBee is that besides the publication of the standard in 2003 only few devices integrate ZigBee

functionality. Up to now no smartphone integrates ZigBee as communication protocol. This might be caused by the possible interferences with WLAN or Bluetooth and the pervasiveness of Bluetooth which to some extent provides the same functionality.

3.2.3.3 WLAN / Wi-Fi direct

The IEEE 802.11 standard [78] describes the two lowest layers of the network protocol stack (MAC and PHY) for data transmission via the radio interface. To be able to provide networks combining wired and wireless medium access the interface to the upper layers is standardized and the same for all devices implementing the protocol stack of the IEEE 802 working group which comprises for example local network solutions like Ethernet (802.3) or token rings (802.5) as well. Several extensions of the original standard denoted by lowercase letters increased the primary required transmission rate of 1 Mbit/s (optional 2 Mbit/s) via 54 Mbit/s (802.11a) to the today most common rate of 240 Mbit/s (802.11n). Currently 802.11 networks emit in the ISM frequency band (2.4 GHz) but for 802.11n also the 5 GHz frequency band can be used. Today's WLAN achieves a transmission range of 70 m within buildings and 250 m outdoors.

There exist two network modes as depicted in Figure 3-9. The most common one is the so called infrastructure-based network (left side). One access point (AP) which is connected to a distribution system (e.g. an Ethernet) serves one basic service set (BSS). Devices connect to the access point and are able to communicate with each other but only via the access point. Using data transmission via the distribution system devices of different BSSs can also exchange data with each other. Several BSSs which are connected via a distribution system are called an extended service set (ESS). Since this type of network is not suitable for the P2P Kalman filter the focus for WLAN is clearly on the second type of networks, the ad-hoc networks. In Figure 3-9 on the right side the idea of a WLAN ad-hoc network is depicted. Several devices communicating with each other form an independent basic service set (IBSS). A direct link from each device to each other device is possible and enables peer-to-peer communication. The IBSSs are either separated by space or by using different carrier frequencies. Unfortunately the configuration of such ad-hoc networks and stations which coordinate the network are not specified in a detailed way by the standard and therefore the implementation of a peer-to-peer network based on IEEE 802.11 is rather complicated and not user-friendly.

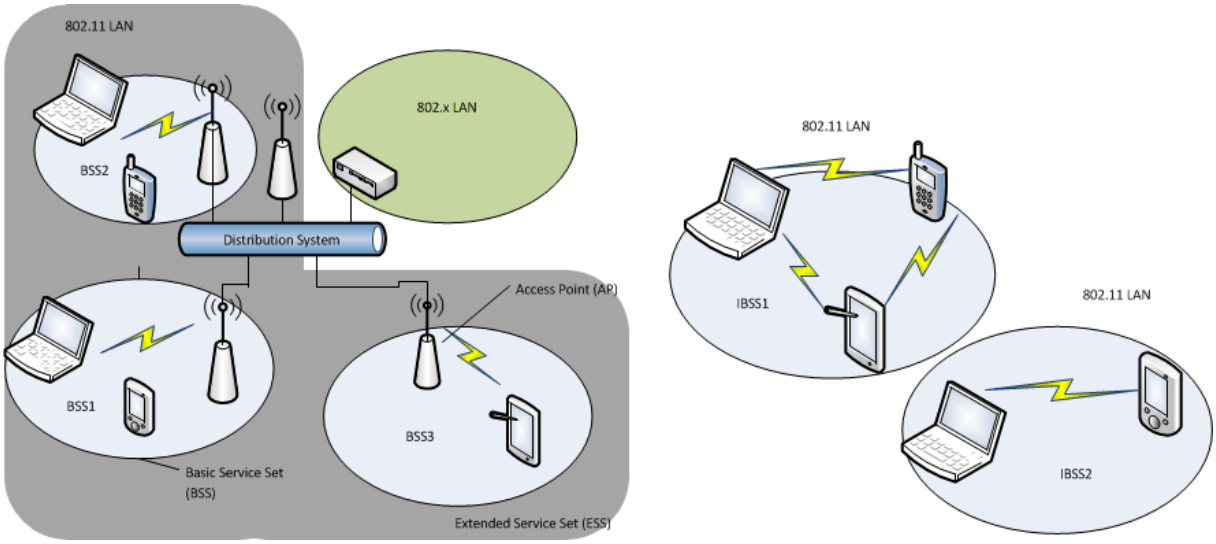


Figure 3-9: Two possible WLAN configurations: On the left side an infrastructure-based network with several basic service sets (BSS) forming an extended service set (ESS) and on the right side a WLAN peer-to-peer network forming two independent basic service sets (IBSS)

Only recently a new standard was developed based on the 802.11 standard specifically for managing peer-to-peer networks avoiding interaction with wireless access points. This standard is called Wi-Fi

direct. The term Wi-Fi was developed only for marketing reasons and has no real meaning but refers as pun to the term Hi-Fi (High-fidelity). Wi-Fi and WLAN can be used as synonyms for wireless local or metropolitan area networks. The Wi-Fi Alliance, founded 1999, is a consortium of more than 300 companies certifying products of different manufacturers based on the 802.11 standard [79].

Wi-Fi direct enables the establishment of peer-to-peer communication links for all Wi-Fi direct enabled devices and allows devices to act as access point for WLAN enabled but not Wi-Fi direct enabled devices to offer downward compatibility. The decision to establish a connection can be driven by an application, a management system or even the operating system of the mobile device. Each Wi-Fi direct certified device is also able to act as station within a common infrastructure-based WLAN. When forming a peer-to-peer network, so called groups, one device (host) is in charge for this group managing the leaving and joining of other devices of this group and the starting and ending of the group. All devices which are certified as WiFi direct compatible are able to manage the ad-hoc network. An optional feature is the Service Discovery which enables the advertisement of services supported by higher layer applications [80]. The current standard of IEEE to which Wi-Fi direct is compliant is 802.11n which results in a transmission rate of 54 Mbps and a range of 150 m indoors and 500 m outdoors. However the transmission rate is dependent on the distance to the other device; the further away a device is the more decreases the transmission rate [79]. Therefore the above mentioned range of 70 m indoors and 250 m outdoors is more likely.

Several companies like e.g. Intel and Broadcomm already accepted this new standard and integrated it into their devices. But of more interest was the announcement in October 2011 of Google to support Wi-Fi direct in their at this time latest Android system update, Android 4.0.

3.3 Simulation based on MATLAB

Before implementing the actual prototype application it has to be examined by simulations if the basic idea of the P2P Kalman filter - the mutual filtering - is beneficial for the position accuracy of the peers. The implementation of the Kalman filter algorithms (EKF and UKF) as described in 3.2.2 for the simulations performed in Matlab is described. Different scenarios are defined and the performance of both filter types is tested on their performance and their results compared to each other. The scenarios differ from each other regarding accuracy of the sensor measurements, distance to other peers, and the total amount of peers participating in the simulation. Especially the amount of the peers is of great interest regarding the position estimation since it is assumed that the position accuracy improves the more peers for mutual filtering are available. But also the influence of the other parameters has to be determined. The results of the different scenarios are therefore compared to each other and discussed in detail. Matlab is used to implement and perform the simulations since it provides pre-implemented functions, Matrix computation and the possibility to depict results and plots in an uncomplicated and fast way.

3.3.1 Simulation configuration

The general configuration of the simulation is explained in detail in the following paragraphs. The settings described here are the same for all scenarios and the different filter types.

3.3.1.1 Overview

Since the P2P Kalman filter is a peer-to-peer approach and therefore is based on data exchanged between large amounts of people, it is necessary to simulate a large amount of peers. Of course this kind of simulation is only realistic to a certain degree since the device is just occupied by navigating and does not receive phone calls or send SMS simultaneously. There are also no other networks that interfere with each other and the processing capacity for the mobile device is not limited. Nevertheless the simulation allows a view on the functionality of this approach.

To be able to compare the results of different filtering algorithms with each other and to evaluate if the amount of peers influences the position accuracy the tracks of all peers moving within an emulated indoor area are recorded once and used for all simulation scenarios. These tracks are stored as so called movement data. The recording of these tracks is done with a varying amount of peers: 100, 200 and 300. To obtain maximum comparability the movement data recorded with 100 peers is used in the simulations with 200 peers for the first 100 peers and the movement data recorded for 200 peers is used in the simulations with 300 peers. When numbering the peers consecutively the tracks of peer 1 to peer 100 are applied to the first 100 peers of simulations with 100, 200 and 300 peers. The movement data of 200 peers again is used in simulations with 300 peers. This means that for all simulation scenarios (100, 200, and 300) the tracks of peer 1 to peer 100 are the same, and for the scenarios with 300 peers the tracks of peer 101 to peer 200 are the same as for scenarios with 200 peers.

The emulated indoor area is a very simple construction, see Figure 3-10. It is a rectangular room with a size of 100 x 200 m with so called reference positions on each side. The reference positions are marked in red and indicate areas where the acquisition of a position estimated by GPS signals is possible to a certain degree of accuracy. The black stars within the rectangular are the peers numbered consecutively. Every peer has a randomly chosen start position within the indoor area. The initial heading and the individual step length of a peer are chosen randomly based on an assumed average step length of 0.7 m and an equal distribution of -0.1m to 0.1 m at the beginning. This individual step length is kept during the whole walk of a peer and only varied marginal. When a peer hits a wall it turns around by 90° and stays therefore inside the indoor area throughout the whole simulation.

The true position of the peer when taking a step is computed by (3-1) and (3-2). In this case the equations are not used as propagation function to model the position of a peer during dead reckoning. Instead they

provide the possibility to calculate the movement of the peers using a well-defined function for an algorithm. To emulate true walking the individual true step length and the heading is varied to a certain degree. These variations are randomly chosen for each step a peer does based on an equal distribution between -0.01m to 0.01 m of the individual step length and between -10° to 10° of the initial heading. The peer therefore does not walk in a straight line and is varying about its individual step length. The new true position is stored as x- and y-coordinate in a data structure called *timesteps* together with the true heading and the true step length.

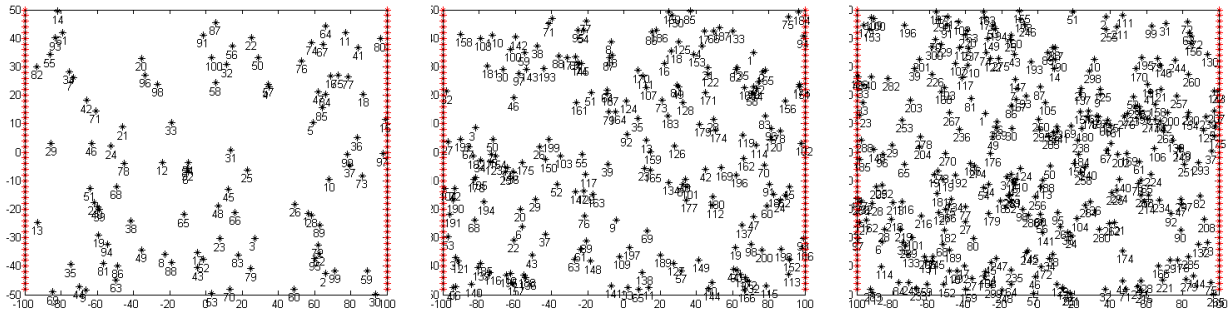


Figure 3-10: The emulated indoor area with 100, 200 and 300 peers. The red marked points indicate reference positions where a GNSS measurement update is possible.

An essential difference to the real world is the movement behavior of the peers. For the simulation they are all synchronized to each other i.e. all peers simultaneously take the next step. To keep the simulation simple the decision to take a step or not is also not provided; the peers never pause. In reality this decision would add another error to the application. If the step detector does not realize that the peer is walking or has stopped the estimated position would differ more strongly from the true position. To keep the simulation simple it is supposed that the step detector works perfectly and the peers are walking constantly. The length of a simulation run depends on the amount of steps. For all tests the peers take 1000 steps to ensure that true and estimated position diverge over time.

The peer of course is not aware of the true position only the estimated position and the erroneous measurements of the sensors are known to the mobile device; and the computation of the position is based solely on this knowledge. The estimated position of each peer for every simulation step is stored in the same data structure as the true position. Therefore it is easy to follow the particular difference between true position and estimated position in a simulation for each step a peer does. In Figure 3-11 the structure for data storing is depicted. For each simulation step (*timesteps(1)* to *timesteps(1000)*) another structure called *peer* contains a list of peers (from 1 to 100 in Figure 3-11) and their current values. In Table 3-1 the meaning of the variables is explained.

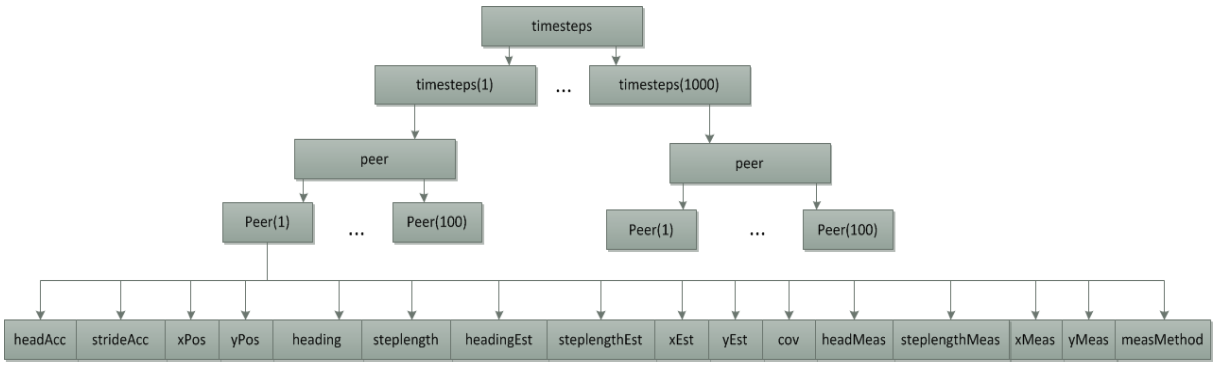


Figure 3-11: The data structure for storing all information about each peer during the whole simulation process. For each simulation step a variable peer is stored containing all peers (100 in this case).

Name of variable	Meaning
headAcc	The accuracy of the sensor measuring the heading (stays the same throughout the simulation)
strideAcc	The accuracy of the sensor measuring the step length ((stays the same throughout the simulation)
xPos	The true x-coordinate of the peer (varied in each simulation step)
yPos	The true y-coordinate of the peer (varied in each simulation step)
heading	The true heading of the peer (varied in each simulation step)
steplength	The true step length of the peer (varied in each simulation step)
headingEst	The estimated heading of the peer dependent on filtering and measurement
steplengthEst	The estimated step length of the peer dependent on filtering and measurement
xEst	The estimated x-coordinate
yEst	The estimated y-coordinate
cov	The covariance matrix
headMeas	The heading measured by the sensor
steplengthMeas	The step length measured by the sensor
xMeas	The x-coordinate measured during measurement update (other peer or reference position)
yMeas	The y-coordinate measured during measurement update (other peer or reference position)
measMethod	Indicates if the measurement update is based on the measurement of another peer, a reference position or only by the peer's sensors

Table 3-1: The variables contained in the data structure and their meanings

For the Unscented Kalman Filter more variables are added which contain the weighting parameters and the sigma points. These values are enumerated and explained in detail in 3.3.2.2.

3.3.1.2 Propagation and measurement prediction function

As stated before the true position, heading and step length are stored in the data structure *timesteps* for each peer and for each step. Based on the first true position the estimated position of the peer is computed. Since it is assumed that the peer comes from outside (regardless where she starts within the building) the first estimated position differs from the true position by -5 to +5 m according to the accuracy of GPS positioning which can normally be achieved with the GPS receiver of a smartphone. This estimated position is as well stored in *timesteps*. Based on this first estimated position as starting point for the dead reckoning process in each simulation step the following processes are carried out:

1) The propagation function computes the new position based on heading and step length using the propagation function (3-50). This is done for each peer.

2) Each peer searches in the neighborhood for other devices providing a position correction or for GNSS signals.

    2.1) If another peer is found within a certain transmission range the position data as well as the variance of this position data is exchanged. The filter algorithm uses the received data and the own estimation to provide a new position based on the output of the filtering algorithm. This new position is then used for further dead reckoning.

    2.2) The peer is near a reference position and therefore able to use the GPS receiver to compute a position. In this case two ways to handle this situation are conceivable. The first possibility would be to use the GPS retrieved position as new estimated position and base the further dead reckoning process on it or to use the GPS position and position accuracy as input for the filtering algorithm with the own estimated position to compute a new estimate and use this one as base for the further dead reckoning process (see 2.5.2.2.4). For these simulations the second approach is used.

    2.3) If no other peer is available and also a reference position is beyond reach the peer also processes the filtering algorithm but only with the step length and heading measurements. This is necessary to stabilize the position estimation of the peer since the heading estimation and the step length estimation are noisy.

3) The results of the filtering process are stored within the data structure as well as the measurements, the input from a reference position or another peer and the new covariance matrix. In the next step the whole process starts from the beginning.

It is important to mention that the peer is only allowed to perform a filtering process if there was no filtering with the same peer in the step before, otherwise the measurement input would be correlated. Therefore this has to be made sure before exchanging position data. This also applies to the situation if the peer exchanged data several steps before with one peer but did not filter her position in the meantime with another peer or a reference position. The last peer a measurement update was performed with has to be stored to avoid a second filtering.

The propagation and measurement functions are the same for all filter variations. The state vector which is computed by the propagation function consists of the four elements step length, heading, x-coordinate and y-coordinate. The propagation function is depicted below:

$$f(x_1; x_2; x_3; x_4) = x_1; x_2; x_3 + x_1 \cdot \sin(x_2); x_4 + x_1 \cdot \cos(x_2) \quad \textbf{(3-50)}$$

As can be easily seen the propagation function is based on the dead reckoning equations (3-1) and (3-2) for x- and y-coordinate which are element three and four of the state vector. For the heading and the step length the mobile device assumes that those are not changed i.e. the peer continues to walk in a straight line. As mentioned above the peer necessarily changes its direction and with that also her heading but since the propagation function is only a model of a real process this is the best approximation and it is supposed that sudden direction changes are measured by the sensors and correct the model. The heading is the second element of the state vector. As mentioned in [33] the step length tends to be relatively steady around an individual step length. Therefore the modeling of the step length by mapping its value always on itself is a suitable approximation of the real process. The individual step length is the first element of the state vector.

As can be seen from the list above three scenarios for the measurement update are possible. This means that also appropriate functions for the measurement prediction have to be found. The mobile phone is able to measure step length and heading either as output from the algorithm for step length estimation or directly from a sensor providing the orientation. The x- and y-coordinate can only be obtained when dealing with another peer or when the acquisition of satellite signals is available. Therefore two different measurement functions are sufficient. One is used in situations in which the peer only can rely on the measurements coming from its sensors (step length and heading) the other one when either another peer or a reference position are available. Also the coordinates coming from the second situation are directly used as measurement input. The weighting of the measurement input is derived from the measurement matrix $\mathbf{R}_k$ which elements are described in more detail in section 3.3.3. The two functions for the measurement update are depicted below. Equation (3-51) denotes the measurement prediction when only sensor measurement data is available; equation (3-52) when also coordinates either from another peer or a reference position are available.

$$h_1(x_1; x_2) = x_1; x_2 \tag{3-51}$$

$$h_2(x_1; x_2; x_3; x_4) = x_1; x_2; x_3; x_4 \tag{3-52}$$

3.3.1.3 Simulation parameters

In Table 3-2 below all other parameters necessary for the simulation are listed.

Name	Meaning	Value
q	standard deviation of process	0.1
$\mathbf{Q}_k$	covariance of process	$\begin{bmatrix} q^2 & 0 & 0 & 0 \\ 0 & q^2 & 0 & 0 \\ 0 & 0 & q^2 & 0 \\ 0 & 0 & 0 & q^2 \end{bmatrix}$
σ_{ref}	standard deviation of reference position	varied
d_{ref}	minimum distance to reference position for measurement update	varied
d_{peer}	minimum distance to other walker for measurement update	varied
h_{acc}	Value of the heading estimation accuracy	varied
l_{acc}	Mean value of the step length estimation accuracy	varied

Table 3-2: Parameters for the simulation

The values of these parameters are varied for different simulation runs. When discussing the simulation results the concrete values of the parameters are specified. The implementation of each filtering technique is described in the following sections. If values are changed which have been mentioned above the reasons and differences are explained within the succeeding sections.

3.3.2 Implementation of the Kalman filter in MATLAB

In 3.2.2 the equations of the EKF and the UKF were derived from the Kalman filter. Both implementations which are described here are accordant to these equations. Only small modifications had to be integrated for the EKF. For the UKF the case is different and several adaptions had to be deployed to enable the use of this filter type. Differences to equations in 3.2.2 are described in detail when discussing the implementation of the EKF and UKF for Matlab.

3.3.2.1 Simulation based on the Extended Kalman Filter

The filtering algorithm is implemented according to the equations (3-24) to (3-31), the propagation function as depicted in (3-50), and the measurement functions as depicted in (3-51) and (3-52). An example for the computation of the Jacobian matrices and the EKF was found on the MathWorks homepage [82]. This was used as base for the implementation to the P2P Kalman Filter. The Jacobian matrices are computed according to the following algorithm:

$$\mathbf{x} = (l, h, x, y) \tag{3-53}$$

$$f(\mathbf{x}) = l; h; x + l \cdot \sin(h\ \ ); y + l \cdot \cos(h\ \ )$$

$$\mathbf{z} = f(\mathbf{x})$$

$$n = \dim \mathbf{x}$$

$$m = \dim \mathbf{z}$$

$$\mathbf{A}_{(m,n)} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

$$h = n \cdot 2^{-52}$$

$$\forall k = 1, \dots, n$$

$$\mathbf{p} = \mathbf{x}$$

$$p_k = x_k + h \cdot (1)i$$

$$\mathbf{A}_{(m,k)} = \frac{\mathrm{Im}\big(f(\mathbf{p})\big)}{h}$$

The Jacobian matrix is computed in its own function to enable modularity since more than one Jacobian matrix has to be created. The algorithm is based on complex step differentiation [82]. A function $f(\mathbf{x})$ and the corresponding vector $\mathbf{x}$ are passed as parameters to this Jacobian-function. The output $\mathbf{z}$ of f is computed to obtain the vector dimension. In n and m the dimension of input $\mathbf{x}$ and the output $\mathbf{z}$ respectively are stored. The return value, the matrix $\mathbf{A}_{(m,n)}$ is initialized with zeros and a dimension based on n and m. The variable h is used to define the maximum computational accuracy for this calculation. The columns of $\mathbf{A}$ are then iteratively completed for each k. Therefore the imaginary part of the complex numbers of the function f of $\mathbf{p}$ is divided by the maximum computational accuracy. The matrix $\mathbf{A}_{(m,n)}$ is then returned to the calling process.

The Jacobian matrix has to be obtained two times for each peer and each step taken. It has to be computed when propagating the state vector $\hat{\hat{\mathbf{x}}}_k^-$ ahead in time. In this case the input is the *a posteriori* state vector

$\hat{\hat{\mathbf{x}}}_{k-1}$ from the step before and the propagation function. The result of this function is then the *a priori* estimation $\hat{\hat{\mathbf{x}}}_k^-$ and the design matrix $\mathbf{A}_k$ necessary to compute the *a priori* error covariance. Normally the Jacobian matrix $\mathbf{W}_k$ of f_k with respect to $\mathbf{w}_k$ process noise has to be obtained for this calculation as well but since it is assumed that the process noise is zero-mean the computation of $\mathbf{P}_k^-$ is reduced to:

$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1} \tag{3-54}$$

The second time the Jacobian matrix has to be computed is during the *measurement update* regardless if this is processed with another peer, a reference position or by sensor measurement. The input is then the according measurement prediction function and again the *a priori* estimation of vector $\hat{\hat{\mathbf{x}}}_k^-$. The return values are the output of the measurement prediction function applied to the *a priori* state vector and the design matrix $\mathbf{H}_k$. Both are then used to compute the Kalman gain and the *a posteriori* error covariance. The equation for the Kalman gain reduces as well to the equation below since $\vec{\mathbf{v}}_k$ is zero-mean.

$$\mathbf{K}_k = \mathbf{P}_k^-{\mathbf{H}_k}^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-{\mathbf{H}_k}^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{3-55}$$

The *a posteriori* state vector $\hat{\hat{\mathbf{x}}}_k$ is computed according to (3-27) and the a posteriori error covariance $\mathbf{P}_k$ according to (3-28). The process starts then from the beginning: a new state vector is propagated ahead in time and corrected by obtainable measurements.

To weight the quality of the measurements the measurement covariance matrix $\mathbf{R}_k$ has to be applied accordingly. As mentioned before the more the measurement is weighted the less the measurement prediction is trusted and vice versa. The quality of the sensor measurement or the accuracy of the x- and y-coordinate is therefore used as input for the matrix. Depending on the situation (two or four measurements) $\mathbf{R}_k$ is a 2-by-2 or a 4-by-4 matrix. On the main axis of $\mathbf{R}_k$ the standard deviation of the sensors and the quality of the received x- and y-coordinate are depicted.

$$\mathbf{R}_k = \begin{pmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_h^2 \end{pmatrix} \tag{3-56}$$

$$\mathbf{R}_k = \begin{pmatrix} \sigma_l^2 & 0 & 0 & 0 \\ 0 & \sigma_h^2 & 0 & 0 \\ 0 & 0 & \sigma_x^2 & 0 \\ 0 & 0 & 0 & \sigma_y^2 \end{pmatrix} \tag{3-57}$$

In equation (3-56) the measurement covariance matrix is a 2-by-2 matrix using the standard deviation of the step length estimator σ_l and the heading sensor σ_h which are assumed to be known as weighting for the residual between real measurement and prediction. The second equation is used for the *measurement update* with another peer or a reference position. Also here the first two elements of the main axis are the standard deviation of the step length estimator and the heading sensor. The last two elements depict the estimated standard deviation of the received position accuracy (σ_x and σ_y). If the position is obtained from a reference position the standard deviation of x and y is simply the accuracy of the measured GPS position. Usually this value can be queried from the GPS receiver. If the position is obtained from another peer the elements of the error covariance matrix of the other peer is used. The maximum distance between two peers is assumed to be 5 m according to the average cell size of a Bluetooth ad-hoc network. In 3.3.3 this distance is varied as well.

The process noise matrix $\mathbf{Q}_k$ is as well applied to the particular process model. $\mathbf{Q}_k$ gives information about the quality of the different parts of the process model. It weighs the components of the propagation function. It is assumed for this simulation that all parts of the propagation function are equally weighted therefore $\mathbf{Q}_k$ is not varied.

Since this is a simulation also the real measurements have to be simulated. For the measurement update with another peer the process is obvious: the position of the other peer which serves as measurement are erroneous enough to count as real noisy measurements. To the position of the reference position and the sensor measurements noise has to be added to get realistic measurements. For the sensor measurement randomized noise is added based on the known standard deviation of the particular sensor or algorithm (it is assumed that both are known to the mobile device) and added to the true value of the step length estimator and the heading sensor. In the same way it is dealt with the reference position; to the true reference position randomized noise is added based on the position accuracy that can be obtained in light-indoors.

3.3.2.2 Simulation based on the Unscented Kalman Filter

The implementation of the UKF is in general based on the equations described in 3.2.2.3 and on a tutorial of the MathWorks file exchange homepage on filtering with the UKF [83]. During the tests it became obvious that the UKF is not as easily adaptable for this approach as the EKF. The UKF is not able to deal with several successive measurements without the interception of a *time update* propagating the new state vector ahead in time. A new *time update* only takes place in the P2P Kalman filter if the user makes another step but before taking this step it is possible that more than one position measurement either from another peer and a reference position or from several peers are available. Applying more than one measurement within one *measurement update* the sigma points take wrong values and the position error is growing extremely.

Therefore two variations have been implemented for the UKF in Matlab to overcome this problem. The first variation allows only one position measurement even if more are available. Thereby a position update from a reference position is preferred to the position estimation of another peer and a peer nearer to the user is preferred to a peer farer away. The distance to a peer might be defined by the current signal strength in an actual implementation. The second variation allows all currently available measurements but intercepts those by a *time update* which uses a propagation function f_1 to map all elements of the state vector onto themselves since no step has been taken. Due to this the sigma points are re-computed and a new measurement can be applied.

$$f_1(\mathbf{x}) = l; h; x; y \tag{3-58}$$

Independently of the chosen variation of the UKF the initial sigma points are created based on the equations (3-31) to (3-33). The square root of the covariance matrix $\mathbf{P}_{ki}^{a}$ is computed using the pre-defined Matlab function *chol* for the Cholesky decomposition. To propagate the sigma points ahead in time the function f already described for the EKF is used if this is a normal *time update*. If the *time update* takes place to enable the application of successive measurements the function defined in (3-58) is used. Nevertheless equation (3-36) describes the way the sigma points are computed for both types of functions. Based on the new sigma points the *a priori* mean of the state vector and the covariance matrix are calculated based on equations (3-37) and (3-38). For this step the weighting parameters are necessary to rate the sigma points. The weighting factors are computed according to (3-47) to (3-49).

As for the EKF described above after one *time update* always a *measurement update* follows either in form of step length and heading only or in form of step length, heading, x-, and y-coordinate. If only sensor measurements are available measurement function h_1 (see equation (3-51)) is used to propagate the expected measurements ahead in time. If additionally a position measurement in form of x- and y-coordinate is available the second measurement function h_2 (see equation (3-52)) is applied. Using the

sigma points from the time update the particular function is processed to compute the expected measurements according to (3-39).

If more than one position measurement is available after a *time update* for the first variation of the implementation it is evaluated if this position update comes from a reference point (highest priority) or from other peers which are chosen according to their current distance to the peer. If the second variation is used a *time update* is automatically processed and the next *measurement update* follows.

Based on the sigma points computed by the measurement function the mean (3-40), the covariance matrix (3-41), and the cross-correlation matrix (3-42) are computed. As for the EKF the Kalman gain has to be obtained by comparing expected measurements coming from the measurement function and real measurement and weighting the comparison, see (3-43). The Kalman gain is then used to compute the *a posteriori* state vector and the *a posteriori* covariance matrix according to (3-44) and (3-45). The measurement noise covariance matrix $\mathbf{R}_k$ is the same as for the EKF as depicted in (3-56) and (3-57) depending on the amount of measurements.

The performance of both UKF variations is described in paragraph 3.3.3.2.

3.3.3 Simulation Results

The intention of these simulations is to evaluate if a better position accuracy can be obtained by using peer-to-peer filtering in GNSS-denied environments. As mentioned in 3.3.1.1 the peers are only able to compute a position based on GPS with certain accuracy at dedicated spots (reference positions) in the building. These places would be windows or doors in a real test environment. To be able to tell under which circumstance the filtering process is beneficial and under which it is not different scenarios are applied to the simulation. Mainly five factors as listed in Table 3-3 affect the position accuracy of the peers which can also be combined for the simulation.

Varied value	Description	Scenarios
Allowing filtering between peers	To evaluate the beneficial effect of filtering the simulations are processed with and without filtering	With peer update Without peer update
Amount of peers	The amount of peers within the simulated building is varied.	100 peers 200 peers 300 peers
Accuracy of stride length estimation	The ability of the smartphone to measure the step length correctly. This value stays the same throughout the whole simulation process for each peer individually. The accuracy is varied based on a randomized normal distribution around a certain value or on equal distribution within certain limits.	Normal distribution around 0.1m Normal distribution around 0.2m Equal distribution between 0.1m to 0.2m
Accuracy of heading estimation	The ability of the smartphone to measure the heading of the peer correctly. This value stays either the same for the whole simulation and is varied based on randomized normal distribution around a certain value or on equal distribution within certain limits. A scenario with growing heading error is also considered	Normal distribution around 10° Normal distribution around 30° Normal distribution around 45° Equal distribution between 10° to 45° Starting from 5° with randomized grow rate

Maximum distance to the peer	Normally this value should be defined by the maximum range of the transmission range of the used wireless communication technique. However it has to be tested how much distance is beneficial.	5 m 10 m

Table 3-3: Description of the different simulation scenarios.

For each scenario the amount of peers is varied according to Table 3-3 and processed with and without peer update. A simulation with a certain configuration is executed five times and the average from these five times is taken as result. As mentioned before the movement of the peers is recorded once to be able to reuse it for all simulation runs. For the run itself only the parameters depicted in Table 3-3 are applied to enable a comparison between the different scenarios. From Table 3-3 the following scenarios have been identified:

Description	Static parameters		Varied parameters		Scenario
Variation of the step length estimation accuracy	h_{acc}	$\mu = 30°$ $\sigma = 10°$	l_{acc}	$\mu = 0.1\text{m}$ $\sigma = 0.05\text{m}$ $\mu = 0.15\text{m}$ $\sigma = 0.05\text{m}$ $\mu = 0.2\text{m}$ $\sigma = 0.05\text{m}$	Scenario 1 Scenario 2 Scenario 3
	d_{peer}	5m			
	d_{ref}	1m			
Variation of the heading estimation accuracy	l_{acc}	$\mu = 0.1m$ $\sigma = 0.05m$	h_{acc}	$\mu = 10°$ $\sigma = 10°$ $\mu = 30°$ $\sigma = 10°$ $\mu = 45°$ $\sigma = 10°$	Scenario4 Scenario 5 Scenario 6
	d_{peer}	5m			
	d_{ref}	1m			
Equal distribution of heading and step length estimation accuracy	d_{peer}	5m	h_{acc}	$[10°, 45°]$	Scenario 7
	d_{ref}	1m	l_{acc}	$[0.1\text{m}, 0.2\text{m}]$	
Decreasing heading estimation accuracy	l_{acc}	$\mu = 0.1m$ $\sigma = 0.05m$	h_{acc}	Start $\mu = 5°$ $\sigma = 2°$ Grow rate $\mu = 0.1°$ $\sigma = 0{,}05°$	Scenario 8
	d_{peer}	5m			
	d_{ref}	1m			
Increasing maximum distance to other peer	l_{acc}	$\mu = 0.1m$ $\sigma = 0.05m$	d_{peer}	10m	Scenario 9
	h_{acc}	$\mu = 30°$ $\sigma = 10°$			
	d_{ref}	1m			

Table 3-4: Scenario description of the simulation for the P2P Kalman filter

To verify the applicability and performance of the EKF several simulations are carried out. On the one hand this is necessary to evaluate if the position accuracy increases in principle and on the other hand it was difficult to obtain information about the quality of the sensors and therefore about the measurement accuracy. As stated in 2.5.2.2.1 no information could be found but it is possible to evaluate the accuracy, see 3.5.1 for the purchased devices. It is therefore difficult to develop an error model especially because the step length and the heading cannot be directly measured but are merely computed based on measurements. The accuracy of the measurements is varied according to the different scenarios.

The simulation results are organized primarily by the filter type (EKF and UKF) and secondarily according to the different scenarios defined in Table 3-4. For each scenario a plot depicting the frequency distribution of the position error in meter is created. In this plot the error (computed as Euclidian distance between estimated and true position) for 300, 200 and 100 peers with P2P update and for 100 peers without P2P update are computed to evaluate if the position improves when more peers are available. The reason for using only the data of 100 peers without P2P update is that when no mutual filtering is allowed the amount of peers should not influence the position accuracy. To assure this assumption for each scenario a table is added containing the average position error of all peers for simulations with 100, 200 and 300 peers with and without mutual position update. Additionally the results of scenarios 1 to 3 and scenarios 4 to 6 are compared to identify which of the two measurements (step length or heading) is the more critical issue.

3.3.3.1 Extended Kalman Filter

The following sections comprise the results of the different scenarios defined in Table 3-4 for the Extended Kalman Filter.

3.3.3.1.1 Scenario 1

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1\text{m}$ $\sigma = 0.05\text{m}$

As expected the position accuracy for all peers improves when the amount of peers increases. This is illustrated in the plot in Figure 3-12 and also in Table 3-5. It was also assumed that if no mutual filtering is allowed the average error is similar for the simulations with 100, 200 and 300 peers. According to Table 3-5 this is true for the simulations with 100 and 200 peers but not for those with 300 peers. According to the results of the other scenarios this is only an outlier.

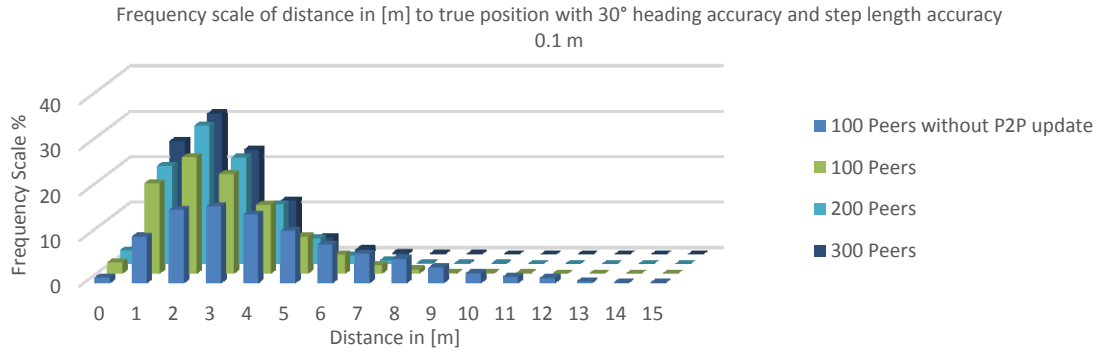


Figure 3-12: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy of 0.1m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.85m	2.52m	2.33m
Without P2P filtering	4.38m	4.05m	6.32m

Table 3-5: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.1.2 Scenario 2

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.15$m $\sigma = 0.05$m

The overall position error for this scenario of course is worse compared to the first scenario due to the worse accuracy of the step length estimation. Nevertheless the overall position accuracy improves with more peers (compare Figure 3-13 and Table 3-6).

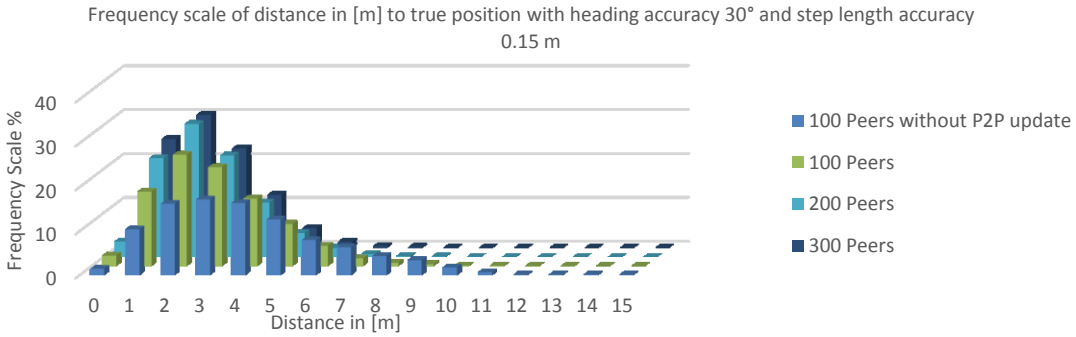


Figure 3-13: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy 0.15m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.98m	2.48m	2.37m
Without P2P filtering	4.18m	4.07m	4.14m

Table 3-6: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.15m

3.3.3.1.3 Scenario 3

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.2$m $\sigma = 0.05$m

Again compared to scenario 1 and 2 the position error increases due to the less accurate estimation of the step length. However, the differences are very small and in case of the simulation with 300 peers even better.

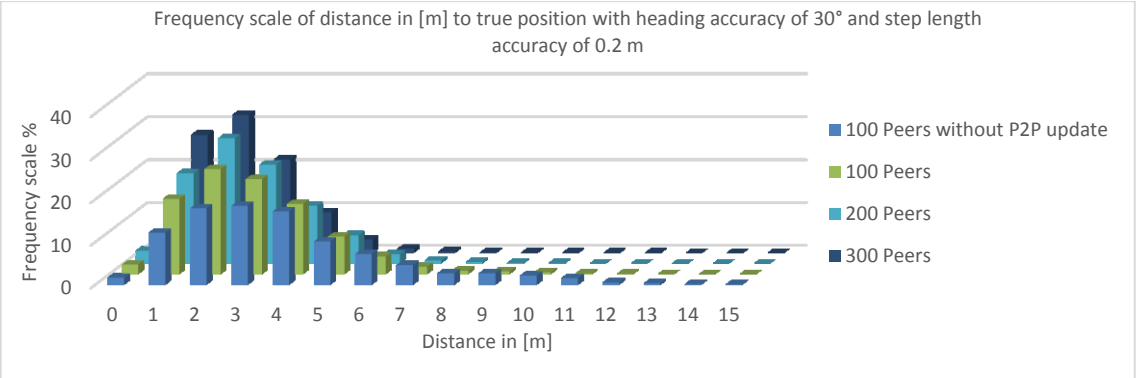


Figure 3-14: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy 0.2m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.99m	2.58m	2.21m
Without P2P filtering	3.99m	4.07m	4.12m

Table 3-7: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.2m

3.3.3.1.4 Summary of Scenario 1, Scenario 2, and Scenario 3

As expected when comparing the first three scenarios with each other for 100, 200 and 300 peers the position accuracy increases the better the step length estimation is. However, when considering Table 3-5 to Table 3-7 the difference in position accuracy between the scenarios is not as strong as the difference caused by the amount of peers. This becomes also obvious when comparing all results of the first three scenarios in Table 3-8. Apparently the influence of the accuracy of the step length estimation has not such a strong impact on the position accuracy. The plots in Figure 3-15 to Figure 3-17 illustrate this.

		100 peers	200 peers	300 peers
Scenario 1 (heading est. acc. 30°, step length est. acc. 0.1m)	With P2P filtering	2.85m	2.52m	2.33m
	Without P2P filtering	4.38m	4.05m	6.32m
Scenario 2 (heading est. acc. 30°, step length est. acc. 0.15m)	With P2P filtering	2.98m	2.48m	2.37m
	Without P2P filtering	4.18m	4.07m	4.14m
Scenario 3 (heading est. acc. 30°, step length est. acc. 0.2m)	With P2P filtering	2.99m	2.58m	2.21m
	Without P2P filtering	3.99m	4.07m	4.12m

Table 3-8: Summary of scenarios 1 to 3

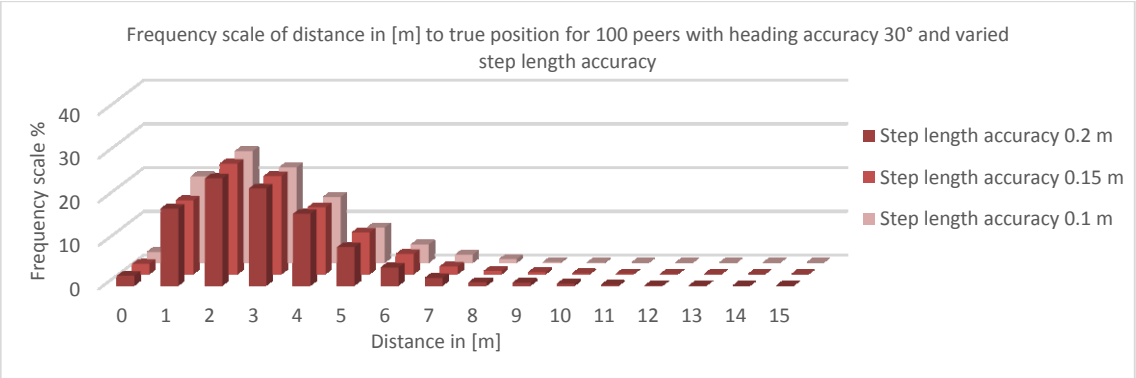


Figure 3-15: Frequency distribution of position error for 100 peers with heading estimation accuracy 30° and varying step length estimation accuracy

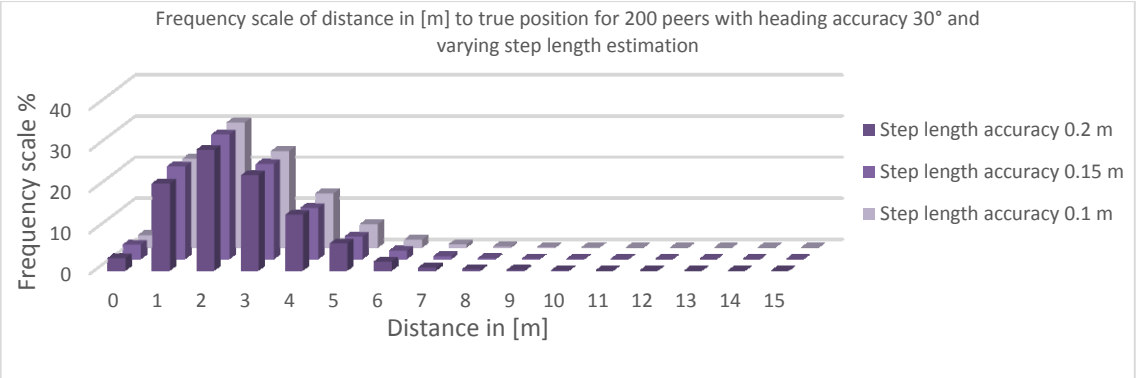


Figure 3-16: Frequency distribution of position error for 200 peers with heading estimation accuracy 30° and varying step length estimation accuracy

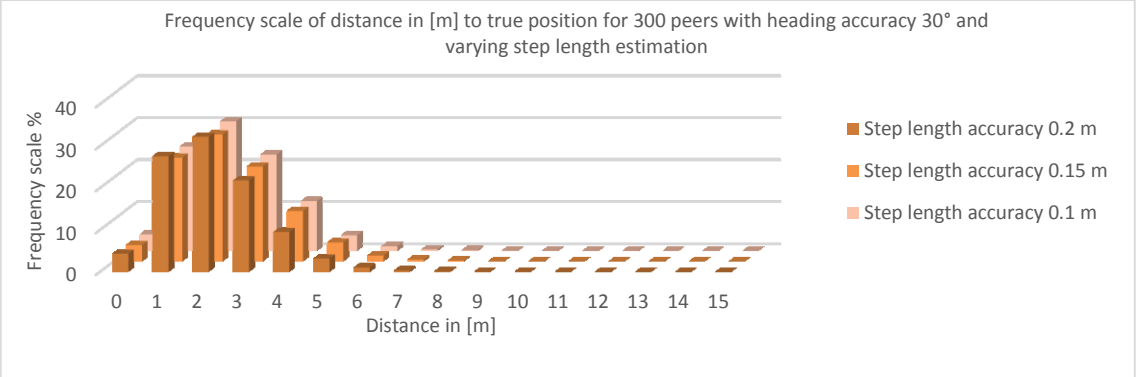


Figure 3-17: Frequency distribution of position error for 300 peers with heading estimation accuracy 30° and varying step length estimation accuracy

3.3.3.1.5 Scenario 4

h_{acc}	$\mu = 10°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1$m $\sigma = 0.05$m

For this scenario the position accuracy grows when more peers are residing within the indoor area and allowing mutual filtering. In general the results from this simulation are better than those of scenarios 1 to 3 since the estimation of the heading can be performed more accurate.

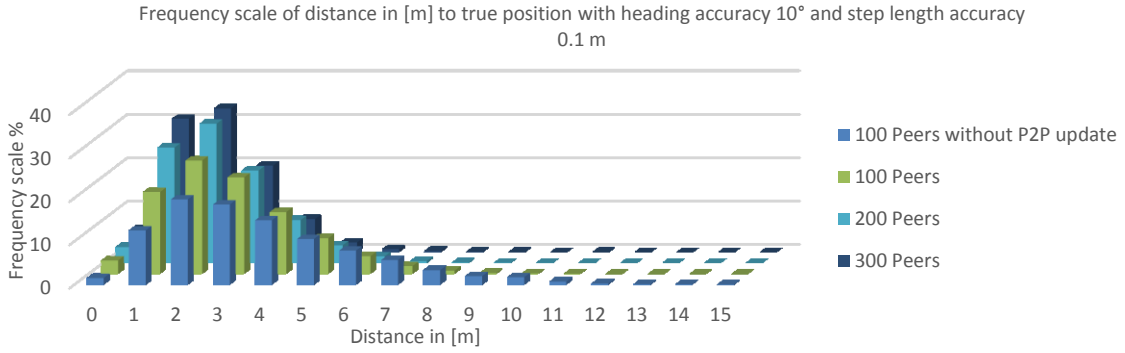


Figure 3-18: Frequency distribution of the position error in meter with 10° heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.83m	2.29m	2.06m
Without P2P filtering	3.87m	3.87m	3.91m

Table 3-9: Average distance in [m] to the true position with 10° heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.1.6 Scenario 5

Refer to scenario 1 in 3.3.3.1.1. The parameters are the same as for this scenario.

3.3.3.1.7 Scenario 6

h_{acc}	$\mu = 45°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1$m $\sigma = 0.05$m

The overall distance to the true position increases compared to scenario 4 and 5 due to the less accurate heading estimation but it decreases with more peers available for filtering.

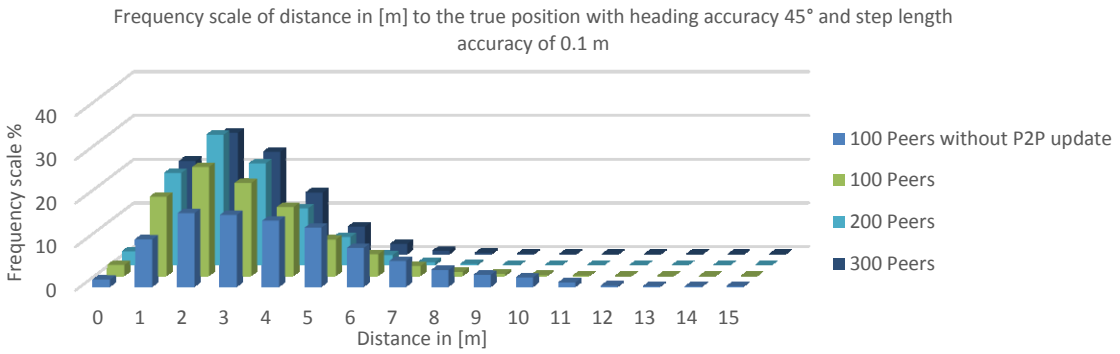


Figure 3-19: Frequency distribution of the position error in meter with 45° heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.96m	2.55m	2.59m
Without P2P filtering	4.15m	4.21m	4.45m

Table 3-10: Average distance in [m] to the true position with 45° heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.1.8 Summary of Scenario 4, Scenario 5, and Scenario 6

Compared to scenarios 1 to 3 it becomes obvious that the heading estimation accuracy has a stronger impact on the position accuracy than the step length estimation accuracy. In Table 3-11 one can see that the result are getting worse according to the degradation in the heading estimation accuracy. Compared to the results in Table 3-8 the degradation of the overall position error is more dependent on the heading estimation than on the step length estimation. Although for these scenarios the differences are very small (only centimeters) there is a definite tendency especially when regarding the results with 300 peers. This also becomes obvious in Figure 3-20 to Figure 3-22.

		100 peers	200 peers	300 peers
Scenario 4 (heading est. acc. 10°, step length est. acc. 0.1m)	With P2P filtering	2.83m	2.29m	2.06m
	Without P2P filtering	3.87m	3.87m	3.91m
Scenario 5 (heading est. acc. 30°, step length est. acc. 0.1m)	With P2P filtering	2.85m	2.52m	2.33m
	Without P2P filtering	4.38m	4.05m	6.32m
Scenario 6 (heading est. acc. 45°, step length est. acc. 0.1m)	With P2P filtering	2.96m	2.55m	2.59m
	Without P2P filtering	4.15m	4.21m	4.45m

Table 3-11: Summary of scenarios 4 to 6

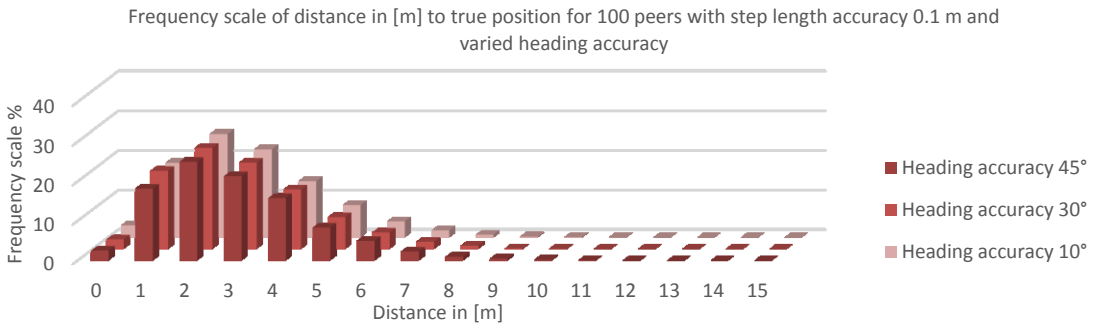


Figure 3-20: Frequency distribution of position error for 100 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m

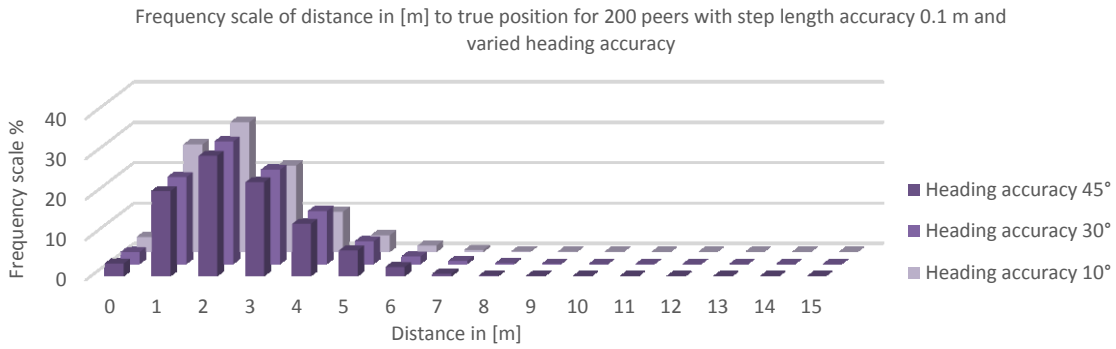


Figure 3-21: Frequency distribution of position error for 200 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m

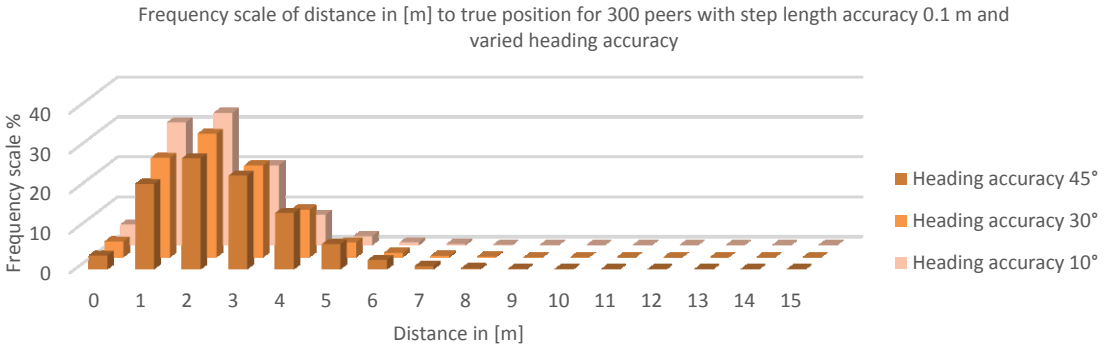


Figure 3-22: Frequency distribution of position error for 300 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.1.9 Scenario 7

h_{acc}	$[10°, 45°]$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$[0.1\text{m}, 0.2\text{m}]$

As depicted in Figure 3-23 and in Table 3-12 peers which have less accurate possibilities to measure heading and step length do not degrade peers with better sensors. The overall position accuracy is better than without peer-to-peer filtering and improves for each additional 100 peers. Also the position error in this scenario is not worse than for the other scenarios.

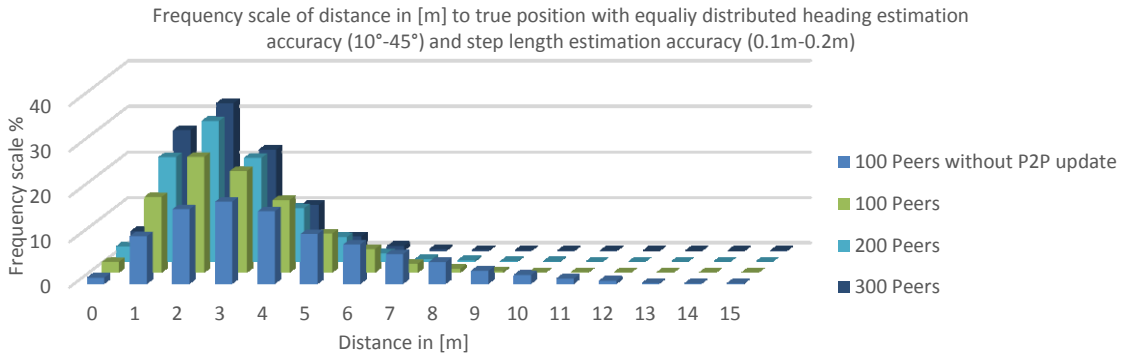


Figure 3-23: Frequency distribution of the position error in meter with equally distributed heading estimation accuracy (10° to 45°) and step length estimation accuracy (0.1m to 0.2m) for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	2.98m	2.45m	2.24m
Without P2P filtering	4.20m	4.16m	4.14m

Table 3-12: Average distance in [m] to the true position with equally distributed heading estimation accuracy (10° to 45°) and step length estimation accuracy (0.1m to 0.2m)

3.3.3.1.10 Scenario 8

h_{acc}	Start	$\mu = 5°$
		$\sigma = 2°$
	Grow rate	$\mu = 0.1°$
		$\sigma = 0{,}05°$
d_{peer}	5m	
d_{ref}	1m	
l_{acc}	$\mu = 0.1\text{m}$	
	$\sigma = 0.05\text{m}$	

For this scenario it is assumed that the heading estimation accuracy deteriorates with each step. This is a very common behavior especially when measuring the heading with gyroscopes and not by magnetometer measurements. The heading accuracy for all peers is normally distributed around a 5°-bias and with a standard deviation of 2°. With each step the accuracy is decreased with a mean of 1° and with a standard deviation of 0.05°. Also here as depicted in Figure 3-24 the position error becomes less the more peers are present.

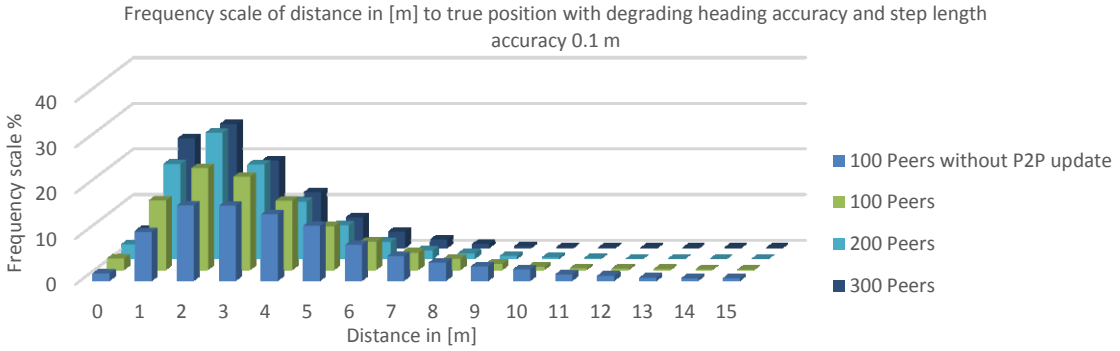


Figure 3-24: Frequency distribution of the position error in meter with growing heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers

	100 peers	200 peers	300 peers
With P2P filtering	3.40m	2.81m	2.67m
Without P2P filtering	4.55m	4.65m	4.77m

Table 3-13: Average distance in [m] to the true position with growing heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.1.11 Scenario 9

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	10m
d_{ref}	1m
l_{acc}	$\mu = 0.1$m $\sigma = 0.05$m

For this scenario the maximum distance to the other peers for performing a filtering algorithm is increased from 5 m to 10 m. This means that the amount of measurements increases for each peer. Therefore with each additional 100 peers more measurements are available. But those measurements are also from peers which are further away and whose position estimation differs stronger from the estimation of the receiving peer than in the scenarios before. Obviously when regarding the values in Table 3-25 and in Figure 3-25 this causes a deterioration of the position accuracy when more peers are available. While in the scenarios before the average position error decreased with more peers it is growing in this scenario. This means for the implementation of a P2P filtering prototype that the transmission range of the peers has to be limited to make sure that the position accuracy is increasing. As already mentioned in 3.2.3 this is not an easy task but when regarding the values below absolutely necessary.

	100 peers	200 peers	300 peers
With P2P filtering	3.19m	6.52m	12.68m
Without P2P filtering	3.98m	4.25m	4.12m

Table 3-14: Average distance in [m] to the true position with heading estimation accuracy 30° and step length estimation accuracy 0.1m. The maximum distance to the other peers for filtering is 10m

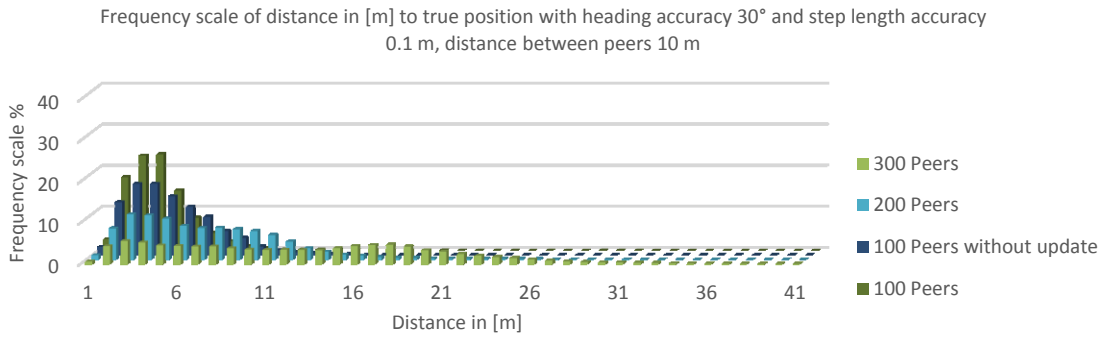


Figure 3-25: Frequency distribution of the position error in meter with heading estimation accuracy 30° and step length estimation accuracy 0.1m for 100, 200 and 300 peers (maximum distance to peer 10m)

3.3.3.2 Unscented Kalman Filter

Compared to the results of the Extended Kalman Filter the results for both variations of the Unscented Kalman Filter are much worse. Although the use of the P2PKF enables a better position accuracy as can be seen in the following tables the position error is always larger than in the simulations using the EKF. But due to allowing only one measurement update for the UKF1 the position accuracy does not increase with the amount of peers as could be observed for the EKF. And although for the UKF2 more measurement updates are allowed also here the position accuracy stays the same independently of the amount of peers. Only the overall position error can be reduced but by a small amount of around 2 m compared to the simulations where P2P filtering was not allowed. Regarding the simulations performed without P2P filtering the position accuracy does not benefit as much from using the UKF to filter the sensor measurements as from using the EKF. The position error in total is larger than for the simulations using the EKF. In general the second version of the UKF works a bit better than the first version.

3.3.3.2.1 Scenario 1

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1\text{m}$ $\sigma = 0.05\text{m}$

As illustrated by the figures in Table 3-15 the position error decreases compared to simulations where no P2P filtering is allowed. Nevertheless the position error is not influenced by the amount of peers regardless if the first or the second variation of the UKF is used. The results are much worse in either case than those of the simulations based on the EKF.

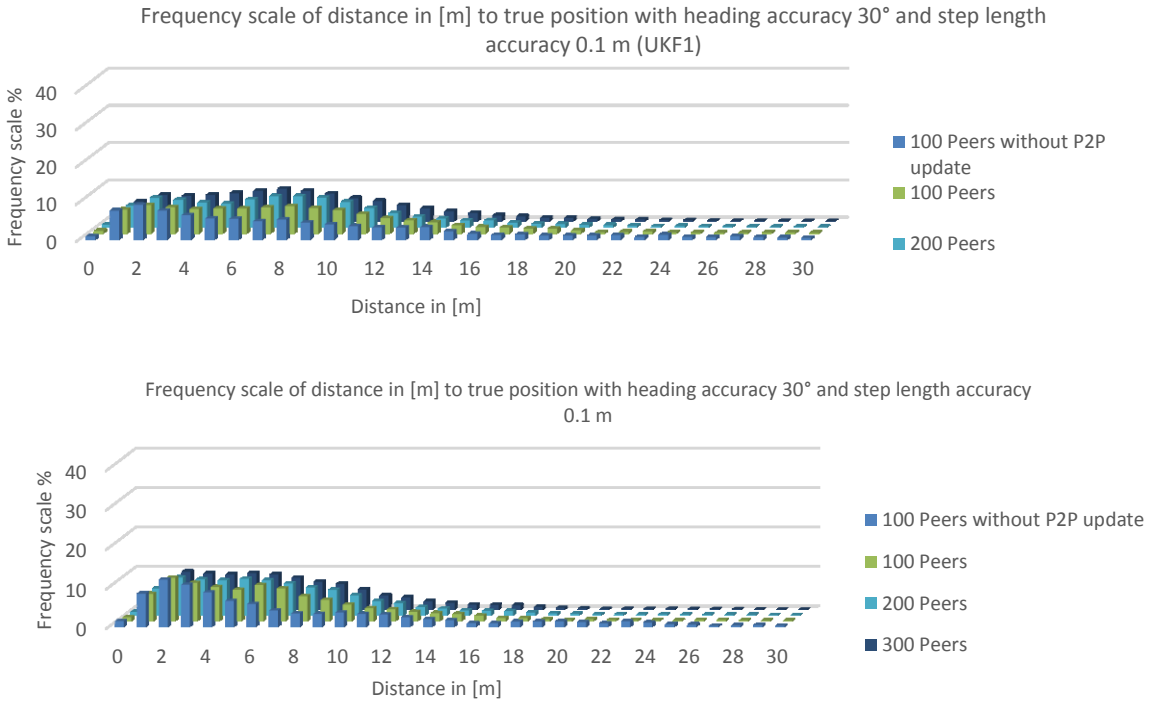


Figure 3-26: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy of 0.1m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	8.56m	8.05m	8.05m
	Without P2P filtering	10.16m	10.60m	10.74m
UKF2	With P2P filtering	7.27m	7.07m	7.43m
	Without P2P filtering	9.15m	9.53m	9.64m

Table 3-15: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.2.2 Scenario 2

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.15\text{m}$ $\sigma = 0.05\text{m}$

Compared to the same simulations using the EKF also the results for this scenario are much worse. As expected the difference to scenario 1 is not this large since the position error is not as strongly influenced by errors in the step length estimation than by errors in the heading estimation. This is the same as for the EKF. However, also in this scenario the position error does not increase with more peers.

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	8.51m	8.11m	8.49m
	Without P2P filtering	10.29m	10.61m	10.69m
UKF2	With P2P filtering	7.25m	7.21m	7.77m
	Without P2P filtering	9.17m	9.49m	9.61m

Table 3-16: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.15m

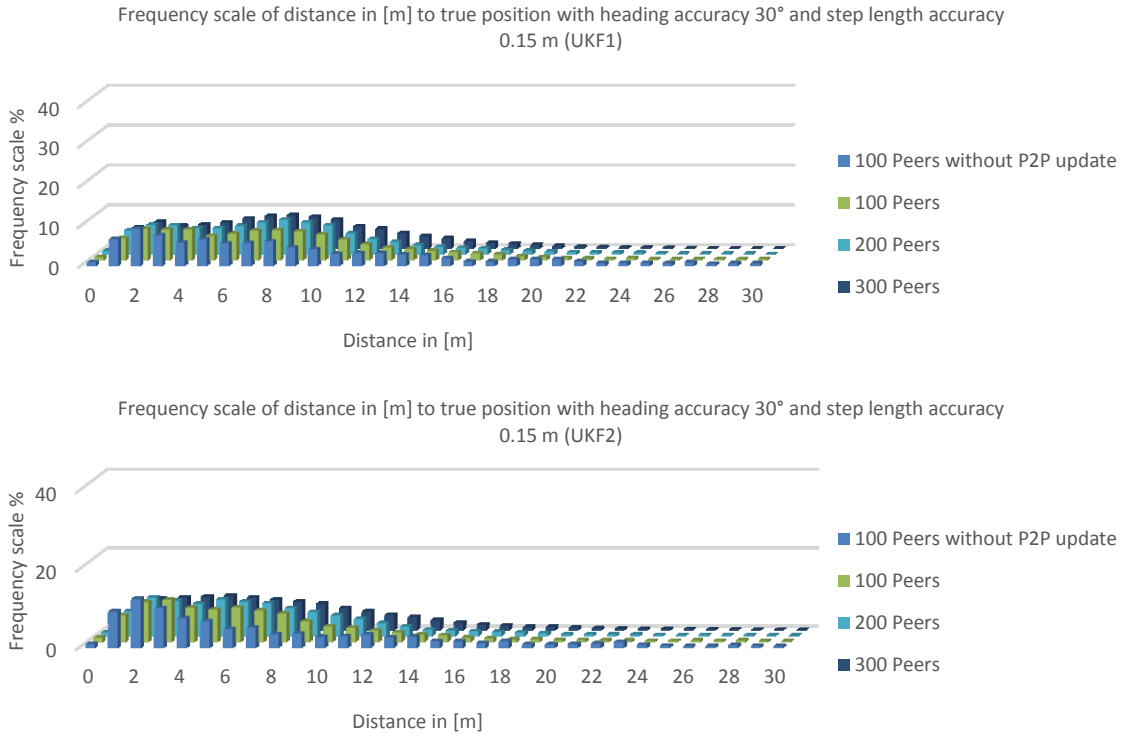


Figure 3-27: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy of 0.15m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

3.3.3.2.3 Scenario 3

h_{acc}	$\mu = 30°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.2$m $\sigma = 0.05$m

Again the position error does not improve with an increasing amount of peers. The differences to scenarios 1 and 2 are only marginal.

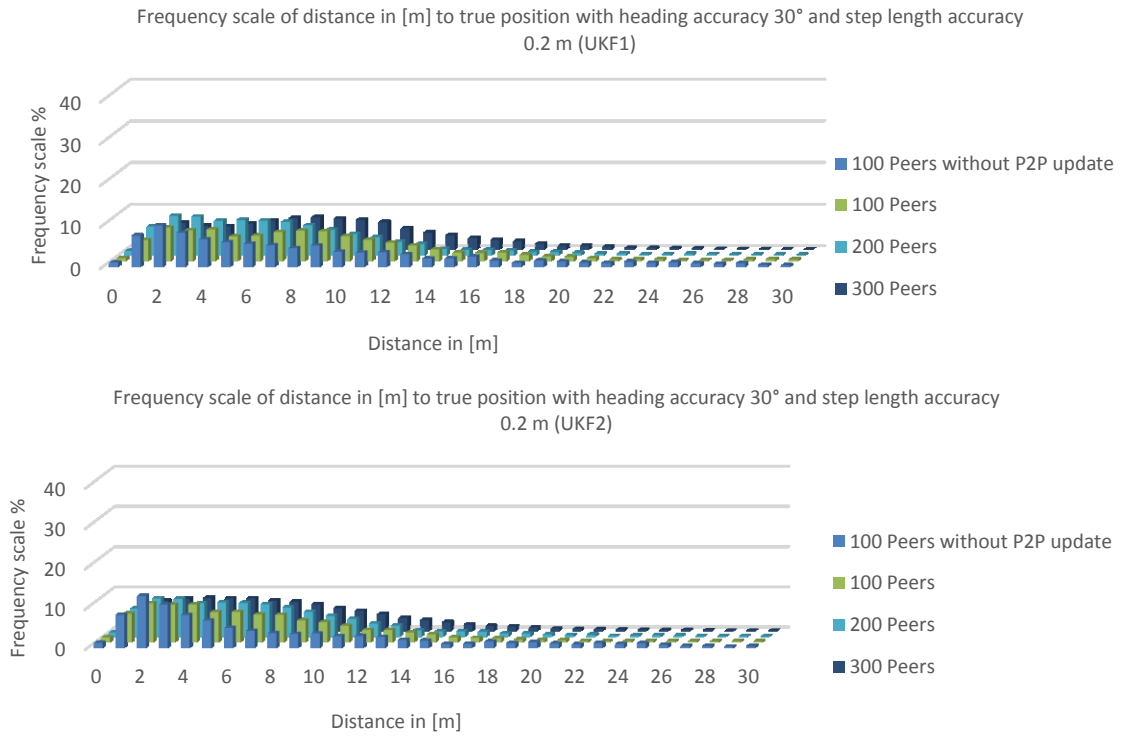


Figure 3-28: Frequency distribution of the position error in meter with 30° heading estimation accuracy and step length estimation accuracy of 0.2m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	8.72m	7.56m	8.83m
	Without P2P filtering	10.20m	11.42m	10.53m
UKF2	With P2P filtering	7.60m	7.25m	8.01m
	Without P2P filtering	9.19m	9.47m	9.51m

Table 3-17: Average distance in [m] to the true position with 30° heading estimation accuracy and step length estimation accuracy 0.2m

3.3.3.2.4 Summary of Scenario 1, Scenario 2, and Scenario 3

Summarizing scenario 1 to 3 produces similar results for the UKF variations compared to those of the EKF. Obviously the influence of the step length estimation is not as essential for the overall position accuracy as the heading estimation. For the UKF variations the amount of peers does not have the same strong effect as for the EKF and therefore the overall position accuracy is very similar for all scenarios and the amount of peers. This is also depicted in the plots in Figure 3-29 to Figure 3-31. The frequency distribution of the error is similar for all different scenarios.

			100 peers	200 peers	300 peers
Scenario 1 (heading est. acc. 30°, step length est. acc. 0.1m)	UKF1	With P2P filtering	8.56m	8.05m	8.05m
		Without P2P filtering	10.16m	10.60m	10.74m
	UKF2	With P2P filtering	7.27m	7.07m	7.43m
		Without P2P filtering	9.15m	9.53m	9.64m
Scenario 2 (heading est. acc. 30°, step length est. acc. 0.15m)	UKF1	With P2P filtering	8.51m	8.11m	8.49m
		Without P2P filtering	10.29m	10.61m	10.69m
	UKF2	With P2P filtering	7.25m	7.21m	7.77m
		Without P2P filtering	9.17m	9.49m	9.61m
Scenario 3 (heading est. acc. 30°, step length est. acc. 0.2m)	UKF1	With P2P filtering	8.72m	7.56m	8.83m
		Without P2P filtering	10.20m	11.42m	10.53m
	UKF2	With P2P filtering	7.60m	7.25m	8.01m
		Without P2P filtering	9.19m	9.47m	9.51m

Table 3-18: Summary of scenarios 1 to 3 for the UKF1 and UKF2

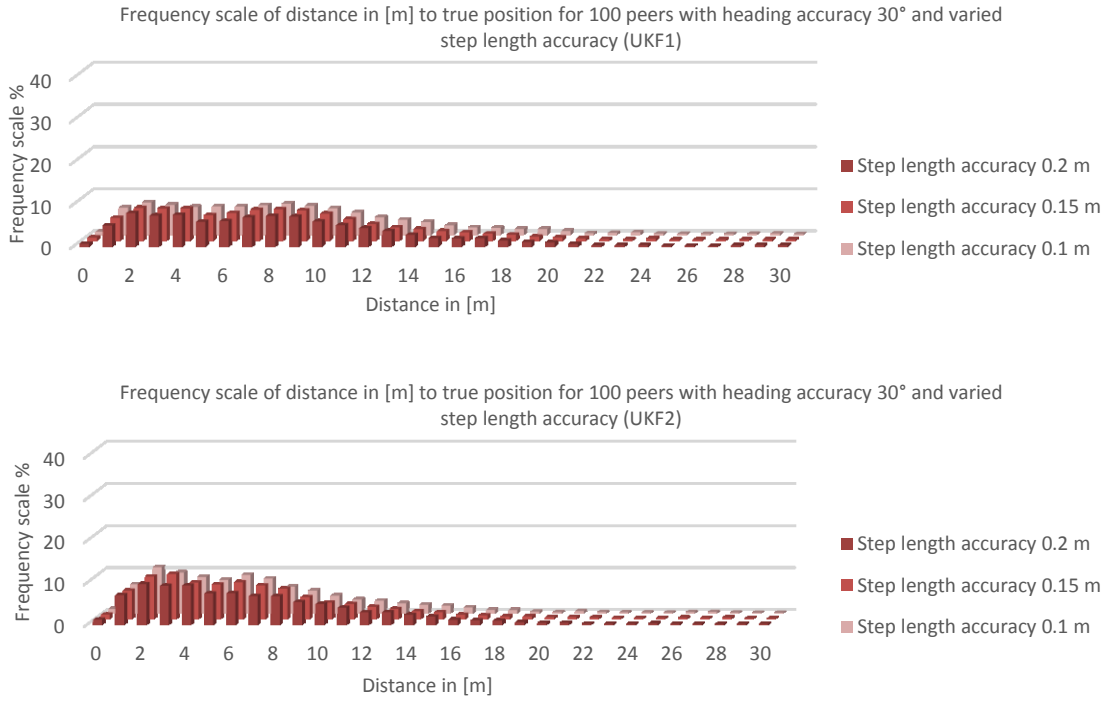


Figure 3-29: Frequency distribution of position error for 100 peers with heading estimation accuracy 30° and varying step length estimation accuracy. Top UKF1, bottom UKF2

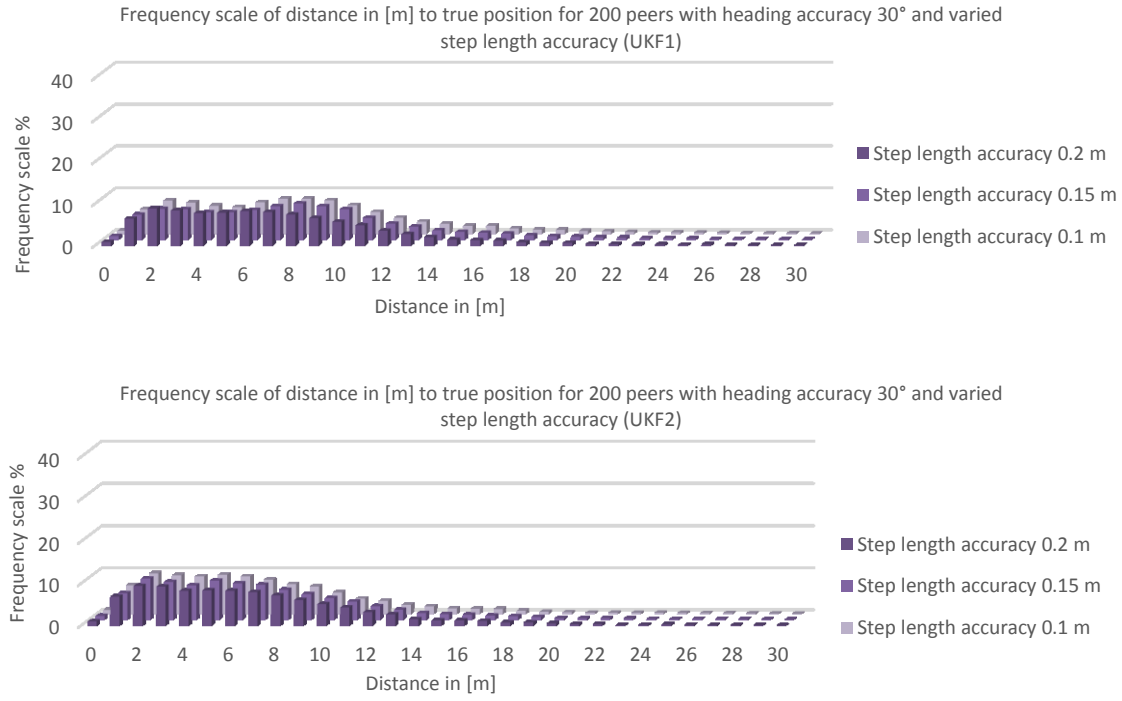


Figure 3-30: Frequency distribution of position error for 200 peers with heading estimation accuracy 30° and varying step length estimation accuracy. Top UKF1, bottom UKF2

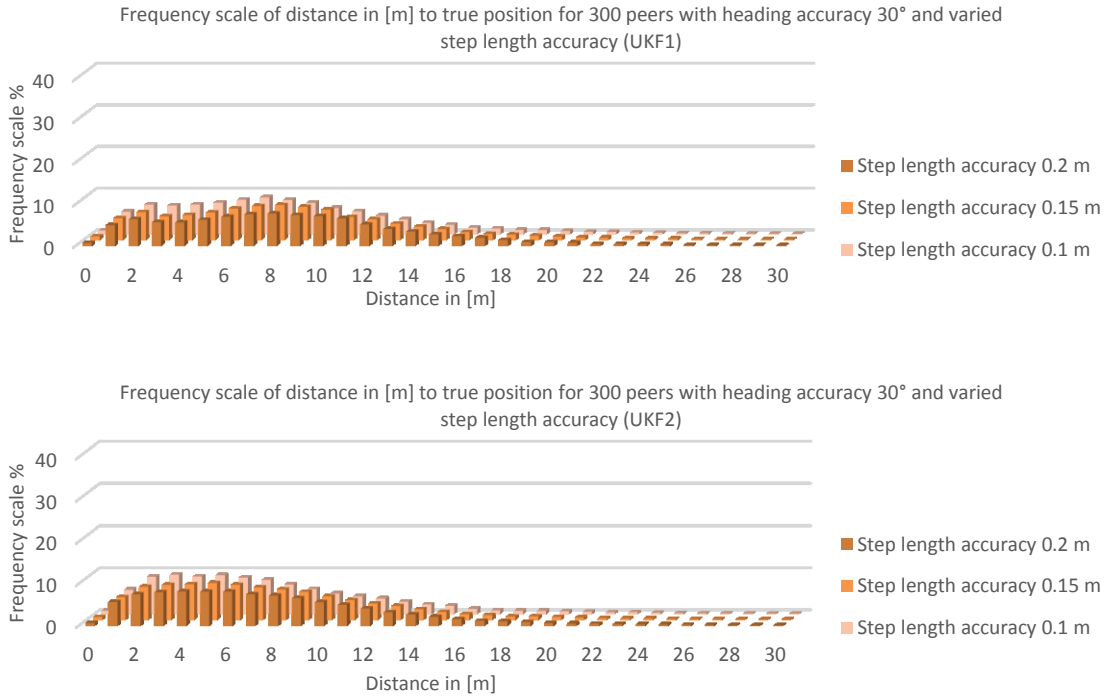


Figure 3-31: Frequency distribution of position error for 300 peers with heading estimation accuracy 30° and varying step length estimation accuracy. Top UKF1, bottom UKF2

3.3.3.2.5 Scenario 4

h_{acc}	$\mu = 10°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1$m $\sigma = 0.05$m

Regarding the figures in Table 3-19 the position error is smaller due to the better accuracy of the heading estimation. Again the results are not as good as those of the simulations using the EKF but as for the EKF it is obvious that the position error is more influenced by inaccurate heading estimation than by inaccurate step length estimation.

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	4.85m	3.83m	4.46m
	Without P2P filtering	4.86m	4.98m	5.14m
UKF2	With P2P filtering	3.89m	3.68m	4.46m
	Without P2P filtering	4.21m	4.31m	5.14m

Table 3-19: Average distance in [m] to the true position with 10° heading estimation accuracy and step length estimation accuracy 0.1m

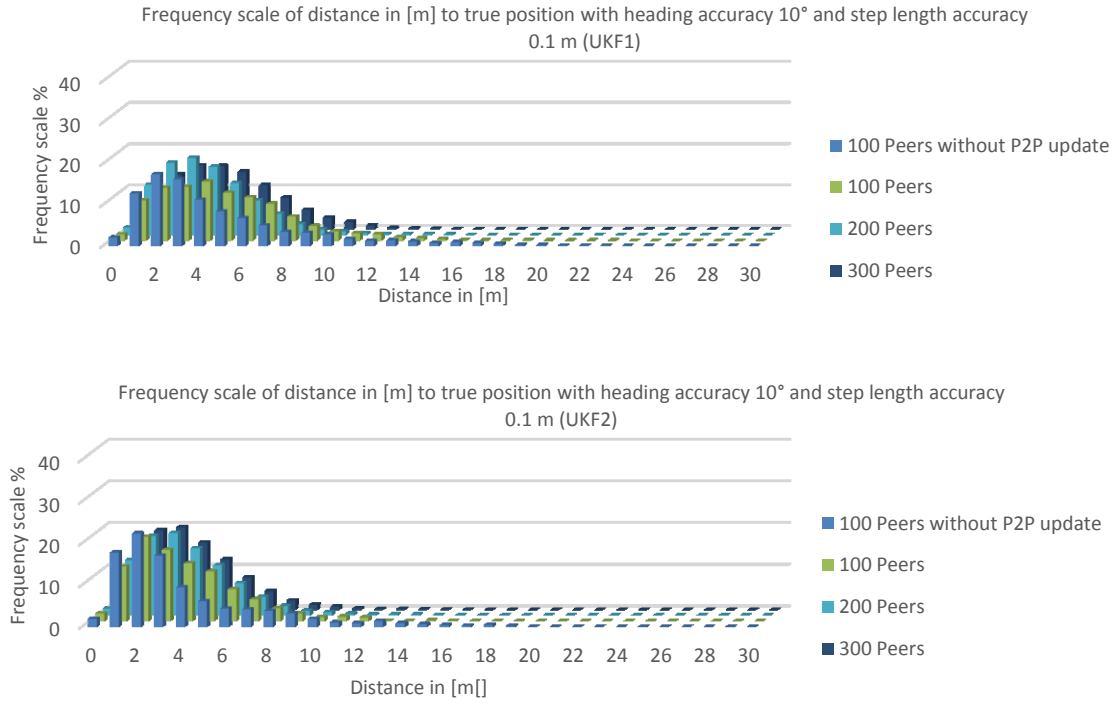


Figure 3-32: Frequency distribution of the position error in meter with 10° heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

3.3.3.2.6 Scenario 5

Refer to scenario 1 in 3.3.3.2.1. The parameters are the same as for this scenario.

3.3.3.2.7 Scenario 6

h_{acc}	$\mu = 45°$ $\sigma = 10°$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$\mu = 0.1\text{m}$ $\sigma = 0.05\text{m}$

The position error is very large in this scenario due to the erroneous heading estimation. It seems that this has an even stronger input on the UKF than on the EKF.

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	11.65m	10.81m	10.65m
	Without P2P filtering	14.62m	15.55m	15.81m
UKF2	With P2P filtering	10.09m	9.91m	11.18m
	Without P2P filtering	12.82m	13.74m	13.76m

Table 3-20: Average distance in [m] to the true position with 45° heading estimation accuracy and step length estimation accuracy 0.1m

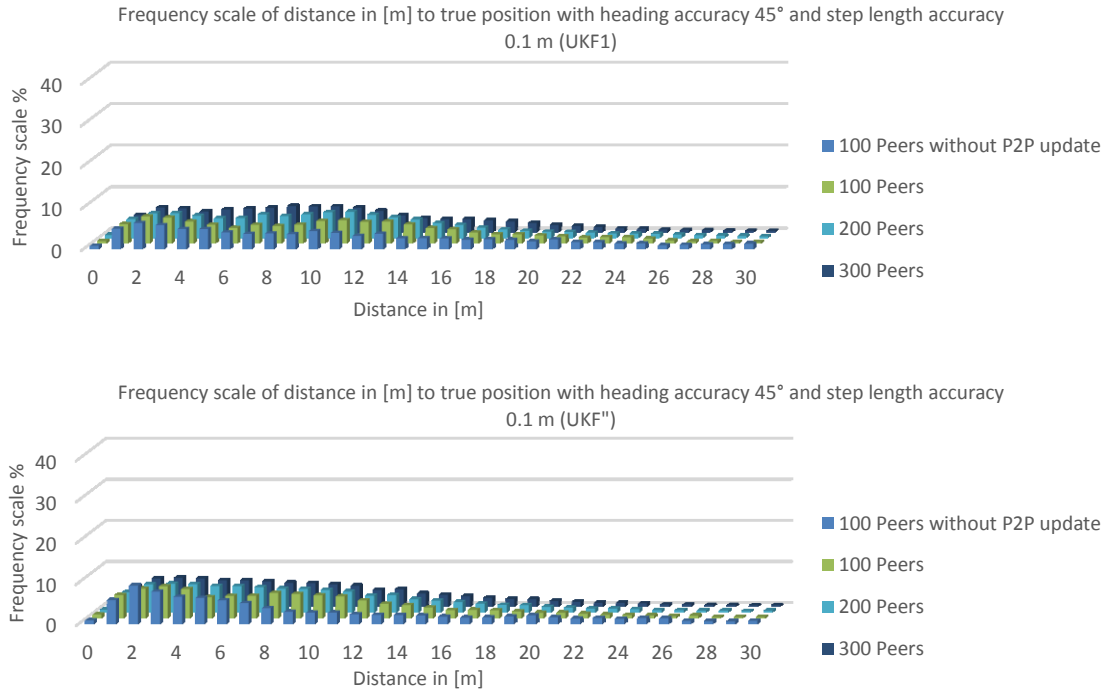


Figure 3-33: Frequency distribution of the position error in meter with 45° heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

3.3.3.2.8 Summary of Scenario 4, Scenario 5, and Scenario 6

Again it becomes obvious that errors in the heading estimation influence the overall position accuracy more than errors of the step length detection. The more degrees of standard deviation the worse the average position error gets. This was also observable for the EKF in 3.3.3.1.8. Again here the UKF2 provides slightly better results than the UKF1 but is still worse than the EKF.

			100 peers	200 peers	300 peers
Scenario 4 (heading est. acc. 10°, step length est. acc. 0.1m)	UKF1	With P2P filtering	4.85m	3.83m	4.46m
		Without P2P filtering	4.86m	4.98m	5.14m
	UKF2	With P2P filtering	3.89m	3.68m	4.46m
		Without P2P filtering	4.21m	4.31m	5.14m
Scenario 5 (heading est. acc. 30°, step length est. acc. 0.1m)	UKF1	With P2P filtering	8.56m	8.05m	8.05m
		Without P2P filtering	10.16m	10.60m	10.74m

	UKF2	With P2P filtering	7.27m	7.07m	7.43m
		Without P2P filtering	9.15m	9.53m	9.64m
Scenario 6 (heading est. acc. 45°, step length est. acc. 0.1m)	UKF1	With P2P filtering	11.65m	10.81m	10.65m
		Without P2P filtering	14.62m	15.55m	15.81m
	UKF2	With P2P filtering	10.09m	9.91m	11.18m
		Without P2P filtering	12.82m	13.74m	13.76m

Table 3-21: Summary of scenarios 4 to 6 for the UKF1 and the UKF2

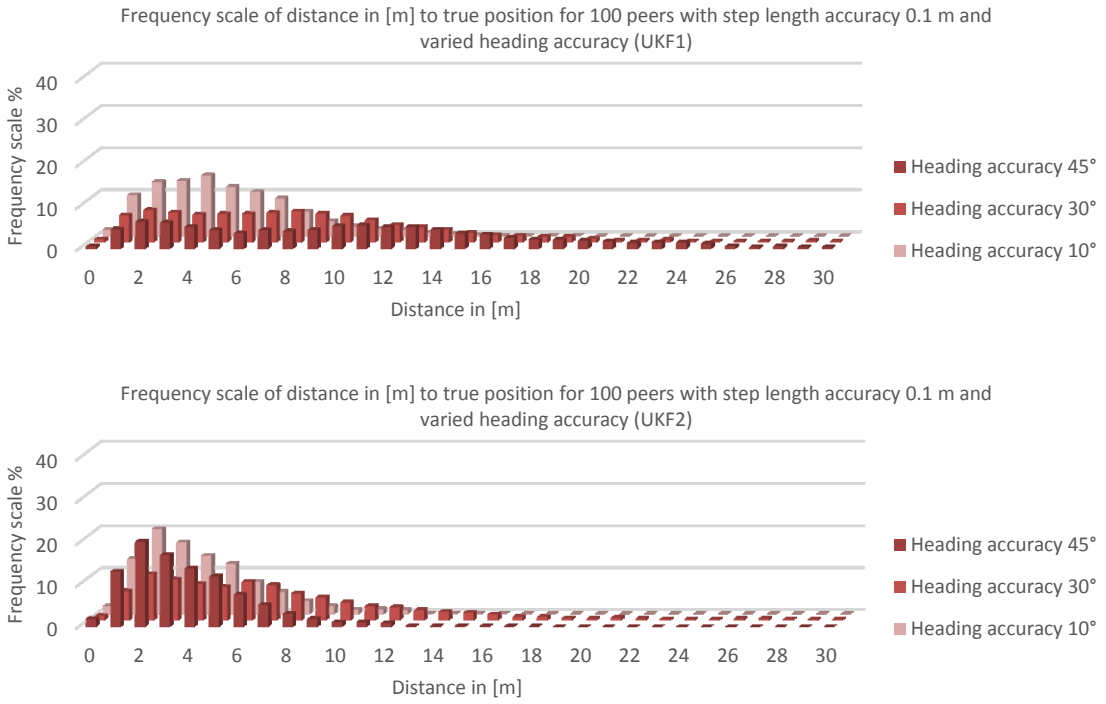


Figure 3-34: Frequency distribution of position error for 100 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m. Top UKF1, bottom UKF2

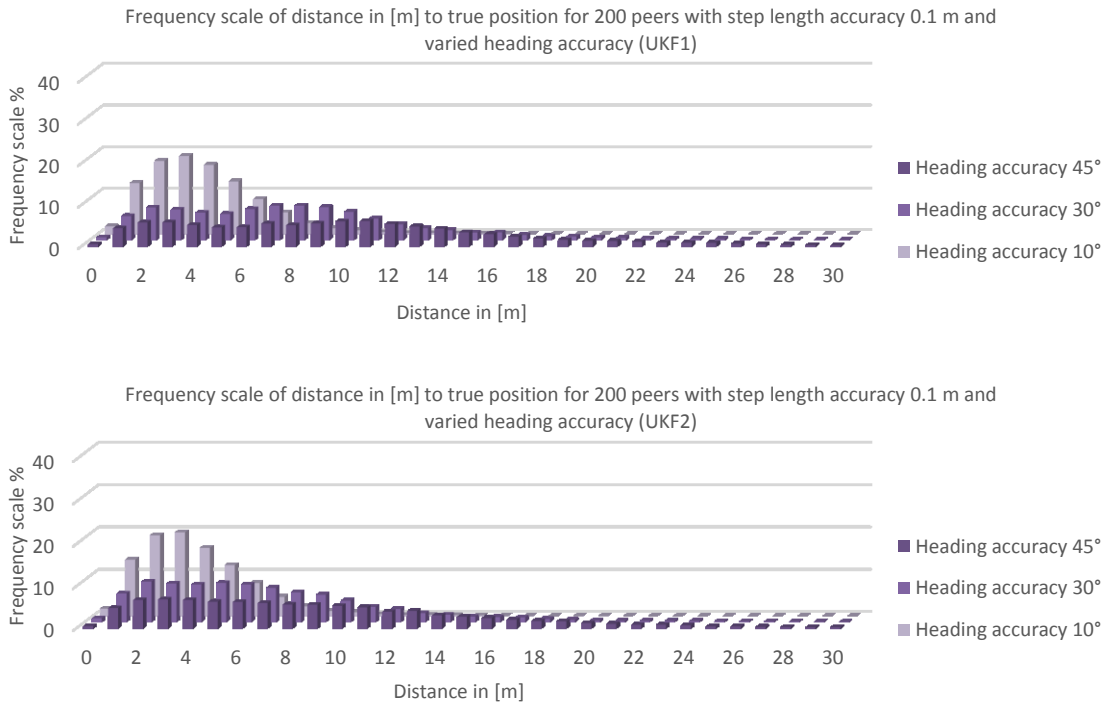


Figure 3-35: Frequency distribution of position error for 200 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m. Top UKF1, bottom UKF2

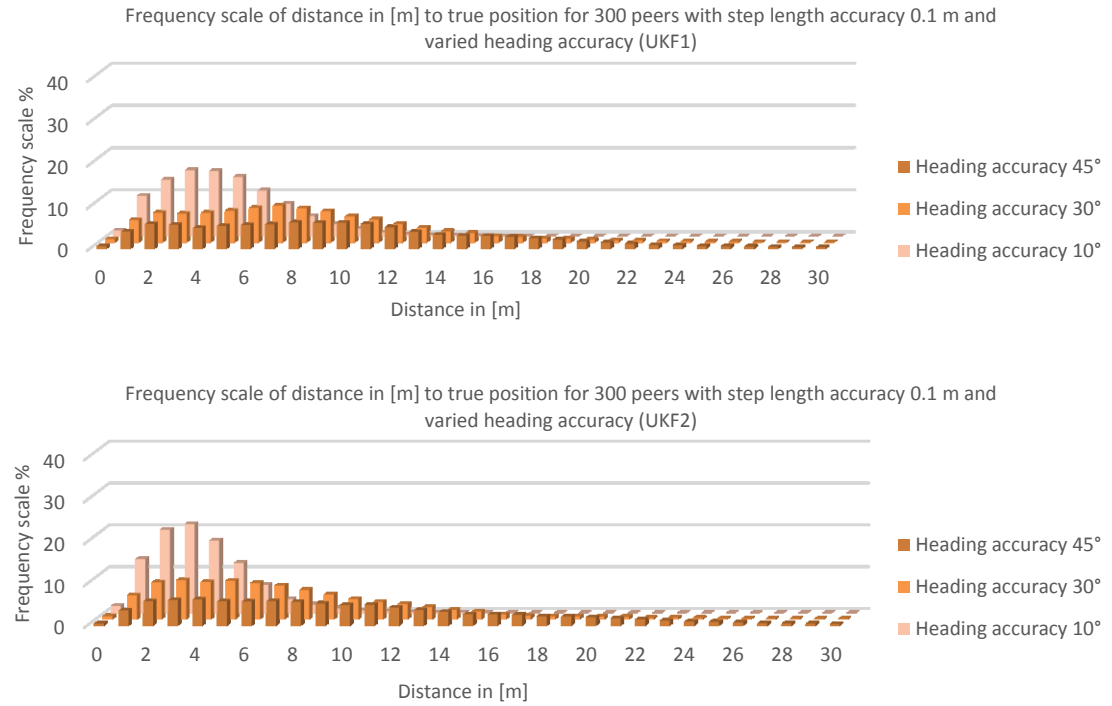


Figure 3-36: Frequency distribution of position error for 300 peers with varying heading estimation accuracy and step length estimation accuracy 0.1m. Top UKF1, bottom UKF2

3.3.3.2.9 Scenario 7

h_{acc}	$[10°, 45°]$
d_{peer}	5m
d_{ref}	1m
l_{acc}	$[0.1\text{m}, 0.2\text{m}]$

Using an equally distributed error for step length estimation and heading estimation the UKF shows the same behavior as the EKF only providing generally larger position errors. But as for the EKF the position of peers with better sensor measurements is not negatively influenced by the measurements of peers with less accurate sensors. The overall result is very similar to scenarios with normal distribution around a common mean.

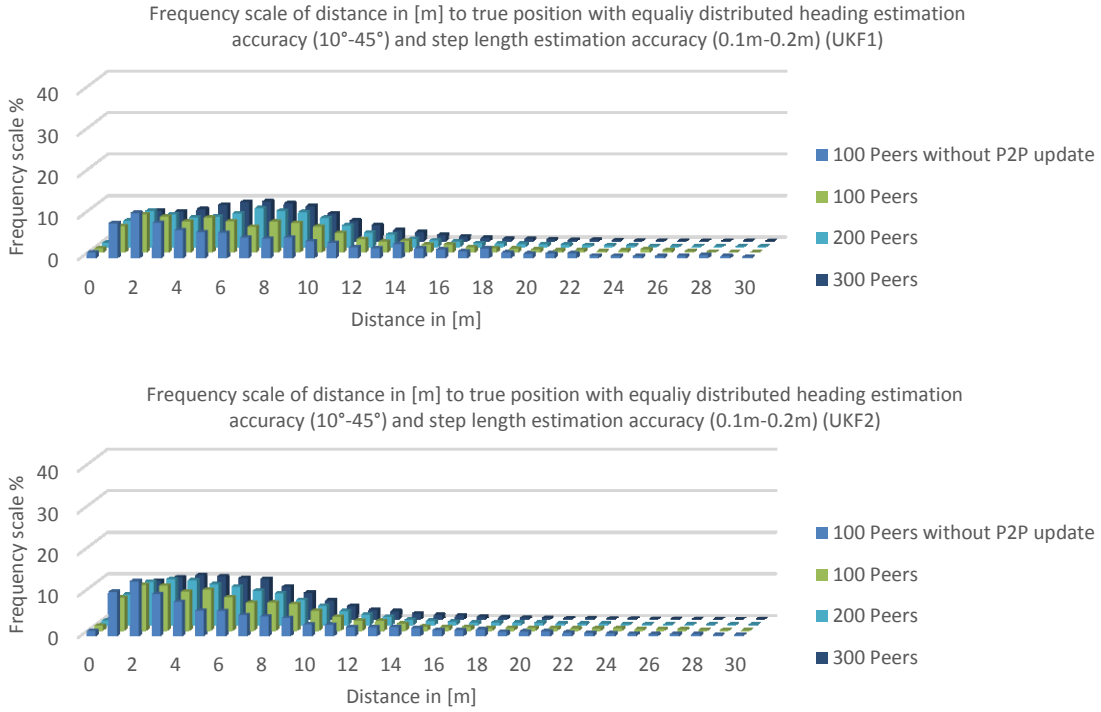


Figure 3-37: Frequency distribution of the position error in meter with equally distributed heading estimation accuracy (10° to 45°) and step length estimation accuracy (0.1m to 0.2m) for 100, 200 and 300 peers. Top UKF1, bottom UKF2

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	7.84m	7.48m	7.34m
	Without P2P filtering	9.49m	9.63m	9.91m
UKF2	With P2P filtering	6.81m	6.39m	6.48m
	Without P2P filtering	8.45m	8.54m	8.78m

Table 3-22: Average distance in [m] to the true position with equally distributed heading estimation accuracy (10° to 45°) and step length estimation accuracy (0.1m to 0.2m)

3.3.3.2.10 Scenario 8

h_{acc}	Start	$\mu = 5°$
		$\sigma = 2°$
	Grow rate	$\mu = 0.1°$
		$\sigma = 0{,}05°$
d_{peer}	5m	
d_{ref}	1m	
l_{acc}	$\mu = 0.1$m	
	$\sigma = 0.05$m	

Again the behavior of the simulations processed with the UKF variations is similar to the one processed with the EKF except that position error generally is larger. But the results of this scenario are much worse than for all other scenarios. Due to the continuous deterioration of the heading estimation the measurement accuracy is so low at the end of the simulation that useful position estimations cannot be processed at least when using the UKF.

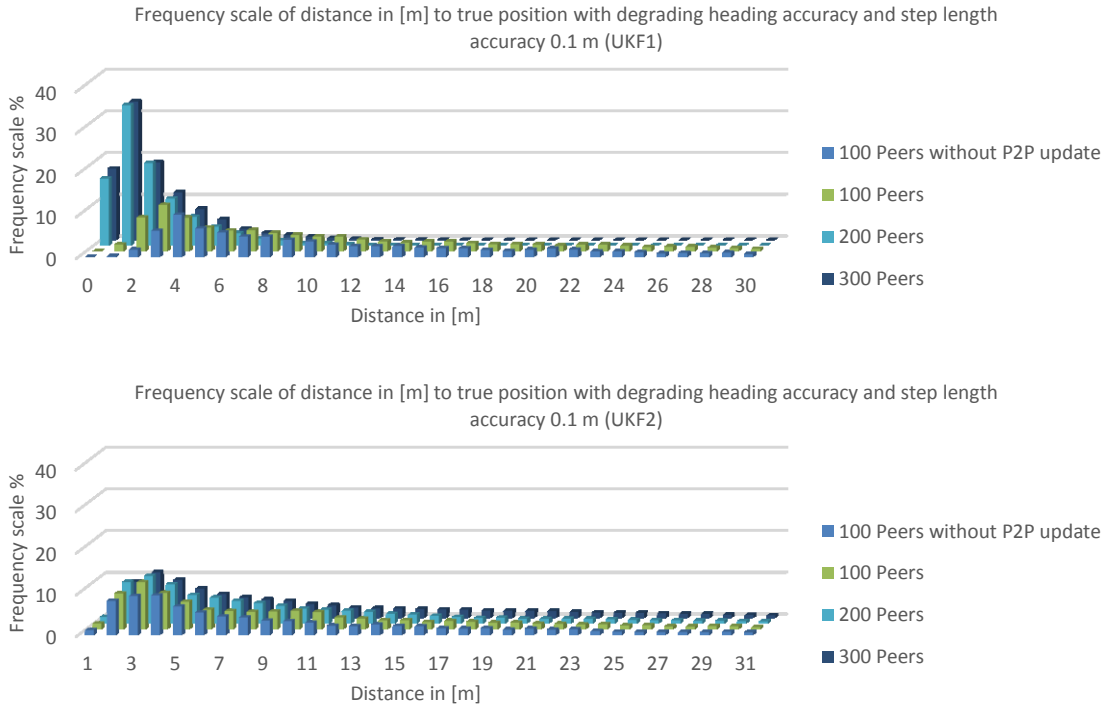


Figure 3-38: Frequency distribution of the position error in meter with growing heading estimation accuracy and step length estimation accuracy 0.1m for 100, 200 and 300 peers. Top UKF1, bottom UKF2

		100 peers	200 peers	300 peers
UKF1	With P2P filtering	13.09m	10.75m	10.86m
	Without P2P filtering	16.67m	16.25m	16.36m
UKF2	With P2P filtering	11.46m	9.86m	11.09m
	Without P2P filtering	13.73m	14.06m	14.13m

Table 3-23: Average distance in [m] to the true position with growing heading estimation accuracy and step length estimation accuracy 0.1m

3.3.3.2.11 Scenario 9

Since the results of the simulation of the EKF already showed that a larger distance to the filtering peer does not influence the overall position accuracy in a positive way it was not processed for the UKF.

3.3.3.3 Comparison of results of EKF, UKF1 and UKF2

Comparing the three filtering types with each other it becomes obvious that the EKF provides the best results also when the P2P filtering is not activated. It is therefore obvious that for the implementation of the prototype the Extended Kalman Filter should be used. One draw with this filter might be the complex computation of the Jacobian matrices. In the simulations this process did not have a big effect on the overall performance since Matlab comprises several functions which enable the computation of the Jacobian matrices and the manipulations of matrices but in object-oriented programming languages such computations are not intended and a workaround has to be found.

		100 peers	200 peers	300 peers
Scenario 1	UKF1	8.56m	8.05m	8.05m
	UKF2	7.27m	7.07m	7.43m
	EKF	2.85m	2.52m	2.33m
Scenario 2	UKF1	8.51m	8.11m	8.49m
	UKF2	7.25m	7.21m	7.77m
	EKF	2.98m	2.48m	2.37m
Scenario 3	UKF1	8.72m	7.56m	8.83m
	UKF2	7.60m	7.25m	8.01m
	EKF	2.99m	2.58m	2.21m
Scenario 4	UKF1	4.85m	3.83m	4.46m
	UKF2	3.89m	3.68m	4.46m
	EKF	2.83m	2.29m	2.06m
Scenario 6	UKF1	11.65m	10.81m	10.65m
	UKF2	10.09m	9.91m	11.18m
	EKF	2.96m	2.55m	2.59m
Scenario 7	UKF1	7.84m	7.48m	7.34m
	UKF2	6.81m	6.39m	6.48m
	EKF	2.98m	2.45m	2.24m
Scenario 8	UKF1	7.84m	7.48m	7.34m
	UKF2	6.81m	6.39m	6.48m
	EKF	3.40m	2.81m	2.67m

Table 3-24: Comparison of a three filter types for scenarios 1 to 8. Scenario 5 is not listed since the results are the same as for scenario 1.

3.4 Implementation of the prototype

The implementation of an executable prototype application was realized as application for the Android operating system. The following sub-chapters will give an overview of this implementation sectioned into three main areas: implementation of the dead reckoning algorithm, the filtering algorithm and the realization of the peer-to-peer communication. Before diving into the details of the prototype an overview of the Android application programming interface (API) and the integrated development environment (IDE) is given.

3.4.1 Android API and Integrated Development Environment (IDE)

The Android Inc. was founded in 2003 amongst others by Andy Rubin who before worked for Apple Inc. Only little was known about the true work of this corporation but due to the background of most of the founders it was assumed that their products aimed at the mobile computing market. When Android Inc. was acquired by Google in 2005 it became obvious that Google wanted to enter the mobile devices market with the help of Android. A mobile platform was developed based on the Linux kernel. At the end of 2007 Google and several other companies (amongst others Broadcomm, HTC, Intel, Qualcom and Samsung Electronics) published as a consortium under the name Open Handset Alliance a preliminary version of their first product: the Android system development kit (SDK). Android is a mobile device platform and maintained and developed within the Android Open Source Project (AOSP) led by Google. Compared to the iOS of Apple and former operating systems for mobile phones as described in 2.5.2.2.2 Android is open source and its use is not limited to the smartphones of certain manufacturers. For more detailed information on Google and Android Inc. see [44], [84] and [85]. In the table below the history of the different Android versions is listed.

Date	Name and number	Comment
Nov. 2007	Android beta	Released together with the SDK
Sep. 2008	Android 1.0	First commercial version
Feb. 2009	Android 1.1	
Apr. 2009	Cupcake 1.5	Based on Linux 2.6.27 kernel
Sep. 2009	Donut 1.6	Based on Linux 2.6.29 kernel
Oct. 2009	Éclair 2.0	Adding several new functions compared to former version, amongst others the use of Bluetooth 2.1 and support for more screen sizes and resolutions
Jan. 2010	Éclair 2.1	Amendments to the API and bug fixes
May 2010 – Nov 2011	Froyo 2.2.x	Release of a new SDK and based on Linux kernel 2.6.32
Dec. 2010 – Sep. 2011	Gingerbread 2.3.x	Updating on Linux kernel version 2.6.35
Feb. 2011 – Feb. 2012	Honeycomb 3.x	An Android version for tablets only. Published together with a new version of the SDK
Oct. 2011 – Mar. 2012	Ice Cream Sandwich 4.0.x	Based on Linux kernel 3.0.1 and should be compatible with any Android 2.3.x device. Can be used on mobile phones as well as on tablets
June 2012 – Nov. 2012	Jelly Bean 4.1.x and 4.2.x	Adding of modified and new functions for the handling

Table 3-25: Overview of the different Android versions [86]

The distribution of the different Android versions on current mobile phones is depicted in Figure 3-39. It is obvious that the most common version still is Android 2.3.3. For this reason the prototype was first implemented for this version. Functionality especially regarding the P2P communication which became

necessary during the evaluation of the prototype required an upgrade of the application for Android 4. This is explicitly mentioned during the description of the evaluation of the prototype in section 3.5.

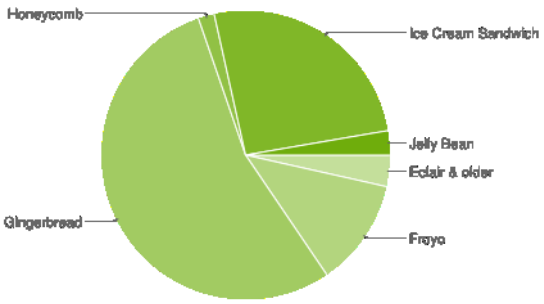


Figure 3-39: Distribution of the different Android version on current mobile phones in June 2012 [87]

3.4.1.1 Android system architecture

The Android Inc. provides a website [88] containing all necessary information for developing Android applications including a reference, tutorials, and guides. For detailed information on application development and fundamentals of the architecture of Android applications please refer to this website. The following paragraphs only reference those parts needed for the description of the prototype application.

Originally Android was developed as operating system for mobile devices like smartphones and PDAs (personal digital assistant). But since version 3 (see Table 3-25) the use on tablets is possible as well. According to [89] it is also possible to enable Android on netbooks, car infotainment systems or gaming consoles in the near future. To enable this kind of device independency Android has a modular architecture as depicted in the figure below.

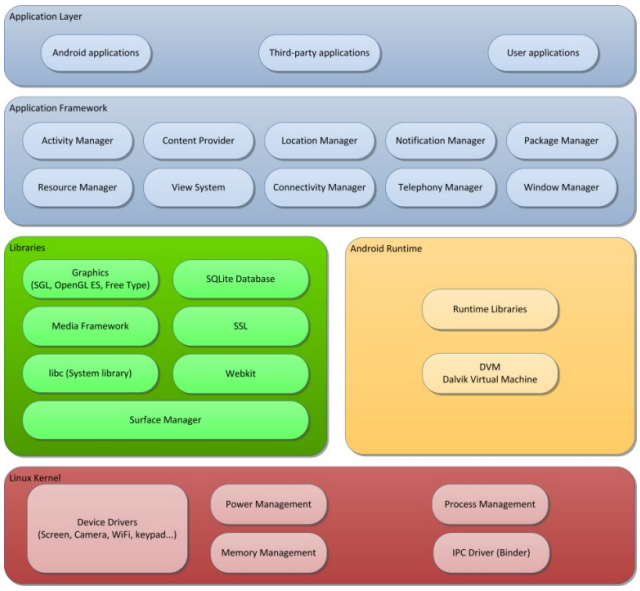


Figure 3-40: Android architecture [89]

As mentioned before the Android system is based on a Linux kernel. In case of Android 2.3.3 it is version 2.6.35. This kernel contains all necessary drivers and additionally optimizations regarding the power management and the memory management compared to the standard Linux kernel for desktops.

Main part of the Android runtime environment is the Dalvik Virtual Machine (DVM). Since Android applications are based on the Java programming language a virtual machine is necessary on which the

applications are processed. The virtual machine processes machine-readable Java byte code which was before compiled from human-readable source code. Due to this procedure Java applications can be executed on each type of platform providing the Java Virtual Machine (JVM). More information on this topic can be found in the appendix 5.2. In contrast to this concept the DVM's platform-independency is limited in favor of its adaptability to poor runtime environments regarding especially a CPU with low clock rate and small memory. These conditions are typical on mobile devices, although the processing power and the memory are continuously growing. Therefore the Java source code is translated to Java byte code and further processed to DVM compatible byte code. This process is called cross-compiling. The DVM processes this byte code during runtime. Additionally to the platform-independency this concept provides an efficient security mechanism which is able to monitor the application during its runtime by avoiding e.g. buffer overflows and the crash of all simultaneously running processes. For more information on the differences between JVM and DVM see [90]. For each application running on Android one operating system process is started within its own DVM.

The core functionality of Android is provided by C/C++ standard libraries summarized in Figure 3-40 as libraries. Those libraries are used by the application framework to provide e.g. browser functionality (LibWebCore), a database system (SQLite) or within the media framework libraries to display and process multimedia content.

The application framework provides classes to access hardware modules from the application. These classes are completely written in Java. For this prototype are especially interesting the Location Manager, the Connectivity Manager and the Sensor Manager which is not depicted within the figure.

On the application layer all applications can be found regardless if they are provided as standard application (Android applications) like phone functionality, contact list, camera etc., as third-party applications purchased from the application store or as applications written by the user. The prototype application is contained in the application layer.

More information on the system architecture of Android can be found in [89].

3.4.1.2 Android application architecture

An Android application is based on different application components dependent on its functionality. Those components enable the re-usability of already written code. This means that if someone writes an application which needs access to the contacts stored on the device it is not necessary to re-write this access in source code instead the pre-installed standard application granting access to the contacts can be used. To enable re-usability of code an application is allowed to use four pre-implemented types of application components listed in Table 3-26:

Name	Functionality
Activity	Activities manage the visible parts of an application. They react on user interaction made on the display. An application which provides user interaction needs at least one Activity but is also able to provide more than one for different views within one application. One activity can be started by another one. Activities act as entry points for the processing of an application.
Services	This component runs in the background and does not necessarily need a graphical user interface. One example for a service is to play music while the user is able to perform other tasks (writing emails etc.) on the device.
Content Provider	The content provider enables the management of data shared by several applications. Based on access rights it provides data access for one or more applications.

Broadcast Receiver	An application which is linked to a certain Broadcast Receiver is able to listen to system messages containing information about changes of the system state and react accordingly

Table 3-26: Components of an Android application

An Android application always contains additionally to the source code resources like media files and a manifest file written in XML. Within the manifest file it is defined which activities belong to this application and which is the starting activity. Permissions for the access of memory, GPS receiver and other hardware are set here. If additional libraries are used within the source code those are linked in the manifest file. More information on fundamentals of Android applications can be found in [89] and [91].

3.4.1.3 Activity lifecycle

The most important component of this prototype application is the activity. Although also the other components are used it is essential to understand the activity's lifecycle to comprehend the architecture and several design decisions of the prototype. A flow diagram is depicted in Figure 3-41.

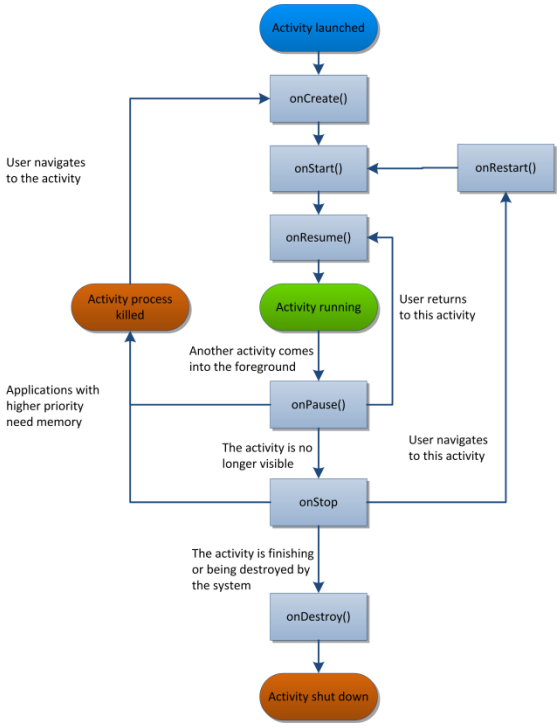


Figure 3-41: The lifecycle of an Android activity [88]

The activity is alive from the call of *onCreate()* until the call of *onDestroy()*. In *onCreate()* the global state of the activity should be set which means for example the definition of the layout or the starting of threads. Within *onDestroy()* these resources have to be freed. When starting an activity at the mobile device it is defined by the manifest file which activity starts and the *onCreate()* method is called by the system like the *main()*-method in a standard Java application (5.2).

The visible lifetime of an activity is between the calls of *onStart()* and on *onStop()*. Also *onStart()* is automatically called by the system as are all other functions depicted in Figure 3-41. The user is able to interact with this activity until *onStop()* is called. Within this time all resources of this activity are maintained.

Between the calls of *onResume()* and *onPause()* the activity is in foreground state. This means that the activity is in front of all other activities on screen and has user input focus. An application normally

moves back and forth between *onPause()* and *onResume()*. Each time a dialog pops up or the device goes to standby mode the activity enters *onPause()* and returns to *onResume()* when the device is turned on again or the dialog is ended.

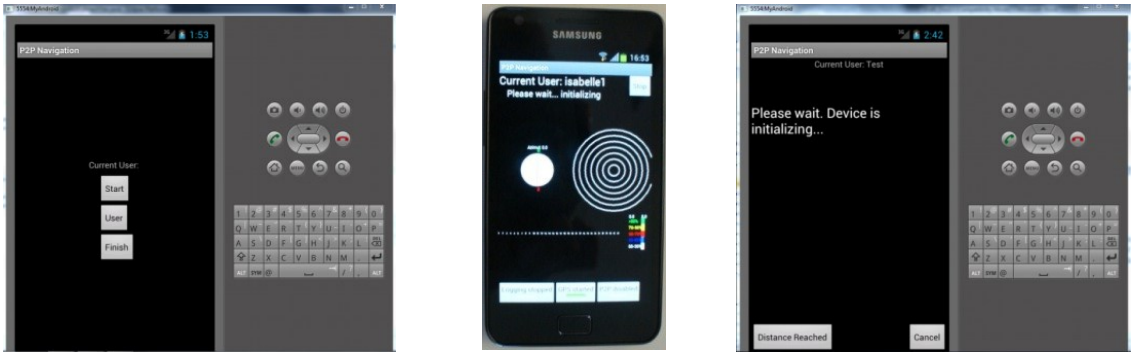
3.4.1.4 IDE

Since Android applications are implemented in Java each common editor can be used to write source code since it is translated in byte code anyway. This also means that all links to the Android library and other libraries as well as the linking of resource files has to be done by hand. It is therefore easier to use an IDE which does most of this work for the developer. The IDE which is recommended and supported by the Android Inc. is as well open source and called Eclipse. The Android API can easily be integrated into the Eclipse IDE to enhance its functionality from pure Java to Android application development. When starting the development of a new Android project in Eclipse the Android libraries are automatically linked, the first activity is pre-implemented as stub and a manifest file is created. Eclipse also detects errors in the manifest file and reports them to the developer. Android Inc. provides so called emulated devices which enable the debugging of applications directly on the PC and avoid the transfer to the mobile phone. This makes the development as well as testing and debugging a lot easier therefore this IDE was chosen for the prototype development.

3.4.2 General setup of the P2PNavigation application

3.4.2.1 Graphical user interface (GUI)

The prototype application's name is P2PNavigation and consists of three activities which are serially processed. The starting activity *P2PNavigationStartActivity* is depicted in Figure 3-42 (1) executed on an emulator on the PC.



(1) (2) (3)

Figure 3-42: The different views of the P2PNavigation application: (1) Start screen running on an Android emulator, (2) The dead reckoning view running on an Android phone and (3) the activity for determining the *k*-factor of a user.

The dead reckoning algorithm is implemented as described in 2.5.3. This means that for each user a *k*-factor has to be determined. If the user is already initialized the *k*-factor has already been stored together with the user name in the memory of the device and is retrieved from the memory when clicking the *User* button on the start screen. A list of names is presented, the user chooses her name and the *k*-factor for this user is read from memory. The device can be immediately used for dead reckoning and the activity depicted in Figure 3-42 (2) is opened. GPS positioning is started automatically. Since a start position for inertial navigation is necessary the user has to start this application in environments where satellite signals are available and a position can be obtained. As soon as a position is estimated the user can use the device for GPS positioning as well as for dead reckoning positioning. The functionality of this activity is explained in more detail in the next paragraphs.

If the application is started for the first time and no user has been initialized up to now a click on the button *Start* or *User* opens a new dialog asking for the name of the new user and parameters to determine the *k*-factor (compare 2.5.3, a more detailed description of the estimation of the *k*-factor follows in 3.4.3).

To determine the *k*-factor of a user a new activity is opened as depicted in Figure 3-42 (3). As soon as here the button *Distance Reached* is clicked the *k*-factor is stored for this user.

The application flow is summarized in the flow diagram below.

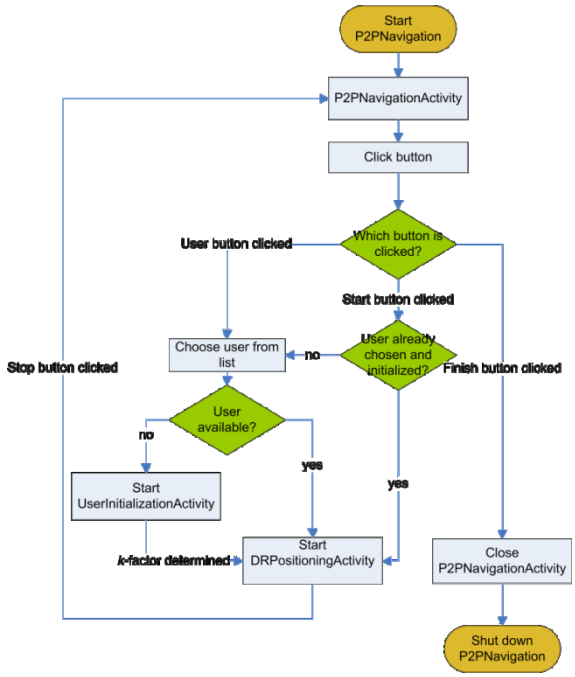


Figure 3-43: Flow diagram of the prototype application P2PNavigation

3.4.2.2 Class diagrams and relations

All classes of the prototype application are divided into three different packages which are contained within one main package. The names of the three sub-packages are navigation, filtering and communication.

Full package name	Description	Classes
de.unibw.navigation	Contains all classes responsible for position estimation (GPS and dead reckoning), user interaction and logging	CompassView, DRPositioningActivity, FileIOTask, OpenCounter, P2PNavigationStartActivity, SatelliteView, SensorView, Step, StepAnalyzer, StepData, UserFileIO, UserInitializationActivity,
de.unibw.filtering	Contains all classes necessary for the Kalman filter algorithm	Complex, DeadReckoningFunction, DiscreteKalmanFilter, ExtendedKalmanFilter, FunctionHandler
de.unibw.communication	Contains all classes managing communication	AcceptBluetooth, ConnectBluetooth, ManageBluetooth

	and data transfer with other devices	

Table 3-27: Summarizes the content of all packages of the prototype application

Figure 3-44 depicts the most important classes of the *navigation* package and their relation to each other.

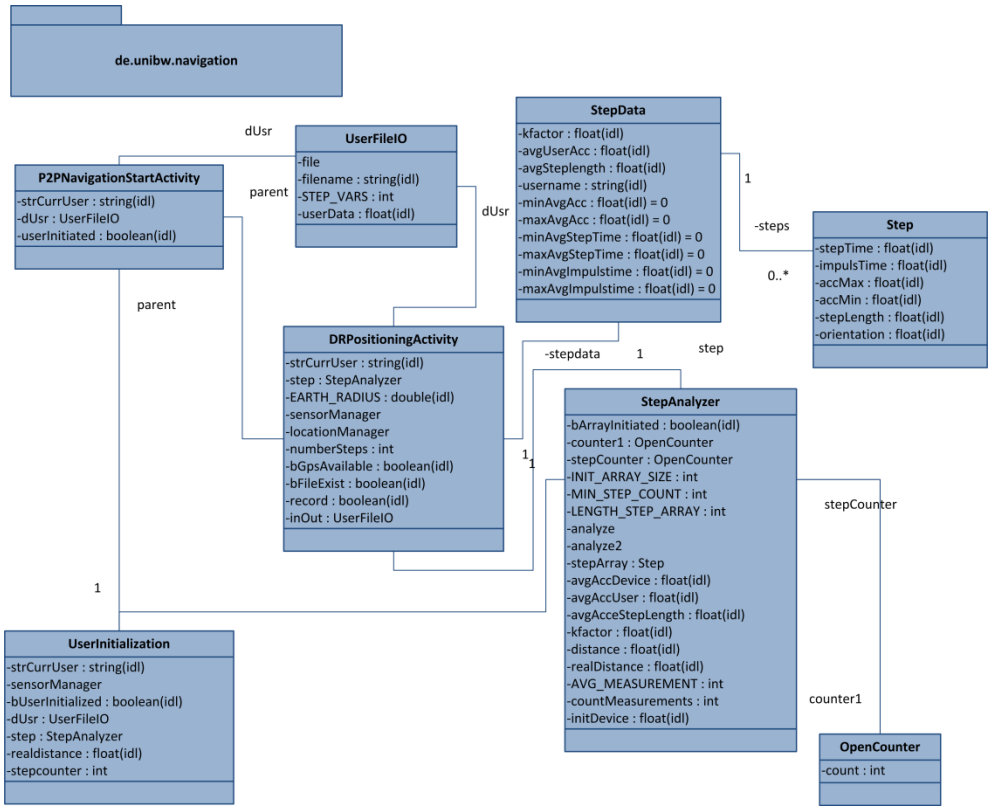


Figure 3-44: Class diagram of the *navigation* package

The class *StepAnalyzer* realizes the detection of steps during the user initialization but also during normal dead reckoning. Therefore both classes –*DRPositioningActivity*, responsible for normal dead reckoning, and *UserInitializationActivity*, responsible to estimate the *k*-factor – are linked with the *StepAnalyzer* class. The characteristic of one step is contained within the structure-like class *Step*. It stores among others information about the step length of each step and its orientation. Several objects of *Step* are kept as array within class *StepData* which additionally contains general information about the user like the *k*-factor or the average step length. Class *UserInitializationActivity* stores *StepData* individually for each user when the *k*-factor has been determined. Each time the user now wants to use the device for dead reckoning the information contained in this class is loaded by the *DRPositioningActivity* using class *UserFileIO*. *UserFileIO* reads the step data from the memory, re-assembles it to an object of class *StepData*, and provides it to class *DRPositioningActivity* for dead reckoning. Some information contained in class *StepData*, like e.g. the average step length, and the time intervals for swing and heel-touch-down phase are only stored for statistical and debugging reasons. The classes *SatelliteView* and *SensorView* listed in Table 3-27 are used to display satellite and sensor information on the screen and the class *FileIOTask* realizes the logging of position data.

Within the package *filtering* classes to enable the Kalman Filter are implemented. Main classes are of course the class *DiscreteKalmanFilter* and its derived class *ExtendedKalmanFilter*. Java and also Android do not provide a library for matrix computation therefore an open source package called JAMA published by the National Institute of Standards and Technology (NIST) is used for this application [92]. The library provides the class *Matrix* and also functionality like Cholesky decomposition, adding,

multiplying and so on. Due to the numerical estimation of Jacobian matrices for the filtering complex numbers are necessary which have been realized in class *Complex*. The classes for Kalman Filtering and for complex numbers have been implemented in the scope of a bachelor thesis [93] according to the algorithm described in 3.2.2.2 and 3.3.2.1. To keep the design as generic as possible the implementation of the prediction function *f* is sourced out to class *DeadReckoningFunction* which inherits from abstract class *FunctionHandler*. In this way different functions can be tested and used for filtering. The *DiscreteKalmanFilter* class is linked only to class *FunctionHandler* and is therefore able to access different classes of functions inheriting from *FunctionHandler* and implementing operation *myFunction()*.

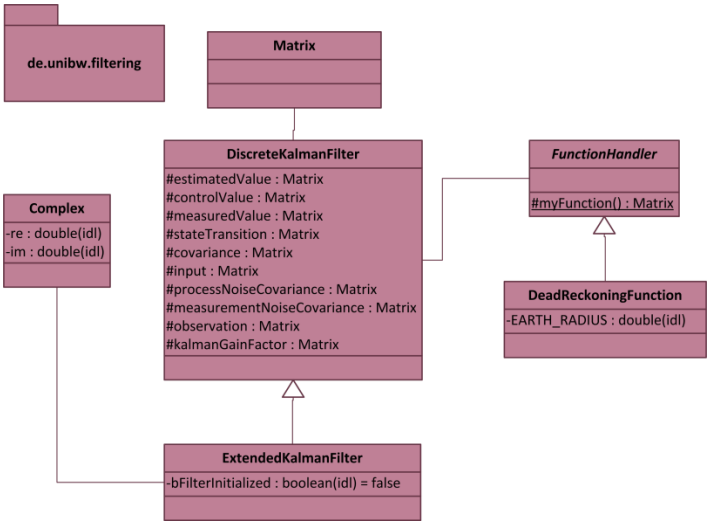


Figure 3-45: Class diagram of package *filtering*

Within the *communication* package depicted in Figure 3-46 the data transfer between peers is managed. Since the mobile device should be able to further compute the user's position the peer-to-peer communication is realized in concurrently running threads. All classes implemented for this package inherit from class *java.lang.Thread* and can therefore be processed in parallel. *AcceptBluetooth* acts here as a server handling incoming requests from other peers while *ConnectBluetooth* acts as client requesting data from other peers. Objects of both classes can be processed simultaneously. The class *ManageBluetooth* handles the actual data transfer and closes the connection when finished. The Android API provides a very rich and simple to use library for Bluetooth communication. Some of the classes of this library are depicted in Figure 3-46 to demonstrate the dependency of the implemented classes. *AcceptBluetooth*, *ManageBluetooth* and *ConnectBluetooth* have been implemented within the scope of a bachelor thesis [93].

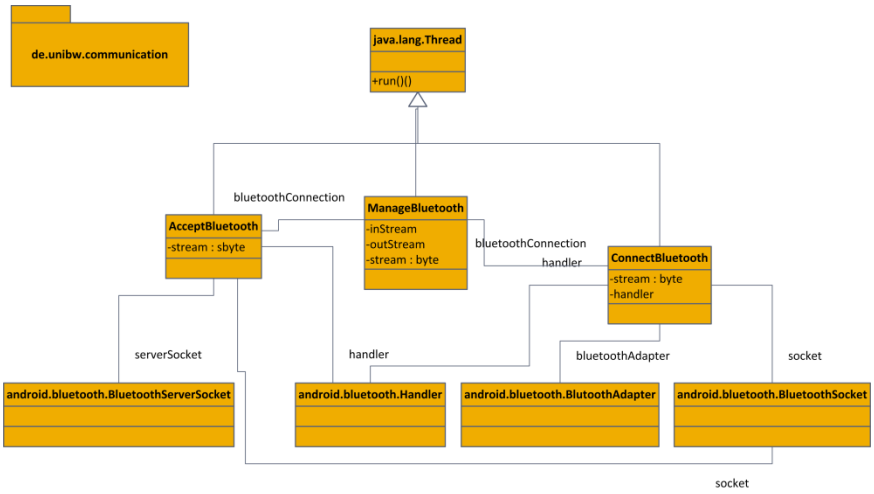


Figure 3-46: Class diagram of package *communication*

The class which connects all packages to each other is *DRPositioningActivity*. It initializes the Kalman filter as well as sockets for Bluetooth connection. How this is realized is demonstrated within the following paragraphs.

3.4.3 Implementation of the dead reckoning algorithm

To use dead reckoning for the computation of a position the *k*-factor has to be determined. As described in 2.5.3 the *k*-factor can be estimated by relating the actual walked distance to the computed distance as depicted in equation (2-13). The computation of the *k*-factor is automatically processed by the application if the user has not been initialized yet. Before starting to walk the average vertical acceleration of the device while the user is standing still and holds the device in the hand has to be computed by taking 100 measurements of the vertical acceleration, summing them up and taking the average of them. This is done each time the device is used for this application regardless if the user was already initialized or not.

After computing the average vertical acceleration the user is asked to enter the distance she wants to walk for estimating the *k*-factor. The distance has to be at least 100 m; then the user starts to walk. A step is detected according to the requirements listed in 2.5.3; during the swing phase the measured vertical acceleration increases to a maximum height and drops then below the average vertical acceleration. When crossing the average vertical acceleration the heel-touch-down phase starts characterized by measuring minimum vertical acceleration smaller than the average vertical acceleration. The heel-touch-down phase stops when the measured acceleration crosses the average vertical acceleration and either a new swing phase is started or the user stands still.

As soon as this pattern is recognized by the application within a defined time interval, it is assumed that the user made a step. If the *k*-factor of the user is not yet defined a *k*-factor of 1 is assumed otherwise the user's true *k*-factor is integrated into the step length estimation. Using the equation in (2-12) the step length is computed and added to the total measured distance. During the initialization of the user the measured distance is much longer than the truly walked distance due to the undefined *k*-factor. For future computations of the position by dead reckoning the once defined *k*-factor is used. It is therefore possible to save different *k*-factors for one user adapted to different paces (strolling, walking, running, etc.).

3.4.3.1 Access to sensor measurements

To estimate the average vertical acceleration and the minimum and maximum vertical acceleration during heel-touch-down phase and swing phase to compute the step length and detect a step it is necessary to access the measurements of the accelerometer. The Android API provides therefore the

pre-implemented interface *SensorEventListener* and class *SensorManager*. The classes *UserInitializationActivity* and *DRPositioningActivity* use *SensorEventListener* as interface which provides two abstract functions: *onAccuracyChanged()* and *onSensorChanged()* which have to be implemented. Both classes also have a member variable of type *SensorManager*. Using the *SensorManager* the calling class is registered as listener for *SensorEvents* of certain sensors. To register for events coming from the accelerometer *DRPositioningActivity* initializes its *SensorManager* and calls:

```
manager = (SensorManager) getSystemService(SENSOR_SERVICE);
```
(3-59)

```
manager.registerListener(this, manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_FASTEST);
```
(3-60)

Each time the accelerometer has new measurements available it calls the function *onSensorChanged()* of the registered class and provides them within an array storing three elements of type *float* representing the measured acceleration in all three axes. In this way all available sensors of the device can be accessed. The time interval between two measurements depends on the second parameter in the function *getDefaultSensor()* in (3-60). Android has no constraints about the minimum time interval between two measurements it is rather dependent on the capability of the device and on its current work load. It is also obvious that the smaller the time interval between two measurements the more CPU power is needed which drains the battery faster. However the Android API provides four constants for the access velocity suitable for different types of applications. For the prototype application the fastest access is chosen to follow the movements of a walker easily.

Originally the accelerometer in mobile devices is used to enhance the user interaction with the device in the form that the screen changes from landscape to portrait view and vice versa when moving it. Therefore it was enough to measure the acceleration directly which means that also the gravitational acceleration is integrated into the measurements depending on the way the user holds the mobile device. This means that if the device is lying on a table vertical acceleration in form of gravitational acceleration is measured. Using a low-pass filter this acceleration can be excluded resulting only in linear acceleration. However, for the detection of steps and their length estimation the direct measurements are completely adequate as can be seen from equation (2-12) since only the difference between the maximum and the minimum acceleration measured within one step are necessary to estimate the step length. Also there is no need for the average vertical acceleration to exclude the gravitational acceleration as long as the user keeps the device as steady as possible during navigation.

As mentioned before all other sensors can be accessed in the same way as described for the accelerometer. They all return an array containing their measurements. The size of the array depends on the amount of measurements which can be provided by the sensor. Since the acceleration, the rotation and the magnetic field are measured in three orthogonal axes all arrays of these sensors contain three elements representing measurements taken in x-, y- and z-direction. But there are also sensors which provide only one measurement like the temperature or the current pressure. Note that not all the described sensors are already integrated into existing mass market devices – the gyroscope for example is a relatively new sensor – but the Android API already provides functionality to access this data.

The table below gives an overview of the measurements different sensors provide and their units.

Sensor	Amount measurements	Unit
Accelerometer	3	$\frac{\mathrm{m}}{\mathrm{s}^2}$
Magnetic Field Sensor	3	µT (micro-Tesla)
Gyroscope	3	$\frac{\mathrm{rad}}{\mathrm{s}}$

Ambient light	1	lx (Lux)
Pressure	1	hPa (millibar)
Proximity	1	cm
Relative humidity	1	%
Ambient temperature	1	°C

Table 3-28: Summarizes the sensors the Android API provides and their measurements

3.4.3.2 Coordinate frames of Android

As mentioned in 2.5.3 for the prototype application only the measurements of the magnetic field sensor and the accelerometer are used to determine on the one hand the heading of the user and on the other hand to detect steps and estimate their length. The use of gyroscope measurements to estimate the heading is currently under development.

Therefore a computation is necessary to retrieve the heading in degree from the magnetic field sensor and the acceleration. Fortunately the Android API provides several functions to do this computation and in former versions even an own sensor type providing exactly this information. But since the use of this sensor type is deprecated only the functions are used here. Before going into detail of these functions it is essential to comprehend the different coordinate frames of the mobile device as defined by Android.

According to Android the axes of the sensors of an Android-compatible device should be adjusted as depicted in Figure 3-47.

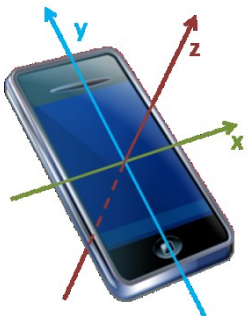


Figure 3-47: Body-frame of an Android-enabled mobile device

Using the function *getOrientation(float [] R, float [] values)* of class *SensorManager* the orientation of the mobile device can be derived, returning its azimuth, pitch, and roll. Two arrays are necessary for this function. One is the rotation matrix *R* as input while the array *values* stores the result in three elements. To compute the correct rotation matrix the call of another function implemented in class *SensorManager* is necessary.

Calling *boolean getRotationMatrix(float [] R, float [] I, float [] gravity, float [] geomagnetic)* stores the results of this function in array *R* (nine elements) and in array *I* (nine elements) using as input the measurements from the magnetometer (*float [] geomagnetic*) and from the accelerometer (*float [] gravity*). The function computes the inclination matrix *I* and the rotation matrix *R* by transforming a vector from the body-frame of the device to the world's coordinate system. The definition of the world's coordinate frame is depicted in the figure below. It is defined as orthonormal basis with the *x*-axis as vector product of *y* and *z*, the *y*-axis as tangent on the ground pointing toward the geographic North Pole, and the *z*-axis pointing toward the sky perpendicular to the ground.

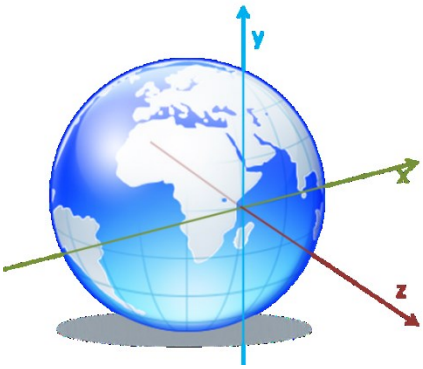


Figure 3-48: The World's coordinate frame as defined by the Android system

Matrix *I* implemented as array of nine elements is a rotation matrix which transforms the geomagnetic vector into the same coordinate space as the gravity vector whose elements are expressed in coordinates of the World's coordinate frame. *I* is therefore a rotation around the *x* axis. Matrix *R* is the identity matrix if the device would be aligned with the world's coordinate system, which means that device's *x*-axis points east, the *y*-axis towards the North Pole and the device is facing the sky (compare Figure 3-47). The function returns true if the input could be successfully used to compute *R* and *I* and false otherwise. In the Android reference for this function the following relations are defined [94]:

$$[0\ 0\ \mathrm{g}] = R * gravity \tag{3-61}$$

$$[0\ \mathrm{m}\ 0] = I * R * geomagnetic \tag{3-62}$$

With

g magnitude of gravity

m magnitude of geomagnetic field

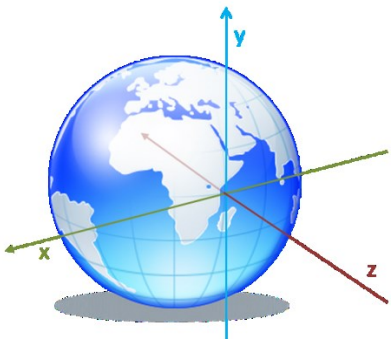


Figure 3-49: The reference coordinate system which is used to express the orientation of an Android enabled device

As described above the rotation matrix *R* as computed by *getRotationMatrix()* is then applied to the function *getOrientation()*. The reference coordinate system is a different one than the World coordinate system defined for the rotation matrix. Here the *x*-axis points West and the *y*-axis to the center of the Earth while the *y*-axis still points to the geographic North Pole. The function returns the azimuth (rotation around *z*-axis), the pitch (rotation around *x*-axis) and the roll (rotation around *y*-axis) as positive radians in the counter-clock direction within an array.

To compute the position of a user by dead reckoning only the azimuth of the device is of interest. It is used to determine the new position together with step detection and step length estimation. The following section explains how a new position is computed.

3.4.3.3 Position computation

To enable the prototype application to read GPS data from the receiver another listener class implemented in the Android API is necessary. The class *DRPositioningActivity* implements as interface the *LocationListener* and the *NmeaListener* of the Android API. Additionally the application has to register itself for receiving notifications from the *LocationManager* (Android API) when the location is changed. The *LocationListener* provides four functions; one for handling location updates and three for handling status changes concerning the GPS receiver. The function *onLocationChanged(Location loc)* is automatically called when a new position (*loc*) is available. Depending on the specifications when registering the application at the *LocationManager* an update happens within a certain time or when a certain distance is reached between the new and the old position. For the prototype application a time interval of 1 s was chosen. The GPS position is contained as ellipsoidal coordinates in the *Location* object passed by the *onLocationChanged*-function. This object additionally contains altitude, bearing (if available), speed (if available), and accuracy (if available).

It is relatively easy to obtain position information as can be seen from the description above. What causes more trouble is to detect when the receiver is not able anymore to provide a valid position estimate by GPS signals. The *onLocationChanged* function is called within the defined time interval even when no new position is available and may still provide some sort of location. Also the other three functions provided by the *LocationListener* do not offer valuable information if GPS is still available. The only way to be sure about the validity of an incoming GPS position is to listen to NMEA strings which can also be obtained from the *LocationManager*. It is therefore necessary that the *DRPositioningActivity* implements additionally the interface *NmeaListener* and registers itself for incoming NMEA sentences. There are several types of NMEA sentences but for this application are only those two of interest which contain a position fix. For more information on NMEA sentences please refer to [95]. *NmeaListener* provides the abstract function *onNmeaReceived(long timestamp, String nmeaSentence)* which is implemented in *DRPositioningActivity*. Each time a NMEA sentence is received it is parsed and checked on a valid position fix. The value in *timestamp* is compared to the value of the last *timestamp* where a valid fix was received; if no valid fix is received within a certain time interval it is assumed that GPS is currently unavailable. Since the minimum update rate between two position fixes received in *onLocationChanged* is 1 s the maximum rate between two NMEA sentences containing a valid fix is 1.5 s. As soon as valid NMEA sentences are again received it is assumed that GPS is available.

As mentioned already in 2.5.1 and 2.5.3 a more or less precise starting position is necessary to enable inertial navigation and dead reckoning. The better the starting position is the better the position estimation by dead reckoning gets. This means for the prototype application that the device has to estimate a position via GPS before it is able to start dead reckoning when the user enters an area with weak or no GPS coverage. Of course also the GPS position computed by a conventional L1 receiver implemented within a mobile phone is not centimeter-accurate but it is the best what is available. These kinds of receivers are able to provide an accuracy of below 10 m depending on the signal availability and assistance data as described in 2.4. Based on the last available position estimated by GPS the device computes a new position as soon as a step is detected using the algorithm for step length estimation and heading estimation. As already mentioned in 3.2.1 the GPS receiver normally provides the current position in latitude and longitude therefore the equations described by (3-3) and (3-4) are used to compute a new position by dead reckoning.

As mentioned in 3.4.3.2 the heading is computed using magnetometer and accelerometer measurements and using them in *getRotationMatrix()* and *getOrientation()*. The step length is derived from the

accelerometer measurements and the class *StepAnalyzer*. The Earth radius is a constant which is defined with 6371.0008 km (WGS84).

The accuracy of the dead reckoning algorithm has been tested to set the parameters of the Kalman Filter accordingly. The results of these tests are reviewed in 3.5.1.

3.4.4 Implementation of the Kalman filter

Unlike Matlab Java does not provide a standard library for matrix computation. There is a package available for several mathematical functions but also this does not contain a matrix class and functions. To realize the Kalman filter the handling of matrices is mandatory therefore an open-source package from the National Institute of Standards and Technology (NIST) as mentioned already in 3.4.2.2 is used for working with matrices [92]. *JAMA* (Java Matrix package) contains the classes *Matrix*, *CholeskyDecomposition*, *EigenvalueDecomposition* and several other useful classes for matrix manipulation.

The Kalman Filter is implemented according to the algorithm introduced in 3.2.2.2 and 3.3.2.1. Only the propagation function *f* differs as already described in 3.4.3.3 since instead of Cartesian coordinates ellipsoidal coordinates are used. As depicted already in Figure 3-45 the propagation function is implemented as class enabling the test of other possible functions for future development. The measurement functions match the functions introduced in 3.3.2.1 completely. There are two possible functions to process measurements.

One measurement function h_1 is used when only measurement data in form of step length and bearing are available, assuming that the user continues to head in a certain direction with her individual step length. Instable sensor measurements especially regarding the heading can therefore be easily balanced by the filter. Nevertheless a change of direction is also traceable based on sensor measurements.

The second measurement function h_2 comprises additionally the position estimation of another peer residing within the communication range of a Bluetooth ad-hoc network. This means four different measurements are available: step length, heading, latitude and longitude of the other peer. The variance of the position of the other peer is introduced by the current error covariance of the other peer. If the other peer is able to compute her position by GPS signals the variance of the position is read from the current *Location*-object provided by the *LocationManager* (compare 3.4.3.3). All measurements are used directly, assuming that the user does not change her direction nor her step length and that the position of the other peer is used as measurement for the own position. Therefore both measurement functions map their variables onto themselves (see equations (3-51) and (3-52)).

Another issue which has to be addressed is the use of complex numbers to compute the Jacobian matrices *A* and *H* by complex step differentiation as described in the algorithm in (3-53). Java also does not provide functionality for complex numbers. Therefore in [93] a class for complex numbers was implemented which is integrated into the prototype application. Class *Complex* enables the manipulation of complex numbers and is used to derive the Jacobian matrices.

As can be seen from Figure 3-45 the class *ExtendedKalmanFilter* inherits from class *DiscreteKalmanFilter*. This provides the possibility to develop other versions of the Kalman filter like an UKF. Within the *DiscreteKalmanFilter* class all functionality and especially all member variables have already been implemented in a way that all inheriting classes are able to access these variables and functions. In class *ExtendedKalmanFilter* the functionality is completed by computing the output of the propagation function *f* and the measurement function *h*, as well as the determination of the Jacobian matrices.

During dead reckoning the Kalman filter is used for each step the user takes, propagating the new position ahead in time based on the measurements step length and heading and the measurement function

as described above. As soon as another peer is in communication range the filtering process is extended to four measurements including the position of the other peer separated into latitude and longitude. When GPS signals are again available the dead reckoning process is stopped as well as the filtering. This is different than described in the simulations with Matlab (compare 3.3.2.1). Here the GPS position measured at certain places indoors (window and door) is filtered with the position computed by dead reckoning since it is assumed that the GPS position in such an environment cannot be very accurate. However it turned out during the test of the implementation that the position accuracy is better when not combining GPS position with dead reckoning position. Dead reckoning is either turned on when no GPS at all is present or turned off as soon as enough signals become available again. Both positioning functions are never used simultaneously in the prototype application.

The *ExtendedKalmanFilter* class is called by *DRPositioningActivity* for each detected step. All necessary data like the state vector consisting of step length, heading and current position are handed to the functions *update()* and *estimate()* of the *ExtendedKalmanFilter*. If the position of another peer is available it is passed to the Kalman filter as well as additional measurement. Within the *estimate()* function the state vector is propagated ahead in time while in the *update()* function the measurements are integrated and the Kalman gain and a new state vector are computed. The position of another peer is retrieved by peer-to-peer communication via a Bluetooth network.

3.4.5 Implementation of the P2P communication

The contact to other peers and their position solution for filtering is managed by the classes depicted in Figure 3-46. As already mentioned all of these classes are realized as threads to ensure continuous processing of the application also during sending and receiving data from another peer. On the screen depicted in Figure 3-42 (2) a button is displayed which enables P2P navigation. As soon as this button is clicked *DRPositioningActivity* calls the function *activateBluetoothConnection()*. Within this function the device is set to discoverable mode, making it visible to other Bluetooth-enabled devices in the vicinity. However the device not only waits for other devices to detect it but also searches actively for surrounding mobile phones.

If the device is detected by another phone it acts as client and the other device as master. In this case a new instance of class *AcceptBluetooth* is called. In the constructor of *AcceptBluetooth* the Bluetooth socket of the other device is stored as member variable. In its *run*-method inherited by class *Thread* it waits until the connection with the other device is established and calls then its function *manageBluetoothConnection*. Within this function a new instance of type *ManageBluetooth* is created. This class handles the incoming stream of the other device as well as the outgoing stream also in a thread. As soon as both streams are completely sent and received this class closes the connection and passes via an internal message the content of the streams to class *DRPositioningActivity*.

If the device detects another Bluetooth-enabled and visible device it acts as master in this connection and a new instance of type *ConnectBluetooth* is created. In the constructor of this class a *BluetoothServerSocket* is instantiated and in the *run*-method the connection to the *BluetoothSocket* of the other device is established. The data transfer is here also handled by a new instance of the threaded class *ManageBluetooth*.

Due to the use of threads for all communication and data transfer the mobile phone is continuously able to compute its position either by dead reckoning or by using GPS signals. It is also able to act as master and as client simultaneously retrieving position data from more than one peer as measurement input. If the device is currently able to compute its position via satellite signals it provides its position as measurement for other peers who may currently use dead reckoning for positioning but it does not use the input for filtering its own GPS position. The incoming position is then simply ignored. As already mentioned in 3.4.4 positioning is either performed using the dead reckoning algorithm or the GPS signals but never using both simultaneously.

To be able to send the position via a byte stream the data has to be prepared accordingly. The latitude and the longitude have to be written to the stream separately. If the covariance matrix additionally needs to be sent the elements of this matrix have to be written consecutively on the outgoing stream in a certain order. When receiving the byte stream it has to be read according to this order and a new matrix has to be instantiated which is filled with the values written on the incoming stream.

3.5 Evaluation of prototype

The evaluation of the application was performed on three different Android devices. All of them comprise a triaxial accelerometer and a triaxial magnetic field sensor. The LG Optimus black 3D and the Samsung Galaxy SII have a dual core processing unit while the Goolgle Nexus S uses a single core CPU. On the LG phone Android 2.3.3 was installed because up to now there was no update available from manufacturer side but the Nexus S and the Galaxy both use Android 4.0.

The prototype consists of three packages and therefore the evaluation is carried out according to this separation. The first section describes the test setup and the results for the dead reckoning algorithm. It follows the evaluation of the Kalman filter implementation. The last part is then the data transmission between the devices. Here also the overall test of the prototype is described.

3.5.1 Evaluation of the dead reckoning computation

The evaluation of the dead reckoning computation is separated in three parts. First part is the test of the step detection and step length estimation algorithm. The second part describes the heading estimation of the mobile phones and in the last part the evaluation of the overall position estimation is described.

3.5.1.1 Testing the step detection and step length estimation

To evaluate the accuracy of the step detection and the step length estimation a track with a known length is walked several times using a pre-determined *k*-factor. The measured walked distance is compared with the true walked distance to define the standard deviation of the step detector. To be able to estimate if the standard deviation is growing linearly with the length of the track first a short track with a length of 100 m is walked and then a track of 400 m is walked. This is done for all mobile phones which have been purchased. The estimation of the *k*-factor is done once on one of the mobiles and is then used on all mobile devices. Both tracks are walked on the running track of the University of Federal Armed Forces near Munich, see Figure 3-50. The green line marks the 100 m track and 400 m is one lap on the running track. For these tests only the comparison of the walked and estimated distance is of interest the heading and the computed position are irrelevant.

Estimated distance walked in meter									
		1st walk	2nd walk	3rd walk	4th walk	5th walk	6th walk	Average	%
100 m	LG Optimus Black	104.3	104.5	101.8	100.2	102.0	99.6	2.1	2.1%
	Samsung Galaxy	110.6	108.5	107.0	111.1	111.5	105.3	9.0	9.0%
	Google Nexus	111.8	106.1	106.3	95.2	98.1	92.8	1.7	1.7%
400 m	LG Optimus Black	389.2	405.2	330.9	428.6	438.9	442.2	5,8	1.5%
	Samsung Galaxy	457.0	446.6	465.6	468.0	473.6	453.5	60,7	15.2%
	Google Nexus S	414.7	434.0	427.1	390.6	402.1	402.7	11.9	3.0%

Table 3-29: The distance walked measured by step length estimation in meter and the difference to actually walked track length

As can be seen from Table 3-29 the accuracy of the step detection varies for the different mobile devices. While the LG and the Nexus S provide relatively useful results the measurements of Samsung Galaxy differs to some extent strongly from the actually walked distance. The total distance between measured and walked track length grows with the length of the track i.e. the error is larger when walking a longer track since the errors measured for all steps sum up. More tests to evaluate the step detection and the step length estimation were performed in the scope of [47] using the LG smartphone. The results are similar to those for the LG in Table 3-29. It is therefore necessary to derive the standard deviation of the step detection and step length estimation algorithm for each device individually.

Figure 3-50: Running track of the University of Federal Armed Forces Munich Germany. The green line marks the 100 m track.

3.5.1.2 Testing the heading estimation

In the above tests not only the measured distance is logged but also the position estimated using the dead reckoning algorithm described in 3.4.3.3. To have a reference especially for the 400 m laps additionally the GPS track is stored as well. The two positioning methods therefore do not alternate for this tests but run simultaneously. To estimate the orientation the measurements of magnetometer and accelerometer are used as described in 3.4.3.2 resulting in an azimuth computation which is used in the dead reckoning algorithm. The tests are also performed on the running track of the University. Low levels of noises influencing the magnetometer are expected in this area. The pictures below show some examples for tracks walked on the running lane. Although these six pictures are only examples the other results look very similar.

What is obvious from Figure 3-51to Figure 3-53 is the fact that the error induced by heading estimation is much more severe than the error induced by step detection and step length estimation. The position error introduced by the wrong estimation of the heading leads to a distance to the true position of several tenths of meters and partly even more. This could already be observed in the results of the Matlab simulation (compare 3.3.3). What can be observed from the pictures in Figure 3-51to Figure 3-53 as well is that for the Nexus S and the Galaxy a once measured and computed heading is kept for a very long time. This means that a heading which is estimated wrongly already in the beginning leads to a complete different track when computing a position by dead reckoning. Since both mobile devices are manufactured by Samsung it is very likely that similar sensors are integrated at least regarding the magnetometer since the Nexus S produces better results for the track length estimation which might be caused by a better or at least an accelerometer which is less sensitive to disturbances. The best results are obtained by the LG as can be seen in Figure 3-53. For the 100 m track the result looks very good and also for the 400 m track the result is not as bad as for the other mobile phones. Nevertheless also here some sort of correction has to be used to obtain a result which makes the navigation by dead reckoning useful. The influence of the Kalman filter on the position accuracy is reviewed in 3.5.2.

Figure 3-51: Test walks with the Samsung Galaxy SII. The blue line is the track measured by the GPS receiver and the pink track is the one computed by dead reckoning

Figure 3-52: Test walks with the Google Nexus. The blue line is the track measured by the GPS receiver and the pink track is the one computed by dead reckoning.

Figure 3-53: Test walks with the LG Optimus black 3D. The blue line is the tracked measured by the GPS receiver and the pink track is the one computed by dead reckoning.

Table 3-30 below depicts the average absolute difference between sensors measured heading and heading estimated by GPS data. As expected, the accuracy of the heading estimation is not influenced by the length of the track since the estimation of the heading does not depend on the measurement taken before. Also the average performance of the devices is not as different as one would expect from Figure 3-51 to Figure 3-53 with the exception of the outlier of the Nexus for the 100 m track which is mainly influenced by the strong diversion in walk 3. What can also be guessed from the table below is the fact that the heading estimation does not improve with each walk. A warm-up time for the sensors therefore seems not to enhance the results, since most of these walks have been performed in succession. The heading estimation based on sensor measurements is not as bad as expected and correlates with the assumptions made for the simulations in 3.3.3; but the wrong estimation of the heading is crucial especially when the sensor measurements are not able to adapt themselves to changes in direction of the user. Especially the two devices manufactured by Samsung seem to have difficulties with adapting to a new heading.

Differences between heading estimation in degree								
		1st walk	2nd walk	3rd walk	4th walk	5th walk	6th walk	Average
100 m	LG Optimus Black	11.6	92.5	38.2	33.43	18.4	28.1	37.04
	Samsung Galaxy	32.5	85.4	39.3	19.7	15.3	16.3	34.75
	Google Nexus S	67.2	54.1	225.5	56.0	37.9	24.3	77.5
400 m	LG Optimus Black	26.9	30.6	26.4	30.1	34.9	33.2	30.25
	Samsung Galaxy	31.4	27.4	27.2	43.3	25.7	58.4	35.66
	Google Nexus S	32.8	43.0	33.7	39.3	41.6	36.9	37.88

Table 3-30: The average absolute difference between measured heading and heading estimated by GPS data

3.5.1.3 Testing the position estimation

Every 1 s the current position estimated by dead reckoning and by GPS is logged as latitude, longitude and heading. This data is used to review the overall position accuracy of the dead reckoning algorithm. The position estimated by the GPS receiver of the device is used here as reference to define the position accuracy of the dead reckoning algorithm. As already depicted in 3.5.1.2 the position estimation strongly depends on the quality of the heading estimation. Since the LG Optimus had the best results regarding the heading estimation this device should be able to provide the best position estimations by dead reckoning. The walks which have been evaluated in regarding step length estimation and heading estimation in 3.5.1.1 and 3.5.1.2 are used another time to rate the overall position accuracy. In Table 3-31 the maximum measured distances between the dead reckoning position and the GPS position are listed. The LG Optimus shows the smallest average maximum distance for both the 100 m walks and the 400 m walks. The Samsung Galaxy and the Google Nexus S show especially for the 400 m track strong deviations from the GPS estimated position.

Maximum distance between true position and estimated position in km								
		1st walk	2nd walk	3rd walk	4th walk	5th walk	6th walk	Average
100 m	LG Optimus Black	0.008	0.009	0.012	0.012	0.013	0.016	0,010
	Samsung Galaxy	0.078	0.049	0.088	0.056	0.047	0.051	0.062
	Google Nexus S	0.094	0.006	0.150	0.087	0.045	0.040	0.070
400 m	LG Optimus Black	0.062	0.063	0.066	0.071	0.073	0.079	0.069
	Samsung Galaxy	0.156	0.178	0.111	0.128	0.142	0.131	0.282
	Google Nexus S	0.143	0.146	0.148	0.126	0.135	0.116	0.139

Table 3-31: The maximum distance measured between the GPS position and the estimated position in kilometer

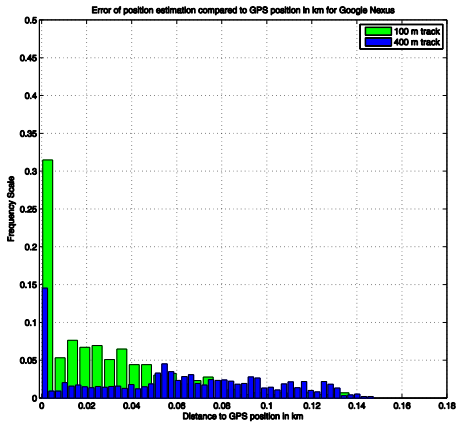


Figure 3-54: Frequency scale of the distance between estimated position and GPS position on the 100 m track (Google Nexus)

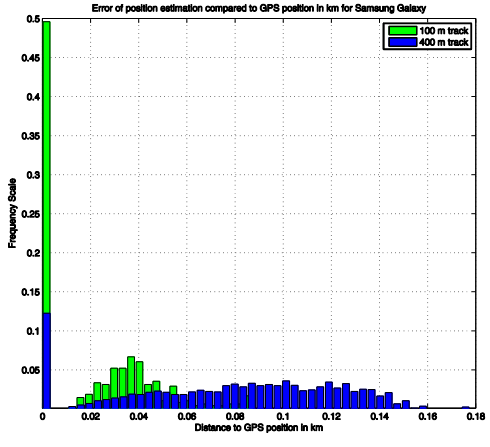


Figure 3-55: Frequency scale of the distance between estimated position and GPS position on the 100 m and 400 m track (Samsung Galaxy)

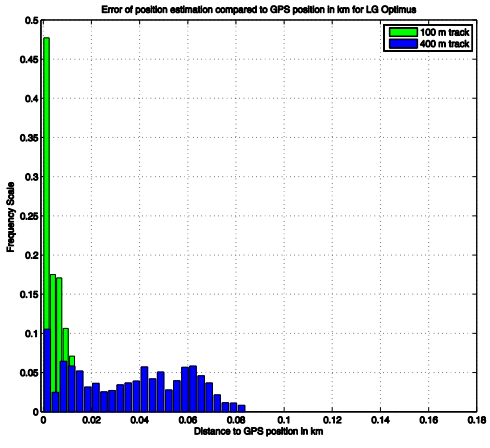


Figure 3-56: Frequency scale of the distance between estimated position and GPS position on the 100 m and 400 m track (LG Optimus)

In the plots depicted in Figure 3-54 to Figure 3-56 the distances between the dead reckoning position and the GPS position measured during all walks separated into the 100 m and the 400 m track are depicted as frequency distribution. It is obvious that the distances are smaller during the 100 m walk

than during the 400 m walk since the position error is not able to sum up for such a long time. The last two plots in Figure 3-57 and Figure 3-58 compare the performance of the different devices. Also these plots show that the LG Optimus is able to provide a better position accuracy than both other devices regarding the size of the distance between GPS provided position and dead reckoning position.

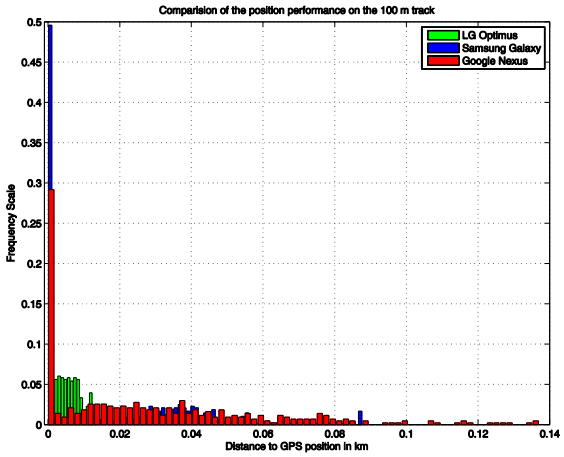


Figure 3-57: Comparison of the distance between estimated position and GPS position on the 100 m track for all devices

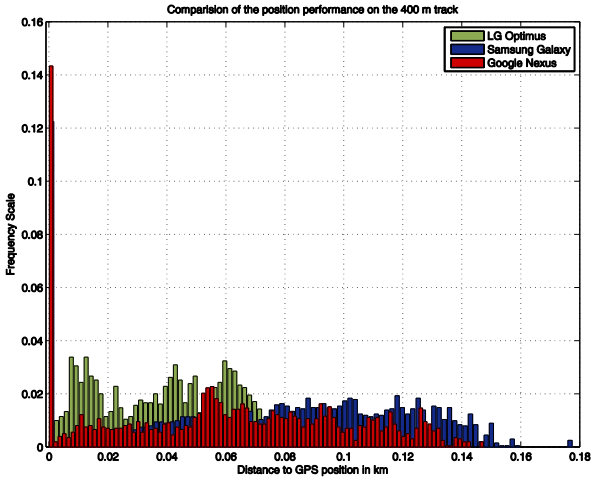


Figure 3-58: Comparison of the distance between estimated position and GPS position on the 400 m track for all devices

The following three plots have been chosen as example to compare the GPS position and the dead reckoning position within one 100 m walk and one 400 m walk. Additionally the difference between the heading estimated by the sensors and the heading provided by the class *Location* is depicted. Unfortunately it is not published how this bearing is computed but it is very likely that the bearing between two successive positions is estimated. The first plot of the figures below shows the position in latitude, longitude for GPS positioning (black) and positioning by dead reckoning (red). The plot in the middle depicts the absolute distance between the computed positions and the last plot illustrates the measured heading (red) and the heading estimated by GPS data (black).

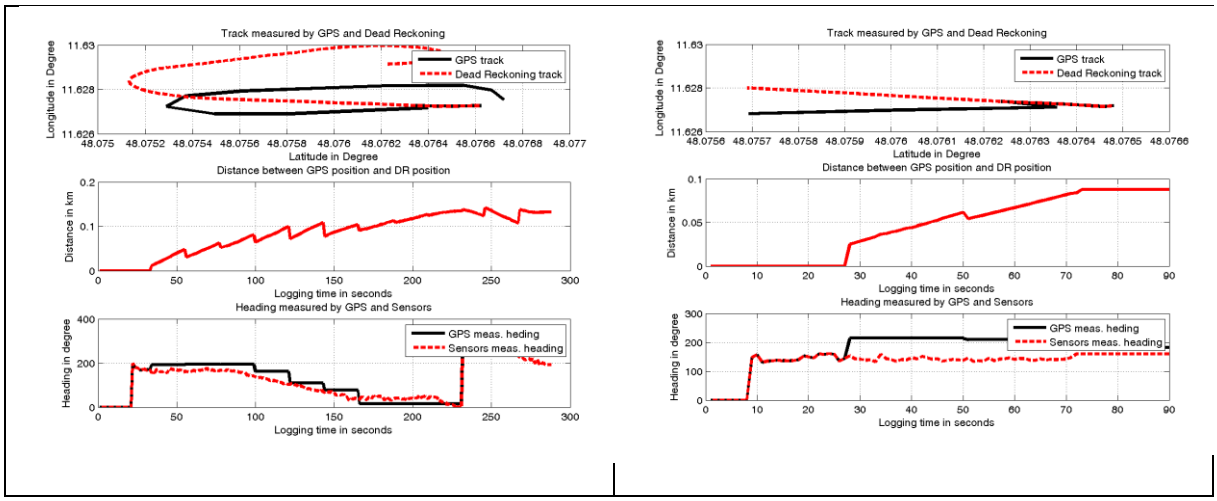


Figure 3-59: Measurements of the Samsung Galaxy; left side: 400 m track; right side: 100 m track

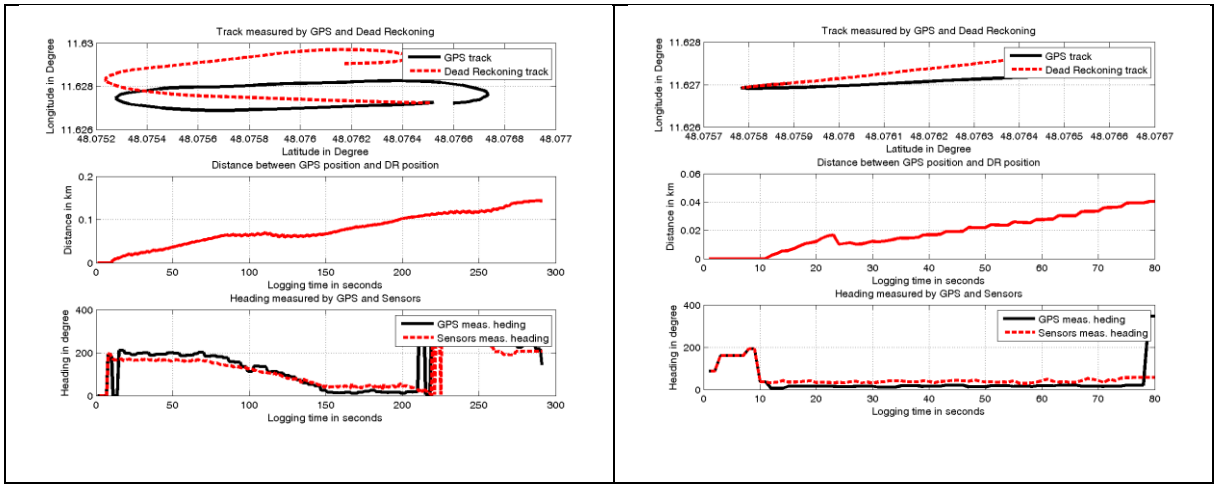


Figure 3-60: Measurements of the Google Nexus S; left side: 400 m track; right side: 100 m track

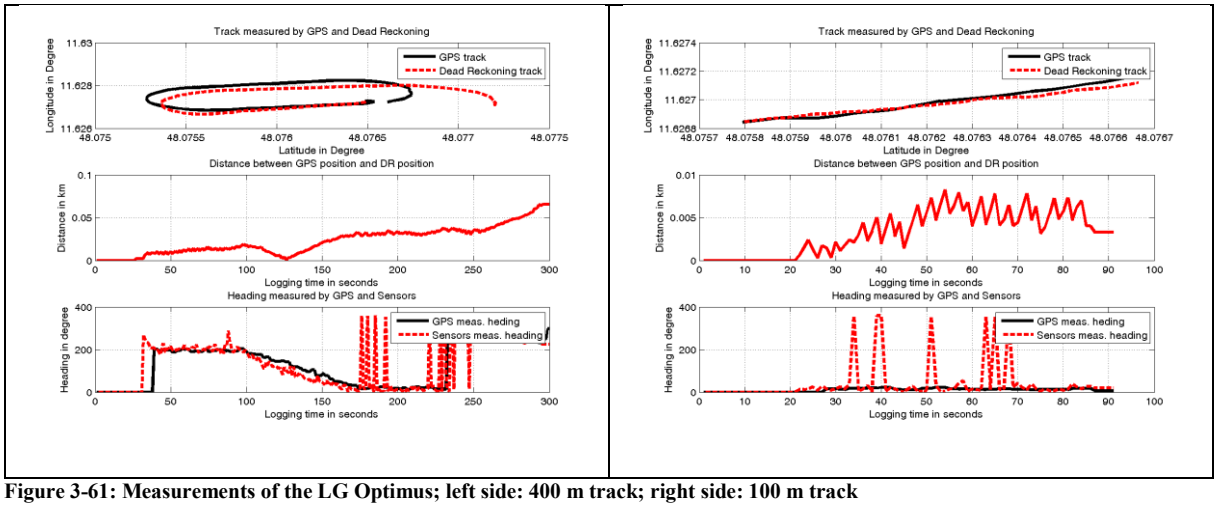


Figure 3-61: Measurements of the LG Optimus; left side: 400 m track; right side: 100 m track

In contrast to the two Samsung devices whose heading computation seems to be very lazy, the heading estimation of the LG device is rather sensitive resulting in spikes as depicted in Figure 3-61. Due to this the distance to the position estimated by GPS is much smaller than for the other two devices resulting in a better position estimation by dead reckoning. This means that the Kalman filter has to be adapted to the particular characteristics of the mobile phone. While the filter of the LG device can trust its

measurements coming from the sensors the filters of the two Samsung devices should trust the measurements less and rely stronger on the measurement function.

During the test walks it became also obvious that the accuracy of the magnetometers is subject to deviations. This is not only caused by distorting electromagnetic sources but also by the dynamics of the Earth's magnetic field. Although the LG smartphone provided useful results on some days and in certain environments on other days it performed similar to the Samsung smartphones. It could also be noted that the performance of all mobile devices became better when the orientation computation was started before the actual measurements were performed but also this did not always improve the heading accuracy. However, it is likely that the average user will not process a warm-up for the sensors nor will she watch fluctuations in the Earth's magnetic field. Dead reckoning relying only on the sensor measurements is therefore with today's embedded sensors not possible.

Fortunately it is often not necessary that the user has to walk for such a long time completely without access to GPS signals as it is suggested by the 400 m lap above. Rather the user makes use of the mobile phone for navigation when walking in urban areas, entering and leaving stores. In this environment GPS signals are only unavailable for a short time and dead reckoning can be used to bridge times of signal outage. The behavior and performance of the three mobile devices was tested as well in such situations and is reviewed below. Also here strong differences regarding the quality of GPS receiver can be determined. As mentioned in 3.4.3 the application notes the unavailability of GPS signals when no NMEA sentences containing a fix can be received anymore and starts computing the position via sensor measurements based on the last GPS position fix. While the LG Optimus and the Google Nexus S relatively fast lose GPS signals and start computing the position based on DR the Samsung device is able to maintain the connection to satellite signals even when turning off all data connections. Unfortunately the position estimation is then very inaccurate and differs strongly from the real position. The pictures below depict the behavior of all devices when entering and leaving a building. As usual before dead reckoning can be used a GPS position outside of the building has to be obtained.

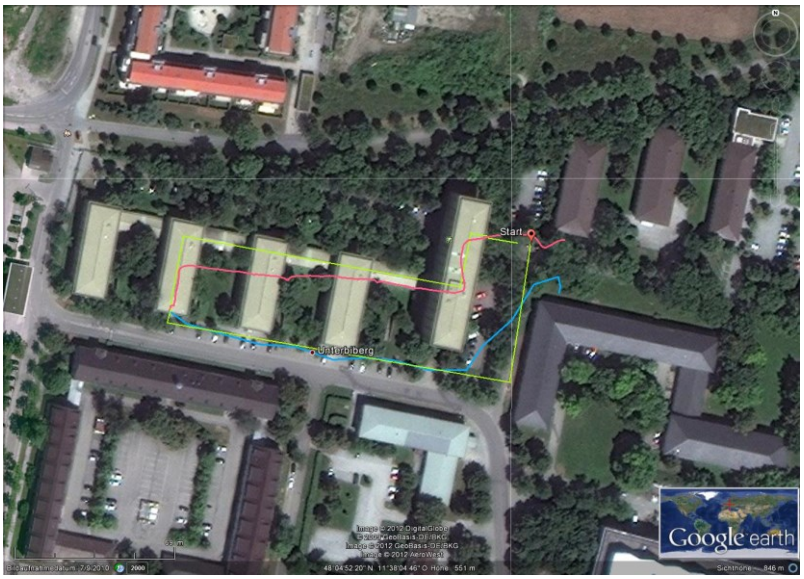


Figure 3-62: Combination of outdoor and indoor environment walked with the LG Optimus. Blue indicates the GPS computed track, pink the dead reckoning track and the green line the true track walked.

According to Google Earth the distance between the estimated track and the real track has a maximum distance of 13.24 m (see Figure 3-62).

Figure 3-63: Combination of outdoor and indoor environment walked with the Samsung Galaxy. Blue indicates the GPS computed track and the green line the true track walked.

In contrast to the other two devices the Samsung Galaxy needs a smaller time interval to recognize the absence of GPS signals. Figure 3-63 above depicts the same indoor-outdoor track walked with the Galaxy. As already discussed in the sections before, this device performs worse than the LG with a maximum distance to the true track of almost 28 m. But in contrast to the results of the indoor-outdoor walk with the Google Nexus S (see Figure 3-64) the result might still be good enough when using a P2P Kalman filter.

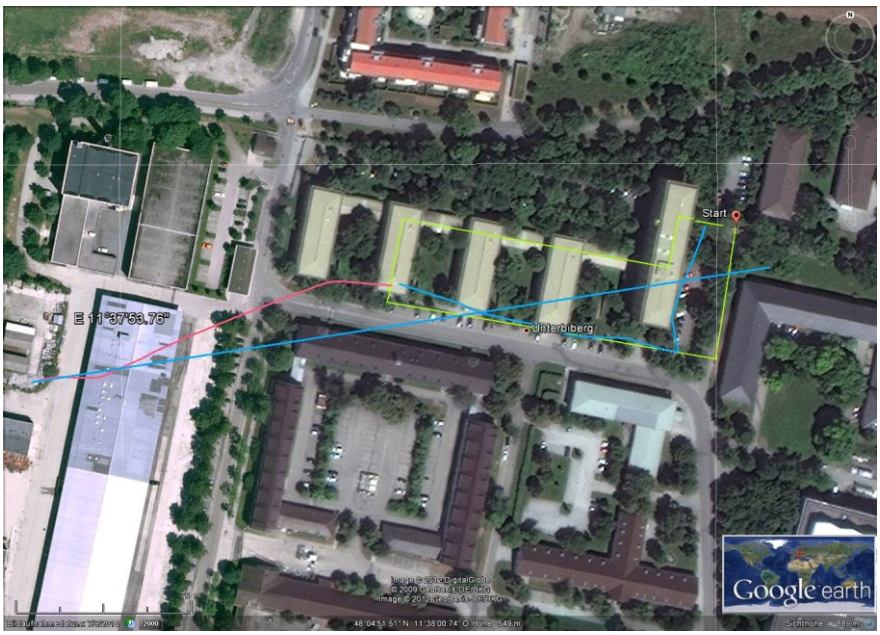


Figure 3-64: Combination of outdoor and indoor environment walked with the Google Nexus. Blue indicates the GPS computed track, pink the dead reckoning track and the green line the true track walked.

As already discussed in 3.5.1.2 the Google Nexus S is extremely lazy when estimating the orientation of the device. A once defined heading is not changed during the walks and accordingly low is the position estimation accuracy. It is very unlikely that a Kalman filter can make strong changes since the

measurements are too noisy. If the P2P exchange of position data makes a difference for this device has to be tested. These tests are reviewed later in section 3.5.3.

3.5.2 Evaluation of the Kalman filter algorithm

To test the Kalman Filter new walks were performed to compare the position estimated by filtered dead reckoning and GPS positioning. The implementation of the Kalman filter as explained in 3.4.4 was used with small modifications. Since the performance especially regarding the heading estimation differs from device to device the user is able to enter the standard deviation of the heading estimation and the standard deviation of the step length estimation for each use. Additionally the user is able to define if for the initial estimate of the heading directly the sensor measurements are used or if the bearing estimated by GPS is used. For the Samsung and the Google device it might help to use the GPS estimated heading as initial value while the LG device should be able to rely on the sensor measurements. Nevertheless both options have been tested in several walks.

Maximum distance between true position and estimated position in km										
			1st walk	2nd walk	3rd walk	4th walk	5th walk	6th walk	7th Walk	Average
	LG Optimus Black	DR only	0.082	0.106	0.044	0.105	0.088	0.037	0.048	0.073
		KF	0.082	0.109	0.037	0.062	0.047	0.029	0.038	0.058
	Samsung Galaxy	DR only	0.038	0.036	0.048	0.029	0.038	0.042	0.078	0.044
		KF	0.034	0.042	0.023	0.022	0.035	0.036	0.059	0.036
	Google Nexus S	DR only	0.072	0.251	0.094	0.045	0.057	0.062	0.158	0.106
		KF	0.077	0.243	0.099	0.055	0.056	0.066	0.154	0.107

Table 3-32: The maximum distance measured between the GPS position and the dead reckoning only position and the Kalman filtered position in kilometer

Average differences between heading estimation in degree										
			1st walk	2nd walk	3rd walk	4th walk	5th walk	6th walk	7th Walk	Average
	LG Optimus Black	DR only	98.36	123.62	91.74	124.45	150.36	79.19	90.00	108.75
		KF	37.68	86.07	24.78	48.08	94.30	15.12	34.85	48.70
	Samsung Galaxy	DR only	85.36	101.44	138.36	153.61	148.87	90.85	100.03	116.93
		KF	22.67	17.23	30.12	56.44	41.21	28.84	37.82	33.47
	Google Nexus S	DR only	110.42	78.93	100.58	147.35	103.52	117.07	141.83	114.25
		KF	49.01	49.83	23.00	39.00	75.30	36.23	66.88	48.22

Table 3-33: Differences between heading estimation in degree for dead reckoning only and Kalman filtered heading

Seven walks have been performed with all mobile devices and the results are depicted in Table 3-32 and Table 3-33. Table 3-32 presents the maximum distance in km measured throughout the whole walk for dead reckoning only and for the Kalman filtered position. In Table 3-33 the average differences of the heading as measured by the sensors only and by filtered measurements compared to the bearing estimated from GPS are indicated in degree. It can be noticed from the tables that the use of a Kalman filter using only the measurements of the sensors is able to improve the position in almost all cases. Since this was not tested in the Matlab simulations this is an important fact for this approach. The walks were alternately processed using the bearing coming from the GPS position estimation as initial value for the heading in the state vector and simply the heading computed by the sensor measurements. According to the results in both tables there is no difference if the first or the later variation is used. The plots below (Figure 3-65, Figure 3-66) confirm the results in the tables: The overall position accuracy is better when a Kalman filter is used compared to plain dead reckoning except for the Google Nexus S smartphone. Compared to the tests in 3.5.1.3 the Samsung Galaxy produces results which are partially better than those of the LG Optimus. This might be caused by better conditions regarding especially the Earth's magnetic field since not all tests have been carried out on the same days.

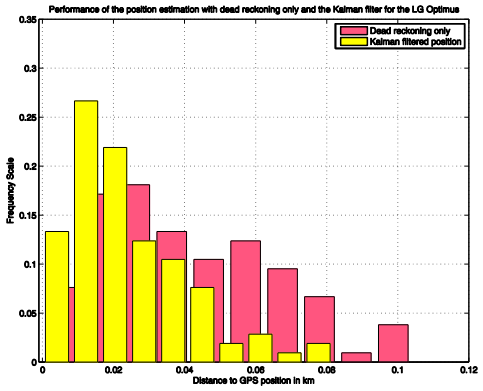


Figure 3-65: Frequency scale of the distance between estimated position, Kalman filtered position compared to GPS computed position on different walks (LG Optimus)

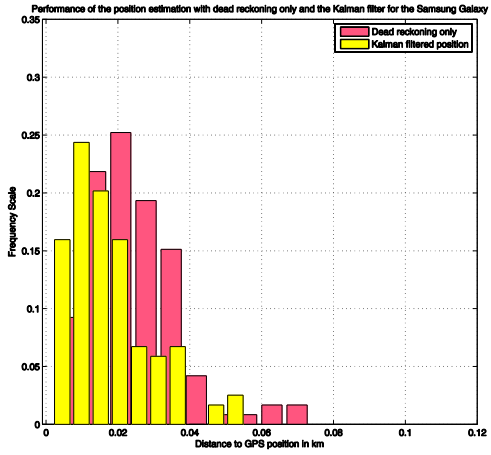


Figure 3-66: Frequency scale of the distance between estimated position, Kalman filtered position compared to GPS computed position on different walks (Samsung Galaxy)

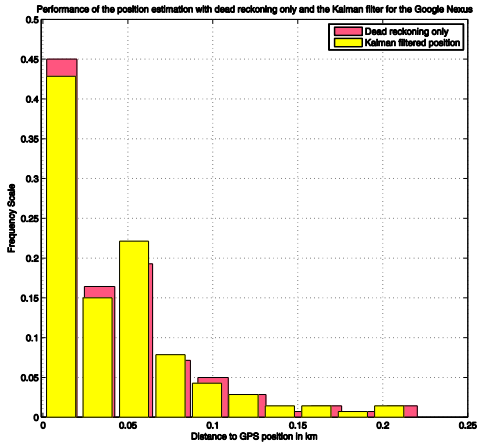


Figure 3-67: Frequency scale of the distance between estimated position, Kalman filtered position compared to GPS computed position on different walks (Google Nexus)

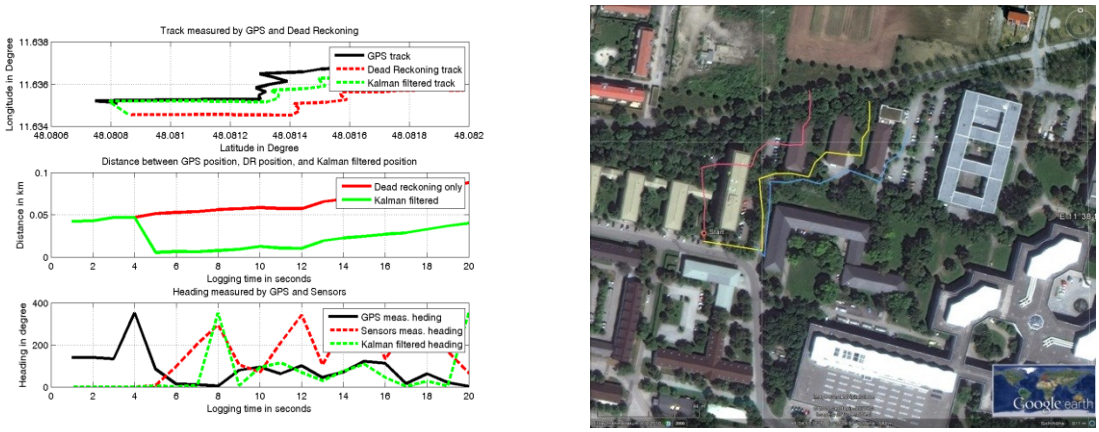


Figure 3-68: Measured track during walk 5 of the LG Optimus. The map on the right side depicts the GPS track in blue, the dead reckoning track in pink and the Kalman filtered track in yellow.

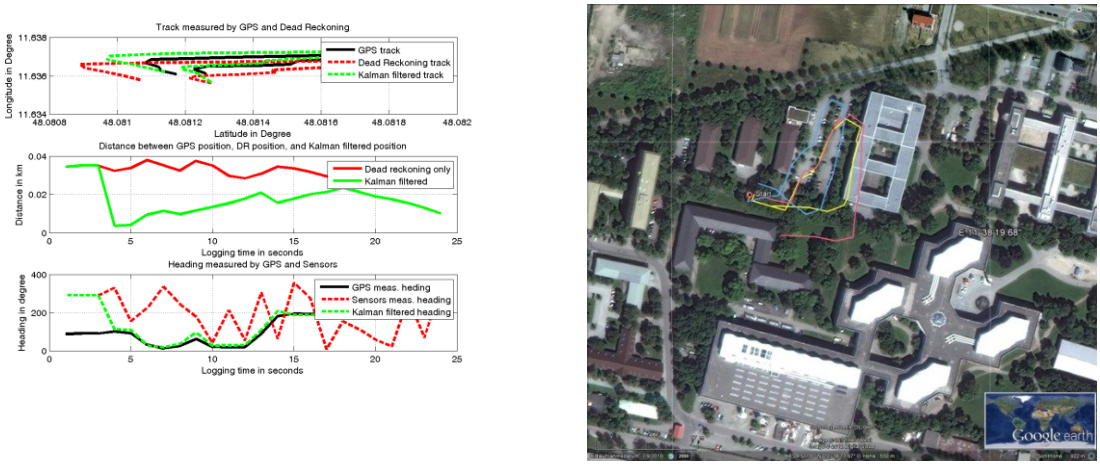


Figure 3-69: Measured track during walk 5 of the Samsung Galaxy. The map on the right side depicts the GPS track in blue, the dead reckoning track in pink and the Kalman filtered track in yellow.

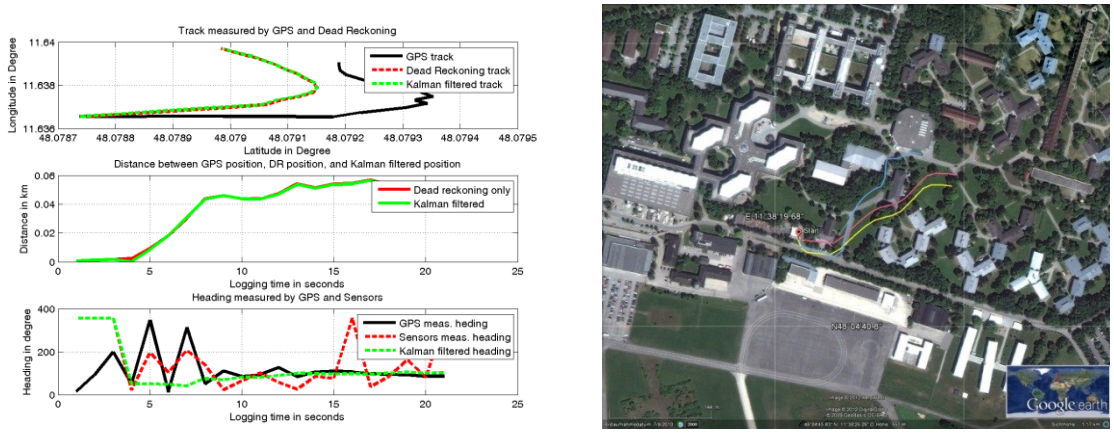


Figure 3-70: Measured track during walk 5 of the Google Nexus S. The map on the right side depicts the GPS track in blue, the dead reckoning track in pink and the Kalman filtered track in yellow.

Figure 3-68 to Figure 3-70 illustrate examples for certain walks with the different smartphones. The first plot shows the position in ellipsoidal coordinates for the GPS position, the dead reckoning only position and the Kalman filtered position. In the second plot the distance between the GPS position and the dead reckoning position with and without Kalman filtering is depicted. The last plot presents the difference

between heading measured by sensors only, filtered heading, and GPS computed heading. Accordingly the tracks are depicted in Google World aside.

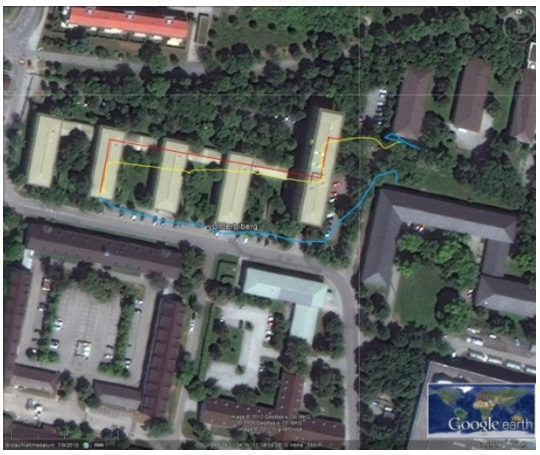


Figure 3-71: Combined outdoor-indoor track of the LG Optimus. The red line indicates the true path through the building, the blue track depicts the GPS estimated position and the yellow line the filtered dead reckoning position.

Figure 3-72: Combined outdoor-indoor track of the Google Nexus. The red line indicates the true path through the building, the blue track depicts the GPS estimated position and the yellow line the filtered dead reckoning position.

Figure 3-73: Combined outdoor-indoor track of the Samsung Galaxy. The red line indicates the true path through the building, the blue track depicts the GPS estimated position and the yellow line the filtered dead reckoning position.

Although the use of a Kalman Filter improves the result of the dead reckoning position estimation in almost all cases the position accuracy is still too vague to use it for indoor navigation purposes especially when regarding the Google Nexus S device. Even with the Kalman filter the dead reckoning position sooner or later starts to drift away from the GPS position which is used as reference here. This is especially true if the mobile device has no access to GPS signals for a long time. All walks described in this section at least comprised 250 steps causing a strong deterioration of the dead reckoning position compared to the GPS position even when deploying a Kalman filter.

Figure 3-71 to Figure 3-73 depict some additional tests which have been carried out where indoor and outdoor environment are alternated. This was done on the one hand to test if the mobile phone is able to switch autonomously from GPS positioning to dead reckoning and vice versa. On the other hand it was tested if the position accuracy improves when the user is only for a short time in an indoor environment. The tracks depict now the filtered dead reckoning position only since the results above show that the position accuracy is better anyway or in case of the Google Nexus that there is no difference. The red track indicates the true track through the building since no GPS signals are available anymore to compare the tracks. According to the tracks all mobile devices were able to switch between dead reckoning and GPS positioning. The LG device is able to maintain relatively accurate position estimation throughout the test. The Nexus S and the Samsung devices were in contrast not really able to provide a position usable for LBS.

3.5.3 Evaluation of the P2P communication

In 3.2.3 different standards are described which are candidates to enable the wireless peer-to-peer connection for this application. For the implementation of the prototype application initially the Bluetooth standard was chosen (3.4.5) to realize the communication between the peers. The reasons were that Bluetooth is a relatively widespread standard which is used not only in smartphones but also in several other mobile devices, that therefore the Android API to implement Bluetooth communication links provides very well-defined interfaces and functions, and that the transmission range which is possible is not too large. In 3.3.3.1.11 it could be shown that if the distance between the peers who exchange position data is too large, the P2P Kalman filter rather deteriorates the estimation of position if the amount of peers increases. The transmission range should therefore be around 5 m which can be provided by Bluetooth piconets indoors.

As described in 3.4.5 the implementation of the Bluetooth communication interface is not a problem but during testing the prototype some issues occurred which conflict with one requirement related to the P2P Kalman Filter discussed in 1.2. It was demanded that the switch between dead reckoning and GPS and the communication between the peers is done automatically and without interaction of the user. This not possible when using Bluetooth for peer communication. First the user has to allow the visibility of the smartphone actively. The smartphone can then be found and contacted by other devices but it stays in this state only for 120 s. After this time the user again has to turn on the visibility of the smartphone. Secondly the user has to allow each incoming connection. Obviously this is not a good choice for an application which is based on the frequent exchange of data with other devices. It would not be possible to do something else with this device anymore. Only changes in the Bluetooth standard directly would enable the use for this application.

Parallel to this thesis therefore some research has been performed in the project work of a student using WiFi direct to realize the peer-to-peer communication. Although WiFi direct was already integrated into the Android 4 system published in June 2012 the API to use this communication standard is still not completely implemented. Since the cell sizes are very large (at least 50 to 100 m depending on the environment) it is necessary to limit the connections to peers who are in the direct vicinity otherwise the position accuracy degrades due to the mutual filtering (see 3.3.3.2.11). This could be realized by querying the RSS of incoming requests. There exists a method to request the current signal strength of WiFi direct connections in the Android API but up to now it is not completely implemented and returns only a static value. The first idea to provide a work around of the current limitation was to connect only

to devices via WiFi direct which are also visible via Bluetooth since those are necessarily in the direct vicinity. Up to this moment none of the mobile phones had problems to execute the application but it turned out that the simultaneous searching of other devices via Bluetooth and WiFi direct had an immense impact on the performance. Another work around had therefore to be found.

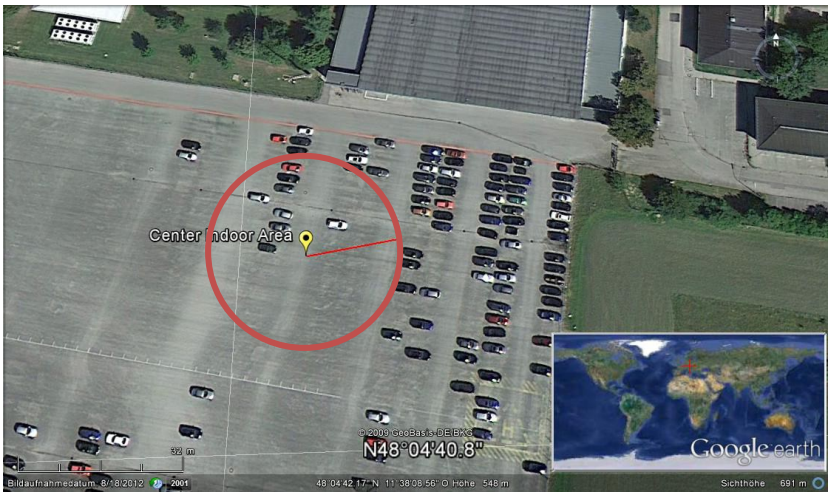


Figure 3-74: The test area for the P2P Kalman Filter with the "indoor" area

To test the overall performance of the P2P Kalman Filter application several tests were performed in the scope of the project work of a student. To observe the performance regarding the position accuracy when allowing mutual filtering it was necessary to process the tests outdoors to be able to compare the position estimation based on dead reckoning with the position estimation based on GPS signals. A circular area with a radius of 20 m was defined as "indoor area" (see Figure 3-74). The participants of the tests resided outside of this indoor area and acquired a position based on GPS signals. As soon as the participants "entered" the indoor area the device uses the last known GPS position as a start position for the filtered dead reckoning navigation. The device continues to compute its current position via GPS but this process is completely decoupled from the position estimation via dead reckoning and only used to evaluate the PDR algorithm. Due to the processing of the tests outdoors the current limitations of the WiFi direct implementation could be bypassed by estimating the distance between the peers via their current GPS estimated position. A maximum distance of 5 m was allowed for data exchange.

WiFi direct currently also has restrictions regarding the connection to other devices which have not been connected before. There are attempts from users' side requesting to withdraw those limitations but up to now the first connection to another device has to be allowed by the user manually. However, in contrast to Bluetooth WiFi direct has the advantage that a once accepted peer can be connected without user interaction. Additionally peers connected within a WiFi direct group are able to exchange data directly with each other regardless if they act as host or as standard group member. For the test of the application the device of one of the participants was therefore denoted as host of a WiFi direct group. The other users connected to this device as clients and were then able to exchange data with peers in the vicinity avoiding additional user interaction.

One pre-test was carried out with three participants. During those tests it could be observed that the especially the constant connection with other peers via WiFi direct had a strong impact on the power supply of the mobile devices. Although for the tests different devices have been used (all deployed with Android 4.x) all of them were out of power after 15 minutes of testing. During the tests described in 3.5.1 and 3.5.2 the battery also drained faster than during normal use when performing longer tests but never this quickly. A second test was carried out with five participants and the application partly froze during the tests due to the heavy load of the WiFi direct connections. Sometimes also the logging of the dead reckoning data was affected and stopped after some time. It was therefore hard to obtain useful data depicting the functionality of the P2PKF. Since a constant connection to other peers is not foreseen

for a later realization this does not mean that the P2PKF is in general not performing as expected according to the results of the simulation. The problem is merely caused by the current limitations of the processing power and restrictions regarding WiFi direct.

To enable the evaluation of the performance it was therefore decided to test the P2PKF implementation in post processing. Eight GPS tracks and the output of the dead reckoning algorithm (step detection, step length and heading) without computing the position were recorded during walks in the area depicted in Figure 3-74. All tracks started outside of the indoor area and entered after some time the indoor area. To process the recorded data the Java application developed in the scope of [93] which was used for simulations similar to those described in 3.3 was modified. The position estimation by dead reckoning starts when the peer enters the indoor area and stops when the indoor area is left. In between the recording of the GPS position continues but is decoupled from the position estimation by dead reckoning. The exchange of position data between the peers for mutual filtering is therefore simulated based on the current position computed by GPS signals to bypass the exchange of data by an ad-hoc network. In Figure 3-75 the eight tracks of the peers are depicted in two separate pictures.

Based on the example of peer 1 the improvement of the position estimation by filtered dead reckoning is illustrated. In Figure 3-76 peer 1 is alone and only able to filter the position estimation with own measurements. In this case the filtering is not beneficial when regarding the yellow track which depicts the filtered track. It is sometimes even worse than dead reckoning only. In Figure 3-77 two peers were present (peer 2) and by exchanging position data with this peer for mutual filtering the position estimation of peer 1 is improved to below 5 m. With the presence of two more peers (peer 3 and peer 4) the filtered track is constantly better than the track estimated by dead reckoning only. In Figure 3-79 all eight peers are present in the scenario. From the plot on the right side it is obvious that the position error in the indoor area most of the time is less than 15 m which almost achieves the accuracy suggested in 2.1.2 for location based service in indoor areas. With more peers a constant position accuracy of less than 15 m seems possible.

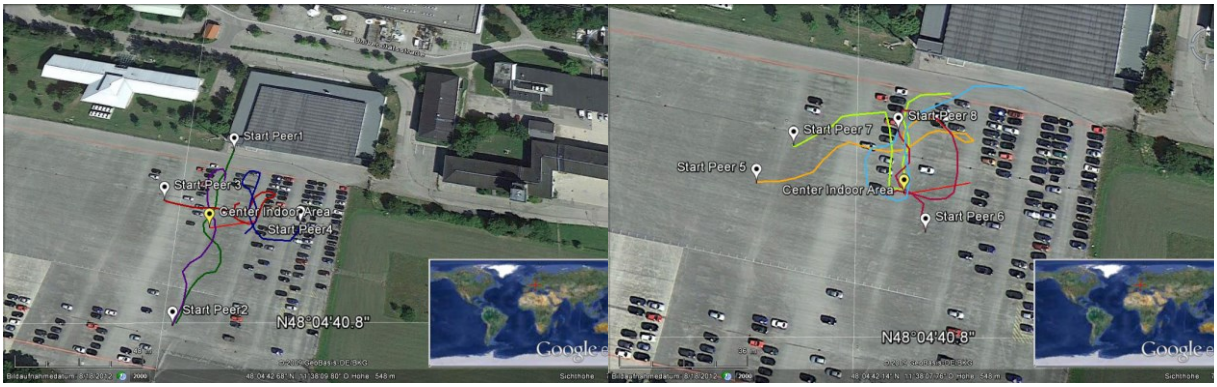


Figure 3-75: The GPS tracks of the eight peers recorded for post processing.

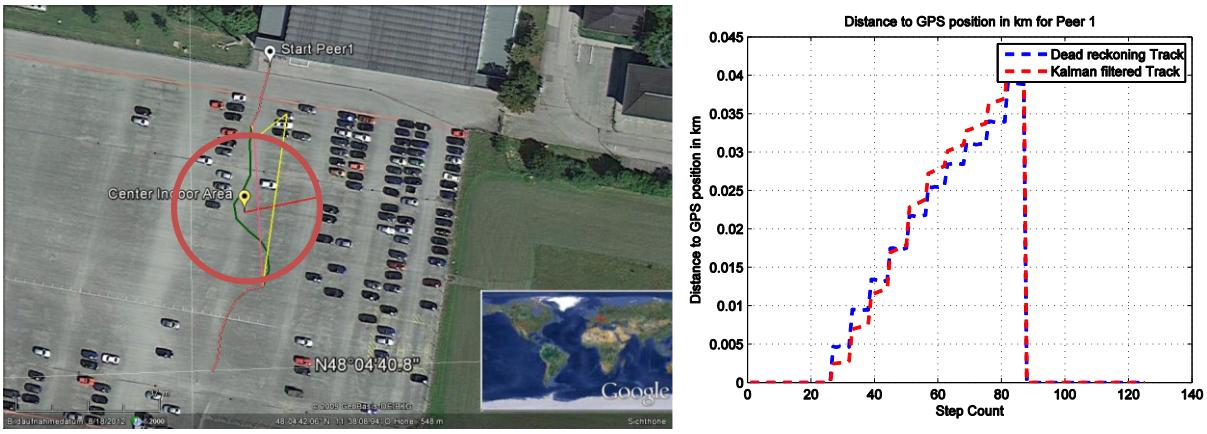


Figure 3-76: On the left side the GPS estimated track of peer 1 in green, the dead reckoning only track in pink and the filtered track in yellow. On the right side the position error of dead reckoning and filtered track in km. One peer was present in this scenario.

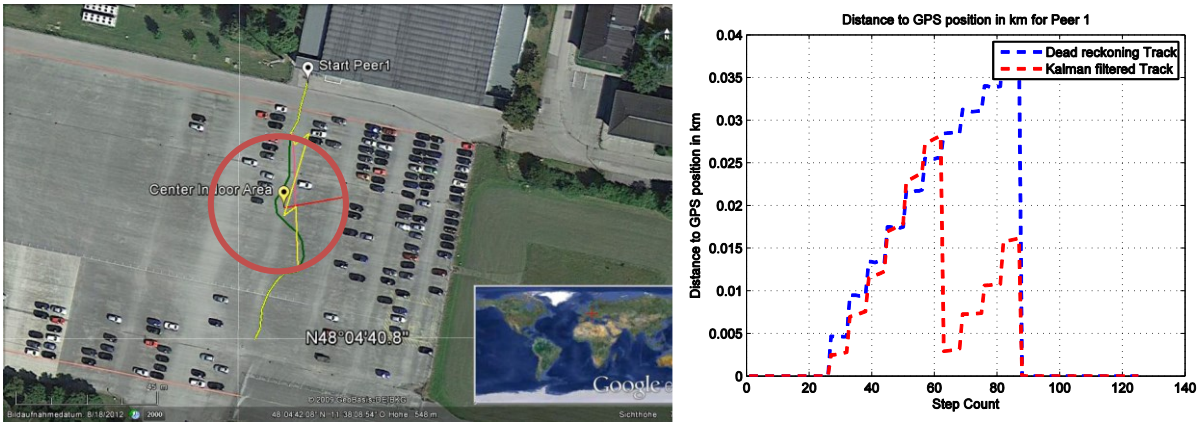


Figure 3-77: On the left side the GPS estimated track of peer 1 in green, the dead reckoning track only in pink and the filtered track in yellow. On the right side the position error of dead reckoning and the filtered track in km. Two peers were present in this scenario.

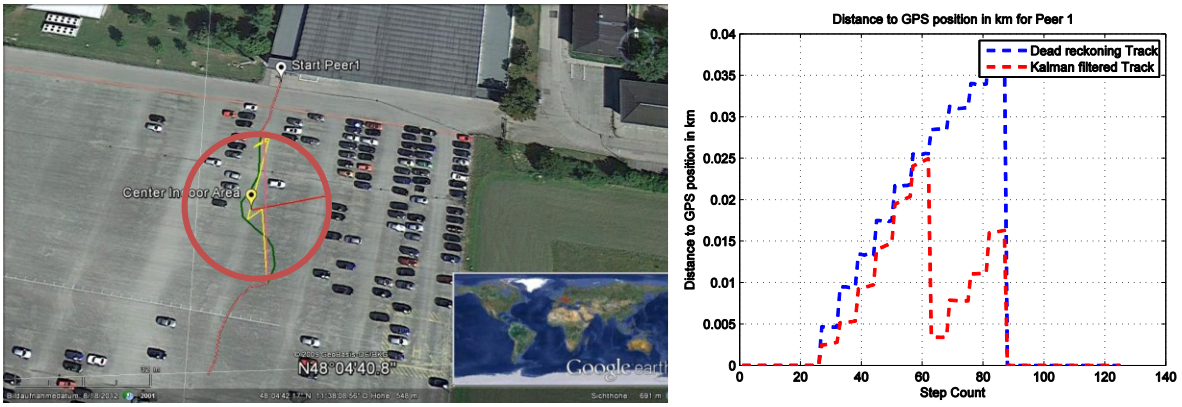


Figure 3-78: On the left side the GPS estimated track of peer 1 in green, the dead reckoning track only in pink and the filtered track in yellow. On the right side the position error of dead reckoning and the filtered track in km. Four peers were present in this scenario.

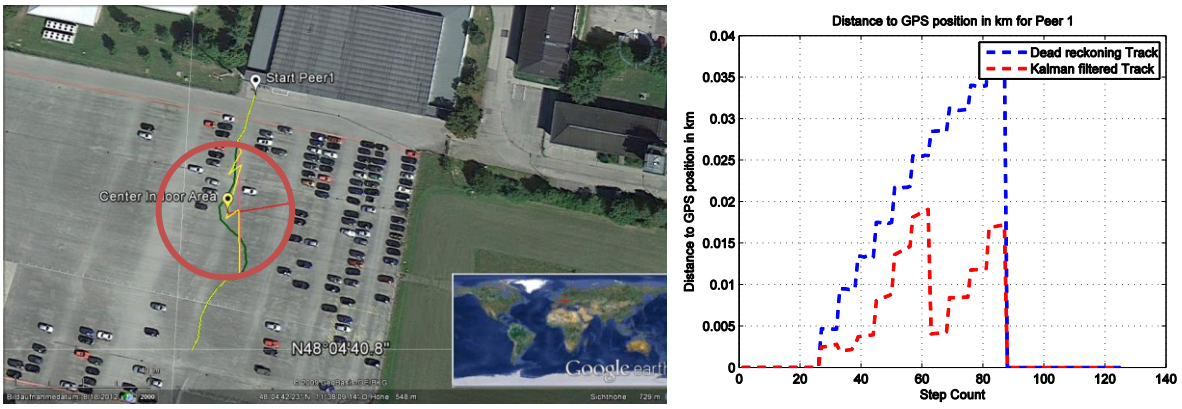


Figure 3-79: On the left side the GPS estimated track of peer 1 in green, the dead reckoning track only in pink and the filtered track in yellow. On the right side the position error of dead reckoning and the filtered track in km. Eight peers were present in this scenario.

For the other peers the results are similar. With more peers present the position accuracy increases as depicted in Table 3-34. Regarding the results the average position error is below 10 m which enables LBS most of the times for indoor environments. This is of course also depending on the size of the indoor area and the time the peer resides within.

	Alone	2 peers	3 peers	4 peers	5 peers	6 peers	7 peers	8 peers
Peer 1	11 m	6.2 m	5.6 m	5.6 m	5.6 m	4.4 m	4.5 m	4.5 m
Peer 2		3.2 m	3.2 m	3.2 m	3.2 m	3.1 m	2.4 m	2.6 m
Peer 3			5.6 m	5.6 m	5.6 m	4.7 m	4.6 m	4.6 m
Peer 4				1.3 m	1.3 m	1.3 m	1.3 m	1.2 m
Peer 5					2.1 m	2.1 m	1.8 m	0.97 m
Peer 6						4.0 m	3.9 m	3.9 m
Peer 7							2.6 m	2.9 m
Peer 8								6.3 m

Table 3-34: Average distance in meter between GPS computed position and filtered position related to the current amount of peers in the scenario.

4. SUMMARY

The importance of smartphones in everyday life is likely to continue to grow regardless who will be the next global player providing the smartphone or mobile device of the future. It is therefore the question how the user experience of such a device can be improved even more. Already today most users feel that several tasks of their everyday life are facilitated by different functionalities of a smartphone. There is no need any more to remember every date and appointment, the smartphone reminds its user regardless of location and time. It is also possible to access documents, photos, movies etc. from every point in the world using a smartphone. It is not necessary to print walking directions at home to find the way from one location to another. The use of smartphones for such tasks becomes more and more naturally for a growing amount of users. For this reason the navigation functionality of a smartphone should not stop at the door of a building.

The original idea of the P2P Kalman filter was to contribute to already existing strategies to enable seamless indoor positioning and navigation. The P2P Kalman filter is based on dead reckoning using inertial sensors which are already or soon will be standard in smartphones. It is obvious from the first description of this approach in 1.1 to the evaluation of the concrete implementation in 3.5 that the position accuracy of all users only benefits from mutual filtering if a considerable large amount of users participate. This is only conceivable if the P2P Kalman filter becomes one of the standard applications which are on the smartphone at delivery. The user must be able to turn this application on and off depending on the current status of the mobile device. But generally the PDR process and the mutual filtering should run in parallel to the GPS receiver to be able to provide seamless navigation everywhere regardless of the environment.

To accomplish this kind of acceptance from the users as well as from the manufacturers the requirements that have been developed in 1.2 have to be fulfilled by the concrete implementation. In the following section each requirement is therefore discussed separately by means of the implementation which was introduced in 3.4 and validated in 3.5.

Although the functionality of the P2P Kalman filer was evaluated in the simulations in 3.3 and in 3.5, there exist still some issues which could not be addresses by the present implementation. The reasons for this range from the state-of-the-art of the operating system Android and the currently available hardware to the fact that some of these issues are beyond the scope of this thesis. These topics are discussed in the second section of this chapter.

4.1 Analysis of the requirements

In section 1.2 six requirements have been established the implementation of the P2P Kalman filter has to address or which are necessary to provide mutual filtering. In this section it is examined if these requirements could be fulfilled by the prototype application introduced in 3.4 and the tests 3.5. Each requirement is in the following separately reviewed.

1. Navigation and positioning should also work deep indoors.

The dead reckoning algorithm which is based on step detection, step length estimation, and heading estimation by inertial sensors which is proposed here is not dependent on satellite signal data which is not able to permeate walls and ceilings of a building (compare 2.2.5). Therefore it is possible to obtain a position with this application also in deep indoors. However in the tests made in 3.5.1 the dead reckoning algorithm alone does not provide position accuracy that enables reasonable navigation or location-based services. The additional use of a Kalman filter algorithm improves the position estimation to a certain degree but the error is still too large for navigation. Not until the mutual filtering is allowed the position estimation reaches an accuracy (compare 3.5.3) that enables location-based services and navigation indoors. Still it is obvious that more accurate sensors and additional measurements coming from the gyroscope, the camera or a barometer for 3D positioning would improve the dead reckoning algorithm and therefore also the overall position accuracy for all peers. Nevertheless position estimation based on inertial sensors embedded in a smartphone alone will not be able in the near future to provide position accuracy that satisfies the user's needs related to pedestrian indoor navigation.

2. The user must not pay additional costs for data transmission of navigation assistance.

The data transmission via WiFi direct as well via Bluetooth is free of costs for all users. No additional costs have to be paid when connecting to other peers to exchange position data for the mutual filtering. Of course the P2P Kalman filter does not exclude the mostly charged connection to a location server to receive AGPS data. As stated in 1.1 and 3.1 the idea is that some peers have access to better position data due to better sensors, better receivers or better data and that the position estimation of these peers is weighted heavier in the mutual filtering algorithm than the position estimation of peers with less advanced devices or no internet connection. But the exchange of position data between the peers is free of costs.

3. The P2PKF should be independent from centralized entities.

As described in 2.4 and especially in 2.4.3 there are several methods to enable indoor positioning based on radio signals but they all have the drawback that centralized entities are necessary to compute the position of a peer and for some approaches the position even has to be stored in this centralized unit (often a server) e.g. RF-ID tags. This poses not only a thread on the user's privacy but can also be a problem when many users simultaneously want their position to be computed and sent back to them. This problem also occurs for AGPS in MS-assisted mode (compare 2.3.1) since one location server has to provide assistance to an area of about 100 km coverage [15]. Unfortunately it is not possible to obtain information about the network traffic caused only by the request of assistance data but in urban areas it is very likely that there are several requests for navigation assistance. This might not break down the location server but results under certain conditions in long response times for the user.

The fact that the position data or information to compute the current position is transmitted to a centralized unit which might store this and the following position updates along with the name of the user does not inspire trust. As a matter of fact in spring in 2011 there were several articles online but also in newspapers suspecting Apple and Google to not only store the locations of a user in a file and a database respectively on the mobile device but also transmit this data back to servers owned by Google or Apple [96]. It is still not clear what the purpose of the storage of the locations is anyway and if this

file is actually transmitted personalized to servers. Nevertheless this illustrates that centralized units might indeed threaten user's privacy. Of course this is also an issue for the P2PKF since position data is exchanged between users that do not know each other. However the P2PKF proposes an autonomous way to estimate the position. Even the exchange of position data and its mutual filtering is handled in a decentralized way.

4. The mobile device is equipped with inertial sensors to estimate step length, step and heading and a satellite navigation receiver.

This is a basic requirement of the P2P Kalman filter. The estimation of the user position is dependent on the measurements of integrated inertial sensors in the smartphone. In contrast to inertial sensors which are used in standard inertial navigation the MEMS integrated in the smartphone provide very noisy measurements and therefore the estimated position is very noisy as well. The sensors can therefore not be regarded as reliable especially when considering the results in the tests described in 3.5.1. Nevertheless the here introduced approach provides a position estimate based on the MEMS integrated in the mobile phone. No additional equipment is necessary only the units which are standard in today's mobile devices (compare 2.5.2.2) are used for the position estimation and the exchange of data. It is expected that the measurement accuracy of MEMS still can be improved. However the reason for the often noisy measurements is not alone based on the quality of the MEMS but also on the high density of electromagnetic sources in the smartphone itself. Due to the strong convergence of technology in one device even more measurement noise can be caused although the MEMS are improving. As stated before the positioning based on dead reckoning can further be enhanced by integrating measurements from other sensors (e.g. barometer) and hardware (e.g. camera). Still the idea is to use the smartphone as it is installing only software in form of an application on it.

5. The mobile device must be able to communicate with other mobile devices within an ad-hoc network.

This is the core of the P2P Kalman filter. Since inertial navigation based on the built-in sensors is not able to provide a position estimate accurate enough for navigation or location-based services as already explained in detail in the paragraphs before the mutual filtering of the position data in a Kalman filter algorithm is necessary to improve the position accuracy. Therefore interfaces to provide access to radio ad-hoc networks like WiFi direct, ZigBee, and Bluetooth are necessary. The ZigBee standard was considered for the P2P Kalman Filter but it is still unfamiliar to a large extent and therefore not integrated into current smartphones although it would be applicable to use it for data exchange of the peers especially regarding its network structure, see 3.2.3.2.

Bluetooth was implemented first to enable the communication between the peers but showed some serious deficits related to its security policy. The user has to allow each incoming connection request and the device is only visible for a certain time. As described in 3.5.3 this is not acceptable for this application although Bluetooth provides a beneficial cell size, a well-defined interface and a large popularity.

WiFi direct was developed and published only recently therefore this standard is not very common yet. Nevertheless it is a suitable candidate for this application since the security mechanisms are not as strict as for Bluetooth. One draw is the relatively large coverage area and the fact that it is only available for mobile phones which are updated to at least Android 4. However modules which provide Bluetooth communication are common in each smartphone and also WLAN functionality can be found in almost all smartphones today and it is expected also for future generations of devices. More complicated is the fact that, due to the current limitations of WiFi direct, all peers have to reside within one group. The maintenance of this permanent connection uses too much processing power and disables the processing of other tasks of the P2PKF let alone that other applications can run simultaneously on this mobile phone. With the upcoming of mobile devices with more efficient processing power and potential changes of the WiFi direct implementation the P2PKF could be processed as planned on smartphones.

Nevertheless it is necessary to control the amount of concurrent connections to enable the use of the smartphone for other tasks than positioning.

6. The P2PKF should enable a continuous positioning when moving from outside into indoors and vice versa. The survey of indoor area should not be necessary to enable positioning.

As described in 3.5.2 the switch between outdoor and indoor positioning can be performed by the application automatically. This behavior is based on the reception of NMEA strings in the mobile phone. As soon as within a certain time interval no NMEA data is received anymore the devices switches to dead reckoning based on the last position estimated with GPS data (see 3.5.2). For the user this does not make a difference. The application continues to compute a position and provides it to the user. Also when leaving an indoor area, the application is able to change from inertial navigation back to navigation via satellite signals. In 2.5.3 it was mentioned that the application currently does not support the filtering of the GPS position with the position estimated by inertial sensors. This is an issue which should be addressed in future work.

4.2 Future work

The issues that could not be addressed by the present prototype implementation coarsely can be separated in two main topics: Modifications regarding the PDR algorithm and modifications regarding the P2P Kalman filter algorithm to obtain better positioning accuracy. Both topics are discussed in the following sections.

4.2.1 Improvement of the PDR

The dead reckoning algorithm which is implemented for the P2P Kalman Filter provides especially when used without filtering very poor positioning results at least for some devices (compare 3.5.1). It is therefore beneficial to improve the position estimation of the dead reckoning algorithm for the whole P2P Kalman filter application.

Some improvements have already been mentioned in this work for the prototype application. One of these improvements is the additional use of a gyroscope to stabilize the heading estimation based on the magnetic field sensor. Since the largest error is introduced by a poor heading estimation this would help to improve the position accuracy of the dead reckoning algorithm. The use of an additional sensor and a filtering algorithm which is implemented to integrate the gyroscope measurements into those of the magnetic field sensor however causes higher power consumption and asks for a fast CPU for the computation of a new position and in parallel the establishment of P2P connections. Nevertheless according to the results of the simulation 3.3.3 the position accuracy of all present peers benefits even if some peers have less accurate sensors. It is also very likely that the CPU power for smartphones continues to increase. Currently there are already the first smartphones on the market comprising a quad-core CPU. Therefore even computationally intensive algorithms sooner or later will be executable on smartphones.

Due to this development the integration of more sensor measurements into the dead reckoning algorithm will be enabled. The PDR of the P2P Kalman filter currently provides only two-dimensional positioning. To extend the position estimation a sensor providing the height is necessary. As explained in 2.5.1.2 the integration of a barometer which reacts on changes in atmospheric pressure to estimate the current height is not yet a standard in smartphones. However barometers are already employed in GPS receivers for hiking to obtain better altitude estimation than with GPS alone. As mentioned in 3.4.3.1 the API to access sensors that are not yet integrated as standard like e.g. the barometer is already implemented. It is therefore only a matter of time until this sensor is integrated into smartphones as well. But not only inertial sensors could be used to enhance the position estimation. There are several approaches which address the use of e.g. cameras for positioning. The integration of such measurements is only a question of processing power and since this will probably be enhanced with each new generation of smartphones these measurements can be used to improve positioning in GNSS-denied environments.

Another issue is the way the device is carried by the user. For this prototype application it is assumed that since the user wants to navigate the device is carried relatively steady in the hand. Currently there seems to be no other way than to fix the smartphone somewhere to the body of the user be it the hip, the head, the foot or the hand to enable dead reckoning with inertial sensors. It is very unlikely that this can be easily changed. However what should be a matter of future research is the possibility for the user to pack the device away e.g. in a pocket or a bag, to later pull it out again, and to be able to continue to navigate with it. The smartphone should be able to “feel” that it is stored away and stop positioning by dead reckoning since the output of the algorithm is unrelated to the actual movement especially regarding the heading. It should also be possible to detect changes in the movement of the user like switching from running to walking and adapt the step detection and step length estimation algorithm automatically by using different k-factors that have been calibrated before. There exist some works ([97] [98]) addressing ways to sense the current type of movement of the user some of which could be used to enhance the dead reckoning algorithm.

4.2.2 Improvement of the P2P Kalman filter

The present implementation of the P2P Kalman filter is based on the results obtained in the simulations with Matlab. Due to these results is was decided to not implement the Unscented Kalman Filter since the results were worse than for the Extended Kalman Filter and in contrast to the EKF the UKF could only be implemented using a workaround to enable more than one measurement. Due to this implementation the position accuracy could be improved compared to simulations where no mutual filtering was processed but not in the same degree as for the EKF. Nevertheless there are several variations of the Discrete Kalman Filter to process non-linear operations and it is conceivable that these could be adapted for the P2P Kalman filter. Since the CPU power of the smartphone sooner or later will not be a limiting factor anymore it is possible to implement even more complex filter computations or even particle filters which could be used for mutual filtering as well as for the dead reckoning algorithm. It might be also possible to implement two independent filtering algorithms, one for PDR and another one for the mutual filtering. Simultaneously it is possible to allow devices with slower CPUs use simpler filtering algorithms like a weighted average. As long as the parameters which are exchanged via P2P network are the same for all peers (position and position variance) different filters in each device are an option. If the position accuracy still benefits in these scenarios needs to be examined carefully.

Another issue is the integration of the sensor measurements into the GPS solution. It was assumed for the simulations that GPS and PDR can be used simultaneously if GPS signals are available. During the implementation of the dead reckoning algorithm it turned out that the position estimation of the dead reckoning algorithm would be of no use to improve the GPS position even in GNSS-challenged areas. With the improvements of the PDR algorithm suggested in section 4.2.1 the position solution by GPS might benefit in indoor areas with limited access to satellite signals. The possibility to use sensor measurements to improve the GPS position solution is available for Android phones by default and can be turned off and on by the user according to her needs. Nevertheless nothing is known about the way the sensor measurements are integrated into the GPS position solution. It is very likely that the accelerometer is used to improve the estimation of the velocity and the magnetic field sensor or the gyroscope for providing an additional measurement for the heading but this can only be guessed. It is therefore beneficial if the P2P Kalman Filter enables the integration of sensor measurements automatically when the device navigates with GPS to be able to pitch in if the signal power degrades and to adopt the positioning process completely when no more signals are available.

A problem which occurred during the evaluation of the prototype was the establishment of connections between the peers. Bluetooth is very protective regarding the privacy of the user. Although the privacy mechanism based on the input of a PIN by the user is often bypassed by the manufacturers it is not possible to establish connections automatically between peers which have not yet been connected to each other even for a certain service. Also the fact that a smartphone is only visible for a certain amount of time constrains the use of the P2P Kalman filter. At least this constraint can be turned off by the user. But it became obvious that also WiFi direct is not an optimal choice at least in its current status. Even here the first connection request has to be acknowledged by the user and the access to values like e.g. radio signal strength was at least at the time this thesis was written not yet implemented in the Android API. The access to the current status of WiFi direct links like the signal strength only is a matter of time since the API already comprises appropriate function stubs. To solve the problem regarding connection restrictions there are attempts by users and developers using the Android OS to ask for fewer restrictions regarding the acknowledgement of connection requests when the request belongs to a certain service which has been approved by the user before. Now it is a matter of wait and see if Google Inc. decides to comply with these requests. In 2.6 other cooperative positioning strategies have been introduced which deal with the same problem. Also here a real solution was not developed.

Privacy is an issue of the P2P Kalman Filter as well which could not be addressed by the implementation of the prototype. Especially as long as this topic is not solved this might prevent the use of this application. Since currently the devices are connected to each other by name due to the constraints of Bluetooth and WiFi direct it is possible to link a position coming from another device with a certain

person. Due to the constraint of the P2P Kalman filter which allows the exchange of position data only with peers that have not been used for filtering the last time to avoid correlated results it is not easily possible to follow a certain person but it might be conceivable. One way to solve this problem would be to wait until this issue sorts itself out when the protection of the privacy is not a matter anymore for the average user because everything is already posted in social networks. But this should not be a real option. A better solution would be the development of a method to disguise the position exchange between the peers. Of course the user should be aware that her position data is transmitted to other users and there has to be the option as well to turn this application off but the received position itself must not be visible to the user. The ideal case would be that the user is aware of the fact that this application runs on her device but is not bothered by connection requests and not able to read the received position from the display. Also the received position should not be stored in any way in the source device. To draw conclusions about the original version of the position data from the newly calculated one is possible but should be avoided by all means.

The prototype which was introduced in this thesis tried to address as many issues as possible but as can be seen from the sections above there are still several topics left which are either directly or indirectly related to the P2P Kalman Filter. Nevertheless it is the author's opinion that P2P networks and therefore also P2P positioning will become an important issue in the next years. On the one hand this is related to the fact that the distribution of smartphones continues to grow and that the handling and establishment of ad-hoc networks will become easier and therefore more ordinary for the users. With a growing amount of users the demand for more applications and an even easier handling also regarding navigation is increasing as well. For this reason also more effort should be put into research regarding indoor positioning with end-user products. It is very likely that future users will not accept that they are not able to user their devices seamlessly for outdoor and indoor navigation.

5. APPENDIX

The understanding of two important topics is required to be able to comprehend all parts of this thesis. The first topic comprises the different coordinate systems and the way to transform one system into another and the second one the idea of OO-programming. Both topics are addressed in the first and the second section respectively of this appendix. In the third section the list of abbreviations used in this thesis can be found and in the fourth section the sources are listed which are cited in this work.

5.1 Coordinate frames and transformation

5.1.1 Coordinate Frames

As mentioned before the difficulty of the *strapdown* algorithm lies in the transformation of the attitude and acceleration from the coordinate system of the body to inertial coordinates. In [22] and [5] four different coordinate systems are described:

1. **Body-Frame (b-frame):** The axes are adjusted relative to the body. The origin is in the center of mass of the body. The x-axis is normally defined as forward (usual direction of travel), the z-axis follows the direction of gravity (down) and the y-axis points to the right. If angular motion is described in this frame the x-axis is the roll axis, the y-axis the pitch-axis and the z-axis the yaw axis.

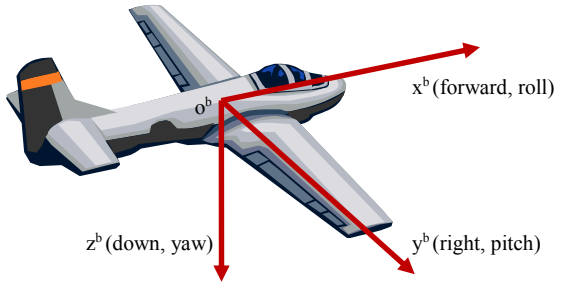


Figure 5-1: The axes of the body frame

2. **Inertial coordinate frame (i-frame):** The origin is the earth's center of mass. The y-axis is oriented along the earth's spin axis pointing to the geographic North Pole. The y- and x-axis lie within the equatorial plane with an angle of 90 degrees in between. Since this is an inertial coordinate system the axes do not follow the rotation of the earth. A common solution to establish this coordinate frame is to define the direction of the x-axis by the direction from the earth to the sun at the vernal equinox. Also the movement of the pole which is following a circular path with a radius of 15 m [5] has to be compensated. Therefore a solution is to relate to the IERS (International Earth Rotation and Reference System Service) reference pole (IRP) or the conventional terrestrial pole (CTP). An inertial frame adopting IRP/CTP and is otherwise established according to the rules described above is known as the inertial reference system (CIRS).

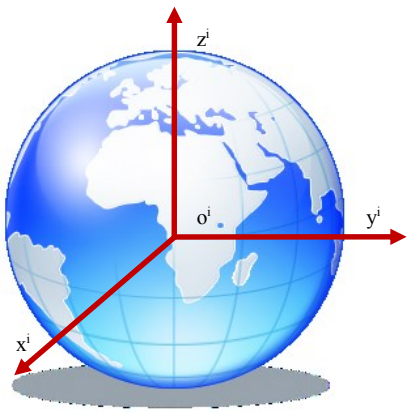


Figure 5-2: Axes of the inertial frame

3. **Earth-Centered Earth-Fixed Frame (ECEF, e-frame):** The origin of this frame is the same as for the i-frame: the earth's center of mass but here the axes are fixed relative to the earth. The z-axis points along the spin axis to the geographic North Pole. The x-axis is defined by the direction from the center to the intersection of the equator with the IERS reference meridian (IRM) or the conventional zero meridian (CZM) defining the 0 degree longitude. The y-axis points to the intersection of the equator with the 90 degree east meridian.

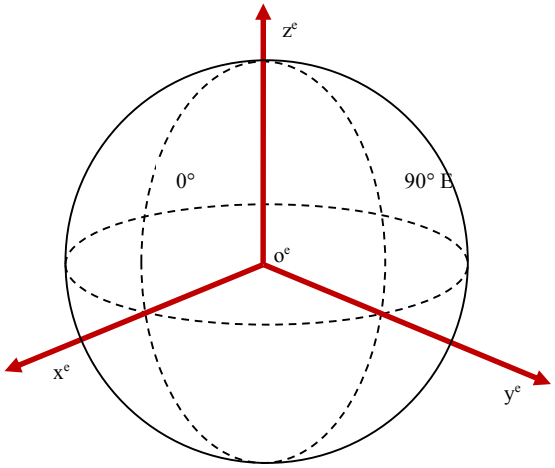


Figure 5-4: Axes of the e-frame

4. **Local Navigation Frame (n-frame):** Its origin is at the point the coordinate frame is used for. This can be a user, a vehicle or a navigation system. The z-axis is the normal to the surface of the reference ellipsoid pointing roughly toward the center. The x-axis points to the north orthogonal to the z-axis and tangential to the surface of the earth. The y-axis' direction is to the east orthogonal to both other axes and also tangential to the earth's surface. There exist also other representations with x pointing to east, y to north and z has an upward direction.

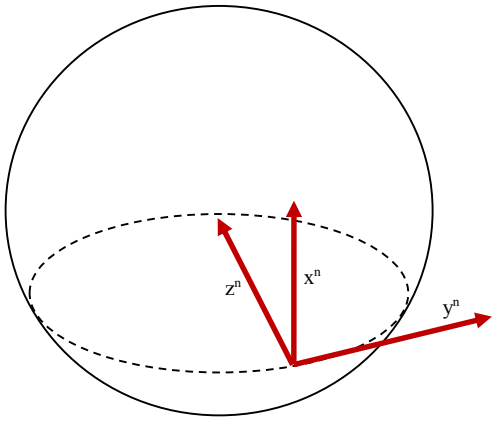


Figure 5-3: Axes of the local navigation frame

5.1.2 Transformation

5.1.2.1 Euler Attitude and Coordinate Transformation Matrix

To transform the coordinates from one coordinate system into another system different possibilities exist. The so called Euler Attitude provides one way to process the transformation. The attitude consists of three successive rotations called yaw, pitch and roll. These rotations have the same names as the axes of the body frame but are used here for the conversion from one coordinate system into another. The transformation from one system to another is explained with the help of one example from [5]: The axes of the $\beta - system$ are transformed into axes of the α-frame. The first rotation is the yaw-rotation ψ_{bn} around the z-axis of the body frame which results in an intermediate coordinate system with new x-axis and y-axis:

$$x^{\psi} = x^{\beta} \cos\psi_{\beta\alpha} + y^{\beta} \sin\psi_{\beta\alpha} \tag{5-1}$$

$$y^{\psi} = -x^{\beta} \sin\psi_{\beta\alpha} + y^{\beta} \cos\psi_{\beta\alpha} \tag{5-2}$$

$$z^{\psi} = z^{\beta} \tag{5-3}$$

The pitch rotation θ_{bn} is then performed around the y-axis of the intermediate coordinate frame and results into another intermediate coordinate frame:

$$x^{\theta} = x^{\psi} \cos\theta_{\beta\alpha} + z^{\psi} \sin\theta_{\beta\alpha} \tag{5-4}$$

$$y^{\theta} = y^{\psi} \tag{5-5}$$

$$z^{\theta} = x^{\psi} \sin\theta_{\beta\alpha} + z^{\psi} \cos\theta_{\beta\alpha} \tag{5-6}$$

Finally the roll rotation ϕ_{bn} is performed resulting in the final coordinate frame:

$$x^{\alpha} = x^{\theta} \tag{5-7}$$

$$y^{\alpha} = y^{\theta} \cos\phi_{\beta\alpha} + z^{\theta} \sin\phi_{\beta\alpha} \tag{5-8}$$

$$z^{\alpha} = -y^{\theta} \sin\phi_{\beta\alpha} + z^{\theta} \cos\phi_{\beta\alpha}$$

The transformation between one frame into another can be denoted using a vector [5]:

$$\boldsymbol{\psi}_{\beta\alpha} = \begin{pmatrix} \phi_{\beta\alpha} \\ \theta_{\beta\alpha} \\ \psi_{\beta\alpha} \end{pmatrix} \tag{5-9}$$

The Euler angles are enumerated in reverse order. It is important to mention that the order as described in the paragraphs above has to be followed strictly. If the angles are applied in another order the result is an erroneous transformation. Since the Euler rotation $(\phi_{\beta\alpha} + \pi, \pi - \theta_{\beta\alpha}, \psi_{\beta\alpha} + \pi)$ gives the same result as $(\phi_{\beta\alpha}, \theta_{\beta\alpha}, \psi_{\beta\alpha})$ the pitch is limited to $-90° \leq \theta \leq 90°$. A coordinate transformation matrix C_{β}^{α} can be expressed as function of the three Euler angles [22]:

$$C_{\beta}^{\alpha} = \begin{pmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} \quad \textbf{(5-10)}$$

A reverse way to determine the Euler rotation using the transformation matrix also exists:

$$\phi = arctan2(c_{32}, c_{33}) \quad \textbf{(5-11)}$$

$$\theta = \arcsin(-c_{31}) \quad \textbf{(5-12)}$$

$$\psi = arctan2(c_{21}, c_{11})$$

By defining three differential equations the change of the Euler rotation with respect to time and based on angular rate $\omega_{\beta\alpha}^{\beta}$ can be expressed:

$$\dot{\phi} = (\omega_{\beta\alpha,y}^{\beta}\sin\phi + \omega_{\beta\alpha,z}^{\beta}\cos\phi)\tan\theta + \omega_{\beta\alpha,x}^{\beta} \quad \textbf{(5-13)}$$

$$\dot{\theta} = \omega_{\beta\alpha,y}^{\beta}\cos\phi - \omega_{\beta\alpha,z}^{\beta}\sin\phi \quad \textbf{(5-14)}$$

$$\dot{\psi} = \frac{(\omega_{\beta\alpha,y}^{\beta}\sin\phi + \omega_{\beta\alpha,z}^{\beta}\cos\phi)}{\cos\theta} \quad \textbf{(5-15)}$$

Although the Euler rotation provides a very intuitive way to transform one system into another it is hardly used because of the occurrence of a singularity when the pitch angle is about $\pm 90°$ and yaw and roll become indistinguishable [5]. It is easy to see from equation (5-1) that in this case the result of the diversion is undefined. Also the computation of yaw and roll using equation (5-2) and (5-3) respectively is then not possible.

5.1.2.2 Orientation Vector and Quaternion Attitude

The elements of an orientation vector $\vec{\sigma}$ display the attitude of two coordinate frames to each other. The orientation vector indicates the rotation axis and its magnitude the angle of rotation. One coordinate frame can be transformed by one rotation around the orientation vector into another. Two possibilities exist for the transformation. If the orientation vector is defined by:

$$\vec{\sigma}_1 = (\sigma_{1,x}, \sigma_{1,y}, \sigma_{1,z})^T \quad \textbf{(5-16)}$$

With length $\sigma_1 = |\vec{\sigma}|$, the orientation vector

$$\vec{\sigma}_2 = -\frac{\vec{\sigma}_1}{\sigma_1}(2\pi - \sigma_1) \quad \textbf{(5-17)}$$

describes the same situation. The idea is that the rotation with angle σ_1 can be described by a right-hand rotation as well as a left-hand rotation with angle $2\pi - \sigma_1$.

To derive the change of the orientation vector based on the angular rate $\omega_{\beta\alpha}^{\beta}$ this equation has to be solved:

$$\dot{\vec{\sigma}} = \omega_{\beta\alpha}^{\beta} + \frac{1}{2}\vec{\sigma} \times \omega_{\beta\alpha}^{\beta} + \frac{1}{\sigma^2}(1 - \frac{\sigma \sin\sigma}{2(1-\cos\sigma)})\vec{\sigma} \times (\vec{\sigma} \times \omega_{\beta\alpha}^{\beta}) \quad \textbf{(5-18)}$$

The derivation of this differential equation can be found in [99].

A normed orientation vector with length $|\mathbf{q}| = 1$ is called a quaternion:

$$\mathbf{q}_{\beta}^{\alpha} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \cos(\frac{\sigma}{2}) \\ (\frac{\sigma_x}{\sigma})\sin(\frac{\sigma}{2}) \\ (\frac{\sigma_y}{\sigma})\sin(\frac{\sigma}{2}) \\ (\frac{\sigma_z}{\sigma})\sin(\frac{\sigma}{2}) \end{pmatrix} \quad \textbf{(5-19)}$$

Element *a* represents the rotation angle and the other three components describe the rotation axis. As for the orientation vector the frame in which the axis' elements are defined is not relevant. The relation between two orientation vectors which was depicted in equation (5-18) and (5-19) results in the following relation between two quaternions:

$$\mathbf{q}_2 = -\mathbf{q}_1 \quad \textbf{(5-20)}$$

This means that the transformation of one frame into another can be realized by a right-sided rotation as well as a left-sided rotation.

Since the use of quaternions is not as intuitive as the one of the orientation vector or the Euler attitude, it is common to convert quaternions to a coordinate transformation matrix and back to transform coordinate frames. The transformation matrix of a quaternion is defined by:

$$C_{\beta}^{\alpha} = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2-b^2+c^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2-b^2-c^2+d^2 \end{pmatrix} \quad \textbf{(5-21)}$$

The reverse conversion is defined by:

$$\zeta = \frac{1}{2}\sqrt{1+c_{11}+c_{22}+c_{33}} \tag{5-22}$$

$$a = \zeta$$

$$b = \frac{1}{4\zeta}(c_{32}-c_{23})$$

$$c = \frac{1}{4\zeta}(c_{13}-c_{31})$$

$$d = \frac{1}{4\zeta}(c_{21}-c_{12})$$

ζ has to be a real number and a division by zero should not be performed.

5.1.3 North Pole

The term North Pole suggests that this is the most Northern location on Earth, in reality there are three different types of North poles which have different positions. First there is the geographic North Pole. It is defined by the intersection of the Earth's rotation axis with the Earth's surface. Its antipode is the geographic South Pole. In contrast to the other definitions of the poles it has a static position of 90° 0' N [100].

The artic magnetic North Pole is the point of the Northern hemisphere where the magnetic field lines of the Earth's magnetic field are vertical to the Earth's surface towards the Earth's center of mass. The arctic North Pole's position is not fixed and changes based on certain patterns. Due to distortions in the Earth's magnetic field the position also has short-term variations. The last measurement in 2005 estimated the position of the arctic North Pole at 82.7°N 114.4°W [100].

Last of the different poles is the geomagnetic North Pole. This is a theoretic Pole assuming that there is a bar magnet at the Earth's center. In 2010 this Pole was at the position 80.1°N 72.13° W. In contrast to the Arctic North Pole the Geomagnetic North Pole is defined by computations while the former is estimated by measurements. But also the Geomagnetic North Pole's position is not fixed it follows similar patterns like the Artic North Pole [101].

Relevant for navigation are only the geomagnetic North Pole and the geographic North Pole. For more information on the geomagnetic North Pole refer to [101].

5.2 Object oriented programming

In World War II the first computers were used to compute trajectories of projectiles. For these first "programs" the developer had to write instructions directly in machine code which meant to write a long line of binary code (zeros and ones). With the introduction of assemblers and assembler code it was possible to map machine instructions on code which was better human-readable. These instructions are called Mnemonics and use commands like *ADD* or *MOV* for computation.

Later higher programming languages providing a larger level of abstraction like BASIC or COBOL were developed. Software could now be written in a human-readable way and is then interpreted or compiled into machine-readable instructions. An interpreter translates the written program directly into activity while a compiler creates a file with machine-readable code. Although the compiler has additionally to create this file and also to prepare the linking to other libraries the software is then able to be executed faster. Furthermore only the machine-readable file has to be exported to another computer and not the interpreter. The programming language Java uses both: a compiler and an interpreter, the so called Java Virtual Machine (JVM) to enable portability to each existing platform system (Windows, Linux, Solaris, and Mac OS).

However, also more abstract languages like BASIC or COBOL were not object-oriented but rather first procedural and later structured programming languages. Although structural programming was widely used until the 1980s and is to some extend still used today object-oriented (OO) programming languages were developed. Structural programming languages are based on a divide-and-conquer strategy: divide a complex problem into a set of several sub-tasks and process these sub-tasks one after another. The problem is that with such a strategy the re-use of certain procedures (like computing the average) which might occur in the same manner in different parts of a program is not possible. The idea of object-oriented programming is therefore based on the modularity and re-usability of certain parts of the program.

Software which is implemented based on an OO-programming language provides modularity not only in form of classes and objects but also by the use of one or more packages. One package consists of several classes and interfaces thematically belonging together. A program might for example comprise one package containing classes for the graphical user interface (GUI) and another package containing classes for background computations which results are then depicted by the GUI. Classes are blueprints for actual objects and provide a template for the state of an object in form of variables and the behavior of an object in form of methods or functions. Methods are furthermore used to change the state of an object. Due to this type of modularity not only the re-usability of code is provided but also the internal implementation and state of an object is encapsulated. Only methods which are necessary to re-use this part of software are accessible by the software developer.

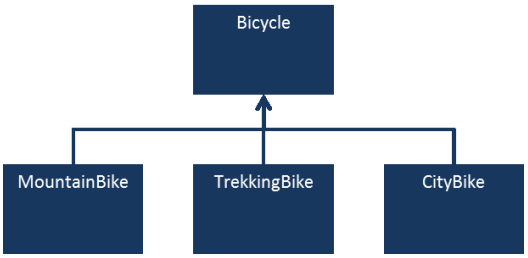


Figure 5-5: Example for inheritance in OO-programming languages. The sub-classes are a specialization of the super-class *Bike* extending its functionality by own methods and variables.

Additionally OO-programming languages normally enable inheritance between classes. An inheriting class has access to all methods and variables of its super-class as long as this access is not restricted on purpose and is able to extend this functionality with more methods and variables. An inheriting class is

therefore often denoted as a specialization of its super-class see Figure 5-5. In contrast to interfaces or abstract classes an actual object can be initialized from a super-class. Interfaces or abstract classes are pure templates for a certain type of object but without providing real functionality since their methods are empty. The functionality is then defined by inheriting sub-classes. The interface *Animal* as depicted in Figure 5-6 provides one empty method for a behavior common to all animals: *eat*. Since the food and the way it is eaten are different for each animal (a bird would pick food while a mammal chows and a reptile uses its tongue) it would make no sense to have an actual implementation of the method *eat* in the super-class *Animal*.

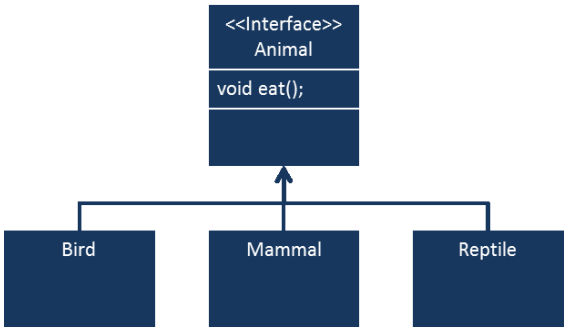


Figure 5-6: An example for inheriting from an interface or abstract class.

The state of an object is defined by different variables. For the example for the bicycle class in Figure 5-5 adequate variables would be *gear*, *size*, *speed*, and *break*. While *gear*, *size*, and *speed* are variables which most likely will be denoted in primitive data types like *integer* or *double* the variable *break* could also be another object of super-class *Break* which may have sub-classes like *DiskBrake* or *HubBrake*.

To change the state of variables methods implemented for this type of object are used. Useful methods for the bicycle example would be:

double decreaseSpeed(double deltaSpeed)

double increaseSpeed(double deltaSpeed)

The primitive data type in the parentheses indicates the amount the current speed is reduced or increased and the *double* before the name of the method denotes the return type containing the newly adopted speed. It is also possible to pass objects as parameters and to use objects as return type.

One method each OO-program needs is the *main*-method. The *main*-method is the entry point from where a program is started and normally is not integrated into a certain class but in its own file. In the body of this method those objects are initialized which are needed to start the functionality of the software. This is the same for all major OO-programming languages not matter if Java, C#, or C++ is used. An exception is Android as described in 3.4.1. Here instead of a *main*-method an *Activity* is used to start the program, nevertheless the concept is the same since it is the *onCreate*-method of the start *Activity* which marks the entry point to the application.

The information introduced here is a short summary of OO-programming and provides only the basic concepts of this type of software implementation. This summary is based on [102] and [103].

5.3 Abbreviations

AFLT	Advanced Forward Link Trilateration
AGNSS	Assisted-GNSS
AGPS	Assisted GPS
AOA	Angle of Arrival
AOSK	Android Open Source Project
API	Application Programming Interface
AU	Application Unit
Bluetooth SIG	Bluetooth Special Interest Group
BPSK	Binary-Phase Shift Keyed
BSS	Basic Service Set
C2C	Car-to-Car
C2C-CC	Car-to-Car-Communication Consortium
C/A code	Coarse Acquisition code
CDMA	Code Division Multiple Access
CIRS	Inertial Reference System
COM	Center of Mass
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTP	Conventional Terrestrial Pole
CZM	Conventional Zero Meridian
DGPS	Differential GPS
DOD	U.S. Department of Defense
DOT	U.S. Department of Transportation
DR	Dead Reckoning
DVM	Dalvik Virtual Machine
EC	European Community
ECEF	Earth-Centered Earth-Fixed
EKF	Extended Kalman Filter
EOTD	Enhanced Observed Time Difference
ESA	European Space Agency
ESS	Extended Service Set
FCC	U.S. Federal Communications Commission
FDMA	Frequency Division Multiple Access
FFD	Full-Function Device
FH	Frequency Hopping
FH-CDMA	Frequency Hopping – Code Division Multiple Access
FHSS	Frequency Hopping Spread Spectrum
FOC	Full Operability Constellation
GFSK	Gaussian frequency shift keying
GIOVE	Galileo In-orbit Validation Element
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile Communication
GUI	Graphical user interface
HOW	Handover Word
HS-receiver	High Sensitivity receiver
IBSS	Independent Basic Service Set
IC	Integrated Circuit
ICD	Interface Control Document
IDE	Integrated Development Environment
IERS	International Earth Rotation and Reference System Service

IF	Intermediate Frequency
IGRF	Geomagnetic Reference Field
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IOV	In-orbit Validation
IRM	IERS reference meridian
IRP	International reference pole
ISM frequency band	Industrial, Scientific, and Medical frequency band
JVM	Java Virtual Machine
KF	Kalman Filter
LA	Location Area
LAI	Location Area Identifier
LBS	Location Based Services
LI	Light indoor
LLC	Logical Link Control
LOS	Line of Sight
MAC	Medium Access Layer
MANET	Mobile Ad-Hoc Network
MCS	Monitor Control Station
MEMS	Micro Electro-Mechanical System
MEO	Medium Earth Orbit
MS	Mobile Station / Monitor Station
NASA	National Aeronautics and Space Administration
NLOS	Non-Line of Sight
NMEA	National Marine Electronics Association
OBU	On-Board Unit
OO	Object-Oriented
OS	Open Sky / Operating system
O-QPSK	Offset-Quadrature Phase Shift Keying
P2P	Peer-to-Peer
P2PKF	Peer-to-Peer Kalman filter
PAN	Personal Area Network
PANC	Personal Area Network Coordinator
PDA	Personal Digital Assistant
PDR	Pedestrian Dead Reckoning
PHY	Physical Layer
PREP	Position Reply
PREQ	Position Request
PRN	Pseudo Random Noise
PSD	Power Spectral Density
RF	Radio Frequency
RFD	Reduced-function device
RFID	Radio Frequency Identification
RSS	Radio Signal Strength
RTT	Round Trip Time
SA	Selective Availability
SDMA	Spatial Division Multiple Access
SDK	System Development Kit
SiP	System-in-Package
SNR	Signal to Noise Ratio
SV	Space Vehicle
TA	Timing Advance
TDD	Time Division Duplex

TDMA	Time Division Multiple Access
TDOA	Time Difference of Arrival
TLM	Telemetry Word
TOA	Time of Arrival
TOW	Time of the Week
TTFF	Time To First Fix
UKF	Unscented Kalman Filter
UMTS	Universal Mobile Telecommunications System
VCO	Voltage Controlled Oscillator
WGS84	World Geodetic System 1984
WLAN	Wireless Local Area Network
WMM	U.S. / U.K. World Magnetic Model
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Networks

5.4 References

[1] Chip.de, July 2001. [Online]. Available: http://www.chip.de/news/Erstes-Handy-mit-GPS-im-Handel_30248961.html.

[2] F. van Diggelen, A-GPS - Assisted GPS, GNSS, and SBAS, 1 ed., Boston: Artech House, 2009.

[3] Margaria D., Presti L. Lo, Kassabian N.,Samson J., "A New Peer-to-Peer Aided Acquisition Approach Exploiting C/No Aiding," in *5th ESA Workshop on Satellite Navigation Technologies – NAVITEC' 2010*, Noordwijk, The Netherlands, 2010.

[4] Federal Communications Commission (FCC), "Notice of Proposed Rulemaking, Third Report and Order, and Second further Notice of Propose Rulemaking," Federal Communications Commission (FCC), Washington, D.C., 2011.

[5] P. D. Groves, GNSS, Inertial, And Multisensor Integrated Navigation Systems, Boston: Artech House, 2008.

[6] A. Köhne und M. Wößner, „NAVSTAR GPS - Geschichtliches," 30 09 2008. [Online]. Available: http://www.kowoma.de/gps/Geschichte.htm. [Zugriff am 5 12 2011].

[7] E. D. Kaplan and C. J. Hegarty, Understanding GPS - Principles and Applications, Norwood: Artech Haouse, Inc., 2006.

[8] NovAtel Inc., An Introduction to GNSS, Calgary: NovAtel Inc., 2010.

[9] U.S. Department of Homeland Security, "Navigation Center," 18 1 2012. [Online]. Available: http://www.navcen.uscg.gov/?Do=constellationStatus. [Accessed 18 1 2012].

[10] A. Köhne und M. Wößner, 06 2012. [Online]. Available: http://www.kowoma.de/gps/.

[11] W. Jones, "Chip-Scale Atomic Clock," IEEE Spectrum, 2011.

[12] Hewlett Packard, "Fundamentals of Quartz Oscillators," 1997.

[13] GPS Joint Program Office, and ARINC Research Corporation, *Navstar GPS space segment/Navigation User Interfaces,* 2003.

[14] B. Eissfeller, A. Teuber und P. Zucker, „Indoor-GPS: Ist der Satellitenempfang in Gebäuden möglich?," *Zeitschrift für Geodäsie, Geoinformation und Landmanagement,* Nr. 4, 2005.

[15] G. Lachapelle, "GNSS Indoor Location Technologies," *Journal of Global Positioning Systems,* vol. 3, no. 1-2, 2004.

[16] European Commission, "112-The European emergency number," [Online]. Available: http://ec.europa.eu/information_society/activities/112/rules/index_en.htm. [Accessed 03 02 2012].

[17] J. Schiller, Mobilkommunikation, Pearson Studium, 2003.

[18] Wikipedia Deutschland, „LTE-Advanced,“ [Online]. Available: http://de.wikipedia.org/wiki/LTE-Advanced. [Zugriff am 27 11 2012].

[19] D. Munoz, F. Bouchereau, C. Vargas and E. R., Position Location Techniques and Application, Burlington: Academic Press, 2009.

[20] IEEE Computer Society, "Telecommunications and Information Exchange between systems - Local and Metropolitan area Networks - Specific Requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," New York, 2010.

[21] V. Honkavirta, T. Perlälä, S. Ali-Löytty and R. Piché, "A Comperative Survey of WLAN Location Fingerprinting Methods," Hannover, 2009.

[22] J. Wedel, Integrierte Navigationssysteme, Munich: Oldenbourg Wissenschaftsverlag, 2007.

[23] B. Vigna, "Future of MEMS: An industry point of view," in *International Conference on Thermal Mechanical and Multiphysics Simulation and Experiments in Micro-Electronics and Micro-Systems, EuroSimE*, 2006.

[24] C. Acar, *Robust Micromachined Vibratory Gyroscopes,* Berkeley, 2004.

[25] O. L. G. Mezentsev, "Pedestrian Dead Reckoning - A Solution To Navigation In GPS Signal Degraded Areas?," vol. 59, no. 2, pp. 175-182, 2005.

[26] R. Stirling, K. Fyfe and G. Lachappelle, "Evaluation of a New Method of Heading Estimation for Pedestrian Dead Reckoning Using Shoe Mounted Sensors," 2005.

[27] u-Blox, "u-blox' Dead Reckoning for Automotive Applications," Thalwil, 2010.

[28] T. Brand and R. Phillips, "Foot-to-Foot Range Measurement as an Aid to Personal Navigation," Alburquerque, 2003.

[29] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka and J. Gutiérrez Boronat, "Comparision and Evaluation of Acceleration Based Step Lenght Estimators for Handheld Devices," Zürich, 2010.

[30] D. Gusenbauer, C. Isert and J. Krösche, "Self-Contained Indoor Positioning on Off-the-Shelf Mobile Devices," Zürich, 2010.

[31] Z. Sun, X. Mao, W. Tian and X. Zhang, "Activity classification and dead reckoning for pedestrian navigation with wearable sensors," vol. 20, no. 1, 2008.

[32] J. Collin, *Investigation of Self-Contained Sensors for Personal Navigation,* Tampere: Tampere University of Technology, 2006.

[33] M. Afzal, V. Renaudin and G. Lachapelle, "Magnetic Field based Heading Estimation for Pedestrian Navigation Environments," Guimarares, 2011.

[34] Apple Nike, *Nike + iPod User Guide,* 2010.

[35] Apple, "Apple Support," 19 08 2010. [Online]. Available: http://support.apple.com/kb/HT2293?viewlocale=de_DE#faq1. [Accessed 20 11 2011].

[36] Nintendo, "Nintendo 3DS," Nintendo, 2010. [Online]. Available: http://www.nintendo.de/NOE/de_DE/nintendo_3ds_23802.html. [Accessed 21 11 2011].

[37] Sony, "PlayStation Vita," Sony, 2011. [Online]. [Accessed 21 11 2011].

[38] Wikipedia Deutschland, "Wikipedia - Siemens Mobile," 08 11 2011. [Online]. Available: http://de.wikipedia.org/wiki/Siemens_Mobile. [Accessed 21 11 2011].

[39] Wikipedia Deutschland, "Wikipedia - Nokia," 17 11 2011. [Online]. Available: http://de.wikipedia.org/wiki/Nokia. [Accessed 21 11 2011].

[40] Wikipedia Deutschland, "Wikipedia - Microsoft Windows Mobile," 20 11 2011. [Online]. Available: http://de.wikipedia.org/wiki/Microsoft_Windows_Mobile. [Accessed 21 11 2011].

[41] Wikipedia Deutschland, "Wikipedia - Apple iPhone," 21 11 2011. [Online]. Available: http://de.wikipedia.org/wiki/IPhone. [Accessed 21 11 2011].

[42] Wikipedia, "iOS," 26 11 2012. [Online]. Available: http://en.wikipedia.org/wiki/IOS. [Accessed 27 11 2012].

[43] Apple, "iOS App Propgrammin Guide," 12 10 2011. [Online]. Available: http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AppDesignBasics/AppDesignBasics.html#//apple_ref/doc/uid/TP40007072-CH2-SW1. [Accessed 21 11 2011].

[44] Wikipedia Deutschland, "Wikipedia - Android (Betriebssystem)," 21 11 2011. [Online]. Available: http://de.wikipedia.org/wiki/Android_%28Betriebssystem%29. [Accessed 21 11 2011].

[45] Wikipedia Deutschland, „Google Play,“ 24 11 2012. [Online]. Available: http://de.wikipedia.org/wiki/Google_Play. [Zugriff am 27 11 2012].

[46] Google, "Android Developers," 16 04 2007. [Online]. Available: http://developer.android.com/index.html. [Accessed 21 11 2011].

[47] P. Dykta, *Position Determination of Jamming Devices by GPS and Pedestrian Navigation,* 2012.

[48] H.-J. Jang, J. Kim and D.-H. Hwang, ""Robust step detection method for pedestrian navigation systems"," vol. 43, no. 14, 2007.

[49] D. Alvarez, R. González, A. López and J. Alvarez, "Comparison of Step Length Estimators from Wearable Accelerometer Devices," in *Encyclopedia of Healthcare Information Systems*, IGI Global, 2008.

[50] Lo Presti L., Margaria D., Samson J. , "A novel peer to peer aided acquisition strategy tailored to Galileo E1 receivers," in *52nd International Symposium ELMAR-2010*, Zadar, Croatia, 2010.

[51] R. Garello, J. Samson, M. Spirito and H. Wymeersch, "Peer-to-Peer Cooperative Positioning Part II: Hybrid Devices with GNSS & Terrestrial Ranging Capability," *InsideGnss,* no. July/August 2012, pp. 56-64, 2012.

[52] R. Garello, L. Lo Presti, G. Corazza and J. Samson, "Peer-to-Peer Cooperative Positioning Part I: GNSS-Aided Acquisition," *InsideGnss,* no. March/April2012, pp. 55-63, 2012.

[53] Car2Car-Communication Consortium, "Car2Car-Communication Consortium Manifesto - Overview of the C2C-CC System," 2007.

[54] IEEE Computer Society, "IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area networks - Specific Requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," New York, 2007.

[55] IEEE Vehicular Technology Society, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Resource Manager," New York, 2006.

[56] IEEE Vehicular Technology Society, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operation," New York, 2006.

[57] Intelligent Transportation Systems Comitee, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages," New York, 2006.

[58] Intelligent Transportation Systems Committee, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services," New York, 2007.

[59] A. Benslimane, "Localization in Vehicular Ad-hoc Networks," in *Proceedings of the 2005 Systems Communications (ICW'05)*, Montreal, 2005.

[60] N. Alam, A. Tabatabaei Balaei and A. Dempster, "Positioning Enhancement with Double Differencing and DSRC," in *Proceedings of the International Meeting of the Institute of Navigation (ION GNSS 2010)*, Portland, 2010.

[61] H. Wymeersch, J. Lien and M. Win, "Cooperative localization in wireless networks",," *Proceedings of the IEEE,,* vol. 97, no. 2, 2009.

[62] D. Park, J. Kang and E. Kim, "Ad hoc Peer-to-peer Tracking using Relative Location Estimation," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zürich, Swiss, 2010.

[63] Lo Presti L., Margaria D., Rao M., "Novel Techniques for a Cooperative Positioning Approach Based on Peer-to-Peer Networks," in *Data Flow from Space to Earth Applications and Interoperability International Conference*, Venice, Italiy, 2011.

[64] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering,* vol. 82, pp. 35-45, 1960.

[65] P. S. Maybeck, Stochastic Models, Estimation, and Control, New York: Academic Press, 1979.

[66] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Chapel Hill, 2001.

[67] S. Haykin, Kalman Filtering and Neural Networks, John Wiley & Sons, Inc., 2001.

[68] Bluetooth SIG, "Bluetooth History," 2011. [Online]. Available: http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx. [Accessed 27 02 2012].

[69] IEEE Computer Society, *IEEE 802.15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs),* New York: IEEE Standards, 2005.

[70] Bluetooth SIG, "Specification of the Bluetooth System," 2010.

[71] Bluetooth SIG, "Bluetooth Core Specification Addendum 2," 2011.

[72] Wikipedia Deutschland, „Wikipedia Bluetooth," 23 02 2012. [Online]. Available: http://de.wikipedia.org/wiki/Bluetooth. [Zugriff am 27 02 2012].

[73] Wikipedia, "Wikipedia Bluetooth," 26 02 2012. [Online]. Available: http://en.wikipedia.org/wiki/Bluetooth. [Accessed 27 02 2012].

[74] ZigBee Alliance, "ZigBee Alliance - The Alliance," 2012. [Online]. Available: http://www.zigbee.org/About/AboutAlliance/TheAlliance.aspx. [Accessed 29 02 2012].

[75] IEEE Computer Society, *802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs),* New York, 2003.

[76] ZigBee Alliance, *ZigBee Specification,* San Ramon, 2008.

[77] Y. Xiao and Y. Pan, Emerging Wireless LANs, Wireless PANs, and Wireless MANs, New Jersey: John Wiley & Sons, 2009.

[78] IEEE Computer Society, *IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks- Specific requirements,* New York: LAN/MAN Standards Committee, 2007.

[79] Wikipedia Deutschland, „Wikipedia - Wi-Fi,“ 27 01 2012. [Online]. Available: http://de.wikipedia.org/wiki/Wi-Fi_Alliance. [Zugriff am 01 03 2012].

[80] Wi-Fi Alliance, *Wi-Fi CERTIFIED Wi-Fi Direct,* 2010.

[81] I. Kraemer, I. Bartunkova and B. Eissfeller, "Evaluation of a Peer-to-Peer Kalman Filter in Weak-Signal Areas using a Software GNSS-Signal-Simulator," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zürich, 2010.

[82] Y. Cao, "Matlab Central File Exchange," 2008. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/18189. [Accessed January 2011].

[83] Y. Cao, "Matlab Central File Exchange," 2010. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/18217. [Accessed January 2011].

[84] Wikipedia, "Android (operating system)," 25 06 2012. [Online]. Available: http://en.wikipedia.org/wiki/Android_%28operating_system%29. [Accessed 26 06 2012].

[85] Android, "Android open source project," [Online]. Available: http://source.android.com/about/index.html. [Accessed 26 06 2012].

[86] Wikipedia, "Android version history," 25 06 2012. [Online]. Available: http://en.wikipedia.org/wiki/Android_version_history. [Accessed 26 06 2012].

[87] Android Inc., "Platform Versions," [Online]. Available: http://developer.android.com/about/dashboards/index.html. [Accessed 26 06 2012].

[88] Android Inc., "Android Developers," [Online]. Available: http://developer.android.com/index.html. [Accessed 27 06 2012].

[89] A. Becker und D. Müller, Android 2, Heidelberg: dpunkt verlag, 2010.

[90] U. N. Nicola, „Einblick in die Dalvik Virtual Machine,“ IMVS Fokus Report, 2009.

[91] Android Inc., "Application Fundamentals," 22 06 2012. [Online]. Available: http://developer.android.com/guide/components/fundamentals.html. [Accessed 27 06 2012].

[92] NIST, "JAMA: A Java Matrix Package," 13 07 2005. [Online]. Available: http://math.nist.gov/javanumerics/jama/. [Accessed 29 06 2012].

[93] T. Reuter, *Implementation of a peer-to-peer Kalman Filter for mobile Android Devices,* Munich: Universität der Bundeswehr, 2012.

[94] Android Inc., "Developers Reference - SensorManager," 30 06 2012. [Online]. Available: http://developer.android.com/reference/android/hardware/SensorManager.html. [Accessed 2 07 2012].

[95] J. Mehaffey, J. Yeazel, S. Penrod and A. Deiss, "NMEA data," [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm. [Accessed 02 07 2012].

[96] J. Angwin and J. Valentino-Devries, "Apple, Google Collect User Data," The Wall Street Journal, 22 04 2011. [Online]. Available: http://online.wsj.com/article/SB10001424052748703983704576277101723453610.html. [Accessed 16 10 2012].

[97] S. Saeedi, Z. Syed and N. El-Sheimy, "A Comparison of Feature Extraction and Selection Techniques for Activity Recognition using Low-cost Sensors on a Smartphone," in *Proceedings of the ION GNSS 2012*, Nashville, 2012.

[98] B. Shin, J. Lee, C. Kim, S. Lee, Y. Byun, D. Yun and T. Lee, "Motion-Awareness 3D PDR System in GPS-Denied Environment using Smartphone," in *Proceedings of the ION GNSS 2012*, Nashville, 2012.

[99] J. Bortz, "A new mathematical formulation for strapdown inertial navigation," in *IEEE Transactions on Aerospace and Electronic Systems*, 1971.

[100] Wikipedia Deutschland, „Nordpol,“ 01 12 2012. [Online]. Available: http://de.wikipedia.org/wiki/Nordpol. [Zugriff am 03 12 2012].

[101] National Oceanic And Athmosphere Administration (NOAA), "Geomagnetism," [Online]. Available: http://www.ngdc.noaa.gov/geomag/geomag.shtml. [Accessed 03 12 2012].

[102] J. Liberty, „Erste Schritte,“ in *C++ in 21 Tagen*, Munich, Markt+Technik Verlag, 2000, pp. 24-30.

[103] Oracle, "The Java Tutorials," [Online]. Available: http://docs.oracle.com/javase/tutorial/java/concepts/index.html. [Accessed 14 11 2012].