Master's Thesis

Locating lost items using pedestrian dead reckoning on smartphones.

Martin Daum, BSc

Knowledge Technologies Institute,
Graz University of Technology

Supervisor: Ass.-Prof. Dr. Viktoria Pammer-Schindler

Graz, February 2015

This page intentionally left blank

Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst)

Lokalisierung von verlorenen Gegenständen durch Dead Reckoning von Fußgängern auf Smartphones.

Martin Daum, BSc

Institut für Wissensmanagement,
Technische Universität Graz

Betreuer: Ass.-Prof. Dr. Viktoria Pammer-Schindler

Graz, Februar 2015

This page intentionally left blank

Abstract

Many people face the problem of misplaced personal items in their daily routine, especially when they are in a hurry, and often waste a lot of time searching these items. There are different gadgets and applications available on the market, which are trying to help people find lost items. Most often, help is given by creating an infrastructure that can locate lost items.

This thesis presents a novel approach for finding lost items, namely by helping people re-trace their movements throughout the day. Movements are logged by indoor localization based on mobile phone sensing. An external infrastructure is not needed.

The application is based on a step based pedestrian dead reckoning system, which is developed to collect real-time localization data. This data is used to draw a live visualization of the whole trace the user has covered, from where the user can retrieve the position of the lost personal items, after they were tagged using simple speech commands.

The results from the field experiment, that was performed with twelve participants of different age and gender, showed that the application could successfully visualize the covered route of the pedestrians and reveal the position of the placed items.

Keywords

indoor localization, smartphone, dead reckoning, lost items tracker, step detection

ÖSTAT classification

1105 30%, 1108 40%, 1109 30%

ACM classification

Ubiquitous and mobile computing > Ubiquitous and mobile computing systems and tools

This page intentionally left blank

Kurzfassung

Tagtäglich sind viele Personen mit dem Problem konfrontiert, persönliche Gegenstände verloren zu haben. Diese Situationen treten meistens unter Zeitdruck auf und verursachen Stress. Bis die Gegenstände wieder gefunden werden können, wird oft viel Zeit verschwendet. Es wurden bereits verschiedene Geräte und Applikationen vorgestellt, die in solchen Situationen helfen sollen. Die meisten dieser Applikationen verwenden Technologien wie Bluetooth Marker oder NFC Sticker, die eine, oftmals teure, externe Infrastruktur benötigen.

Diese Arbeit präsentiert einen neuartigen Ansatz für eine Applikation, die das Auffinden von verlorenen Gegenständen durch die Zuhilfenahme von verschiedenen Lokalisierungstechniken innerhalb von Gebäuden ermöglicht. Es werden dabei nur Daten von Sensoren verwendet, die in handelsüblichen Smartphones verbaut sind. Eine externe Infrastruktur wird nicht benötigt.

Die Applikation baut auf ein Dead Reckoning System, das auf Schritterkennung im menschlichen Gangbild beruht. Die Lokalisierungsdaten, die in Echtzeit berechnet werden, werden dazu verwendet um den zurückgelegten Weg einer Person zu visualisieren. Mit Hilfe dieser Visualisierung ist es Personen möglich Gegenstände aufzufinden, deren Position während dem Gehen markiert wurde.

Die Ergebnisse aus dem Feldversuch, der mit Hilfe von zwölf Testpersonen unterschiedlichen Alters und Geschlechts durchgeführt wurde, zeigten dass die Applikation den zurückgelegten Weg und die Position der abgelegten Gegenstände erfolgreich darstellen konnte, was den Testpersonen das Auffinden ihrer Gegenstände ermöglichte.

Schlüsselwörter

lokalisierung, smartphone, dead reckoning, gegenstände finden, schritterkennung

ÖSTAT Klassifikation

1105 30%, 1108 40%, 1109 30%

ACM Klassifikation

Ubiquitous and mobile computing > Ubiquitous and mobile computing systems and tools

This page intentionally left blank

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, 15.02.2015

Martin Daum

This page intentionally left blank

Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported me throughout the entire process of writing this thesis. First and foremost I want to thank my girlfriend Juliane for never giving up on me and for pushing and motivating me to finish my education. Furthermore I am very thankful for my family and my close friends supporting me my whole life and providing me an environment that enables me to work on this thesis.

This work would not have been possible without the involvement of some people. I share the credit of my work with my supervisors Viktoria and Jörg, who supported me with intense knowledge and insights and always had open ears for arising questions. I would also like to thank Alexander for helping me to understand the complex topic of signal processing. I feel deeply indebted to my fellow colleague Karl Heinz for revising the written thesis and supporting me with discussions and decisions through the development of the prototype application. Last but not least I want to thank my family, especially Anna, Arthur, Bernd, Brigitte, Elfi, Philipp, Otto, Thomas, Wolfgang, my mother and my grandmother, for helping me to evaluate my work by participating in the field experiment.

Martin Daum

Graz, February 2015

This page intentionally left blank

Table of Contents

1 Introduction **1**

2 Theoretical Background **3**

2.1 Navigation vs. localization 3

2.2 Mobile devices 3

2.3 Localization technologies 4

2.3.1 GPS 4

2.3.2 GSM 4

2.3.3 WiFi 5

2.3.4 Bluetooth 6

2.3.5 NFC 6

2.3.6 Camera 7

2.3.7 Microphone 7

2.3.8 Motion sensors 7

2.4 Localization methods 10

2.4.1 Absolute vs. relative localization 10

2.4.2 Inertial localization 10

2.4.3 Satellite localization 11

2.4.4 Radio localization 11

2.4.5 Optical localization 11

2.5 Mobile development platforms 11

2.5.1 iOS 12

2.5.2 Android 13

3 Related Work **15**

3.1 Relevant indoor localization methods 15

3.2 Inertial localization methods 16

3.2.1 Dead reckoning 16

3.2.2 Step detection methods 17

3.2.3 Direction detection 23

3.2.4 Step length estimation 25

3.2.5 Retrieving the initial location 27

3.3 Optical localization methods 28

3.4 Trace visualization 30

3.5 Apps for locating lost items 31

4 Prototype Application **35**

4.1 Concept 35

4.1.1 Requirements 35

4.1.2 Use cases 36

4.2 System design 36

4.3 Technical Requirements 37

4.4 Development environment 38

4.4.1 Hardware 38

4.4.2 XCode 38

4.4.3 CocoaPods 38

4.4.4 Git 39

4.5 Implementation 39

4.5.1 Reading sensor data 40

4.5.2 Client-server communication 44

4.5.3 Simulator 46

4.5.4 Dead reckoning 46

4.5.5 Placing items . . . 52
4.5.6 Visualization . . . 53

5 Field Experiment **61**
5.1 Goals . . . 61
5.2 Experiment setup . . . 62
5.2.1 Hardware setup . . . 62
5.2.2 Test track . . . 62
5.3 Test participants . . . 63
5.4 Walking test . . . 63
5.4.1 Prerequisites . . . 64
5.4.2 Procedure . . . 64
5.5 Interview . . . 65

6 Results **67**
6.1 Reference data . . . 67
6.2 Step detection . . . 69
6.3 Step length estimation . . . 71
6.4 Direction detection . . . 73
6.5 Locating lost items . . . 76

7 Discussion and Lessons Learned **81**

8 Conclusions **85**

9 Outlook and Future Work **89**

List of Figures **93**

List of Tables **95**

References **97**

1. Introduction

Everyone knows the annoying situation when personal items of appreciated value have disappeared and several precious minutes, or even hours, are wasted for frantic searching. Following Murphy's Law telling us that *Anything that can possibly go wrong, does*, these situations always emerge when you are in a hurry and are thereby really uncomfortable.

There are different gadgets available, which are aimed to help in these situations. For example electronic key finders triggered by whistling are available on the market for years, but the acceptance for them was always rather low.
With the enormous growth of popularity of mobile end-consumer devices, like smartphones and tablets, a lot of new approaches were taken to solve the problem of locating lost items. A modern smartphone offers a lot of built-in hardware, like inertial sensors, microphones and cameras, that creative app developers can use for locating various items, including the phone itself.
GPS is the most popular sensor used for a wide range of location-aware applications, like navigation systems or sports tracking applications, and was mainly responsible for making the topic of localization systems one of the most popular and actively researched fields in software development. But GPS has one big disadvantage, because it needs direct visibility to a satellite, it only works outdoors and is rather inaccurate in urban areas.
To counteract this limitation of GPS, the concept of indoor localization system was born. While outdoor localization using GPS is rather easy to implement nowadays, indoor localization is a more complex topic and offers very different solutions. There are implementations that work with *WiFi*, *Bluetooth*, *Near Field Communication*, *Cameras*, *Microphones* and *inertial sensors*, which are using data gained from *ac-*

celerometers, *gyroscopes* and *magnetometers*.
The advantage of having all the different sensors is, that they can be combined to improve accuracy and minimize errors. So it is a common technique to combine GPS with other sensors to improve the localization accuracy in urban areas.
Most of the sensors mentioned above are dependent on an external infrastructure to work properly. So indoor localization using WiFi, Bluetooth or NFC is not possible without utilizing additional hardware.

This thesis presents a novel approach for retrieving lost personal items only by using sensors, that are built into every modern smartphone. The idea is to track the trace of users and giving them the possibility to tag the position of their personal items while they are using the application. With this information, it is possible to retrieve the position of the items in case they are lost.
The goal of this thesis is to verify, that such an application that solely uses sensors that are built into modern smartphones for the tracking, is able to help users to retrieve their lost items.

The first part of the thesis will establish a fundamental theoretical background in chapter 2, including localization technologies and methods, mobile development platforms and mobile devices. An overview of related work in the field of inertial and optical localization methods, trace visualization as well as state-of-the-art applications for locating lost items is given in chapter 3.
The second part will present how the findings of the first part were applied to design and implement a prototype application capable of locating personal items solely with sensors built into modern smartphones. A dead reckoning system is presented including step detection, step length estimation, direction detection and initial position retrieval in chapter 4. Chapter 5 will explain the methods that were used to evaluate the app in an experiment consisting of a walking test and an interview.
Chapter 6 will present all results, that were collected during the field experiment and the following analyzation phase. These results will be discussed and interpreted in detail in chapter 7.
Finally, the conclusion of this work is presented in chapter 8 before the work is closed by giving an outlook into possible future work in chapter 9.

2. Theoretical Background

In order to better understand specific requirements and complex details discussed in later chapters, this section provides definitions and delimitation and gives a brief introduction to mobile development platforms that are used to build consumer smartphone applications and hardware sensors that are eligible to be used in indoor localization systems.

2.1 Navigation vs. localization

The terms *navigation* and *localization* are often used synonymous in literature which is misleading and confusing. Bowditch (2002) defines navigation as "the process of planning, recording and controlling the movement of a craft or vehicle from one place to another". Localization refers to the process of finding a relative location depending on known reference points or base stations. Liu et al. (2007)
Localization is a single step in the more complex process of navigation. The focus of this thesis is on the task of localization.

2.2 Mobile devices

Modern mobile devices refer mostly to the terms *Smartphones* and *Tablets*. Smartphones are mobile phones built on operating systems with advanced computing capabilities and connectivity features. The first smartphones that appeared on the market combined a mobile phone with the functions of a personal digital assistant. Later models included various features like internet connection, cameras or GPS receivers.

2.3 Localization technologies

The availability of a wide range of sensors in modern smartphone devices enables many different approaches for implementing outdoor and indoor localization systems. The following section gives a brief overview of different sensors that are suitable for implementing localization with it.

2.3.1 GPS

The *Global Positioning System* (GPS) was initially invented for military purposes as a passive satellite system which is maintained by the department of defense of the United States of America (Zirari et al. (2010)). It was first used for civil purposes in 1983. The position of GPS is calculated between the receiver and the distance to four to ten satellites in view. GPS always requires a clear view between the receiver and the satellite. The distance is measured based on the time the coded signal needs to get from the satellites to the receiver.
According to Baranski and Strumillo (2012) the positioning accuracy in open area is about 2-3 meters. The positioning error depends on various factors, like atmospheric conditions, sun activity, type of terrain, geographical location. The accuracy decreases significantly in urban areas and the positioning error is often more than 100 meters. In urban areas, the signal often is bounced off walls because there is no clear line-of-sight, so the receiver miscalculates the position, because the signal needs more time to reach the receiver.
Beside the described disadvantages, Kothari et al. (2011) states, that GPS is the widest used technology to retrieve locations and directions. Reasons are that GPS is comparably easy to integrate, because it is well known and documented. Almost all mobile development platforms have implementations of GPS integrated, where utilizing other localization methods is often a very challenging task.

2.3.2 GSM

The *Global System for Mobile Communications (GSM)* is a standard used by mobile phone to communicate via phone calls or short messages. Thus GSM is integrated

in any mobile phone, which, according to Otsason et al. (2005), makes it the worldwide most widespread phone technology.
GSM uses *base stations* or *cell towers* to broadcast a signal to the phone devices. As stated in Bill et al. (2004), the maximum range a signal can be received differs from 250m in urban areas to 35km in open areas.
The GSM cellular network makes it possible to estimate the position of a specific device. According to Otsason et al. (2005), this is done by clearly identifying the base station the device is using at a specific time. Due to the fact that the maximum range of a base station is up to 35km, this technology is only suitable if only a low positioning accuracy is needed.

2.3.3 WiFi

Nowadays, more and more buildings become equipped with *wireless local network area* access points. Private households, offices, hotels, public buildings like hospitals or shopping centers are mostly covered with WiFi.
According to Au et al. (2012), indoor localization systems using WiFi are reliable, rather easy to implement and there is no need for extra hardware except the WiFi infrastructure. Because the primary use case of a WiFi infrastructure is to provide internet access, the required hardware is available.
Evennou and Marx (2006) describe different models to perform user localization with WiFi. Compared to GPS, WiFi based localization system are not able to calculate the position of the user based on the distance to the access point, because this kind of information is not available with this technology. Instead, there are different methods used to retrieve location like *Received Signal Strengh (RSS)*, *WiFi cell id* or *Fingerprinting*.
A big advantage of WiFi systems, according to Ahn and Yu (2009), is that these systems do not accumulate measurement errors. The main disadvantage is, that the technology is environmental-dependent, which means, that if an area is not covered by the environment, no localization is possible in that area.
WiFi based indoor localization systems are a very popular research topic and they are often combined with GPS and/or inertial systems to eradicate the mutual weaknesses.

2.3.4 Bluetooth

As stated in Anastasi et al. (2003), the Bluetooth wireless technology was designed to be a short-range connectivity solution for mobile devices. It allows to transfer data directly between devices. The basic concept of Bluetooth relies on *piconets*, a channel that is shared by all devices that can communicate with each other.
Bluetooth gained even more popularity with the invention of *Bluetooth Low Energy (BLE)* in 2007. As stated in Gomez et al. (2012), Bluetooth Low Energy is an energy-efficient solution for monitoring and controlling applications, which enabled the production of hardware like Bluetooth tags, which are locatable by other Bluetooth enabled devices in short ranges.
In 2013, Apple presented *iBeacon*[1], which is using BLE 4.0. iBeacons basically can establish a region around the transmitter, which is discoverable by a receiver in a range up to 70 meters. Solutions like region monitoring can be easily implemented with this technology. Like WiFi, it is not possible to measure the exact distance from the receiver to the transmitter, because iBeacons are using relative distances rather than absolute distances.

2.3.5 NFC

Ozdenizci et al. (2011) defines *Near Field Communication (NFC)* as an evolving short range, wireless communication technology based on *Radio Frequency Identification (RFID)*. Compared to Bluetooth, NFC operates within very short ranges, usually just a few centimetres.
NFC supports different operating modes like peer-to-peer, card emulation and reader/writer. That means, using NFC on a smartphone, one can communicate with other NFC enabled smartphones, card readers or passive RFID-tags. Compared to Bluetooth tags, RFID tags are rather inexpensive.
NFC support is integrated in many Android devices, but the technology received a big setback when Apple announced that they will not integrate NFC in their devices, but rather use the iBeacon technology based on Bluetooth.

[1] `https://developer.apple.com/ibeacon/`

2.3.6 Camera

Cameras are sensors used for most diverse kind of apps, like shooting photos or recording movies. The usage in the context of localization is rather limited. Mulloni et al. (2009) present a solution where simple recognizable markers, such as QR Codes, are scanned using the camera for deriving absolute positioning from the gathered information. More complex solutions as presented in Aubeck et al. (2011), Hol et al. (2006) and Bleser and Stricker (2009) are implemented in the context of augmented reality using real-time camera tracking together with inertial sensor fusion.

2.3.7 Microphone

Bihler et al. (2011) present a very interesting solution, where the microphone of a smartphone is used to receive ultrasonic signals sent by an emitter. These signals are the foundation of a smartphone museum guide system.

2.3.8 Motion sensors

Nearly all modern smartphones have motion sensors equipped. According to Aviv et al. (2012), they are responsible for measuring the orientation and movement, in terms of velocity and gravitational forces, of the device in space.
Rizqi et al. (2011) define inertial sensors as sensors, that are used for measuring data without external reference. If the features of the environment are considered, than it is a non-inertial sensor, like magnetometers.
Inertial measurement units combine accelerometers and gyroscopes, to deliver a combined three-dimensional measurement of specific force and angular rate.
Motion sensors are very popular and used in various applications in sports, navigation and especially games.

Accelerometer

As stated in Rizqi et al. (2011), accelerometers can measure acceleration in one, two or three orthogonal axes. When reading the sensor data, a data element is retrieved

that contains the acceleration on all three axes with the force of gravity considered. Aviv et al. (2012) state that accelerometers are sensitive to the difference of the local gravitation field and the sensor's linear acceleration, thus an accelerometer resting on a table will measure an acceleration of $g = 9.81 m/s^2$, where an accelerometer free falling due to the gravity of the earth will measure zero.
The linear movement of the accelerometer in respect to the axes of a smartphone is shown in Figure 2.1.

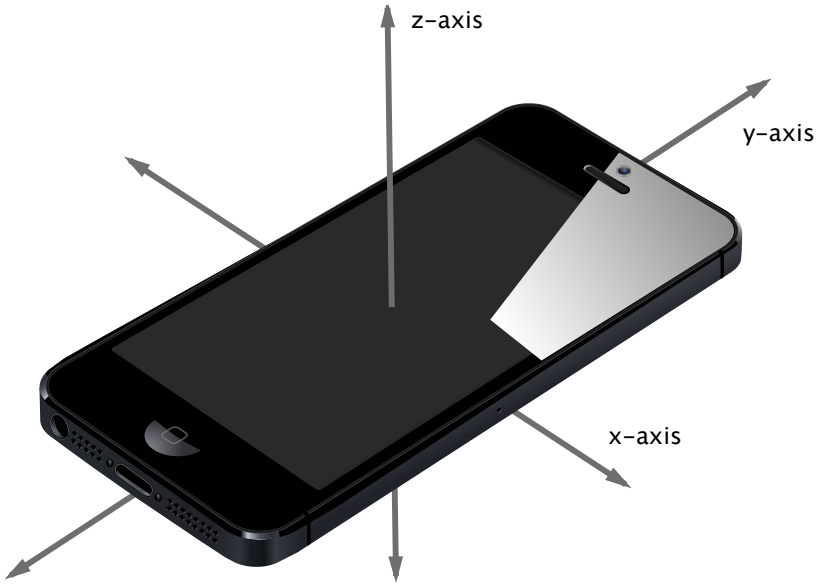

Figure 2.1: Linear three-axis accelerometer (adapted from Aviv et al. (2012))

Accelerometers are well suited for tasks like activity recognition, because when analyzing signals, common patterns are easily noticeable.

Gyroscope

Butikov (2006) defines a traditional mechanical gyroscope, as a body of rotation, which is based on the principle of angular momentum. It typically consists of a rapidly spinning flywheel, which is attached to an axis an can turn freely to assume every direction. The angular momentum of the wheel is responsible to resist against changes in the direction of the axis of rotation, which means the gyroscope is able to maintain its orientation regardless of the movement of its support.
For consumer electronic like smartphones, traditional gyroscopes are not suitable. As stated in Rizqi et al. (2011), mobiles devices integrate optical or *MEMS (micro*

electromechanical system) gyroscopes. MEMS gyroscopes are also known as *vibrating structure gyroscopes*. They are based on the physical principle, that a vibrating object continues vibrating in the same plane as its surrounding structure rotates. MEMS gyroscopes are cheaper and smaller than mechanical gyroscopes but deliver similar accuracy.
Gyroscopes are used in aircrafts and spaceships to measure the *Euler angles* of an object, which are also defined as *pitch*, *roll* and *yaw*, as shown in Figure 2.2 (applied to a smartphone).

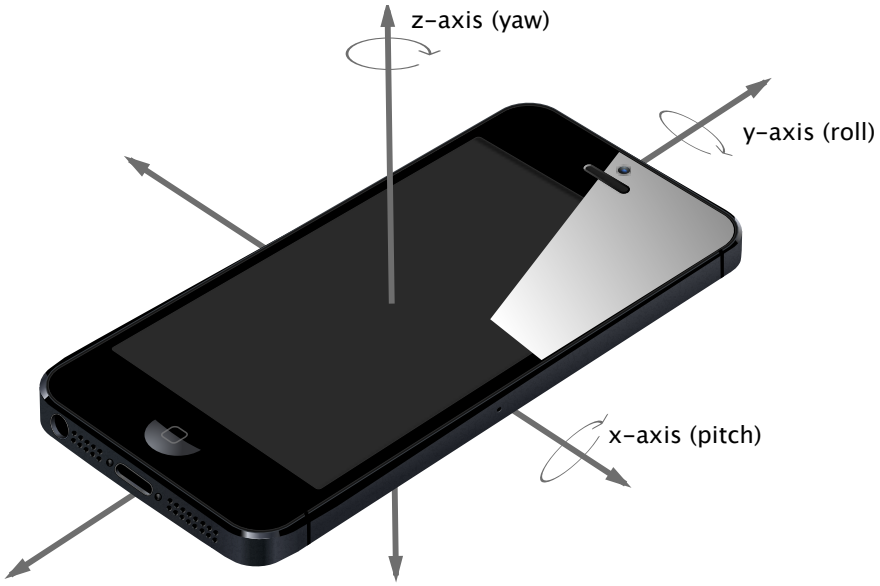

Figure 2.2: Gyroscope measurement (adapted from Rizqi et al. (2011))

Pitch, roll and yaw can be described as follows:

- Pitch is the rotation around the x-axis e.g. tilting the device forward and backwards.

- Roll is the rotation around the y-axis e.g. tilting the device from left to right or vice versa.

- Yaw is the rotation around the z-axis e.g. rotating the device around or opposite the clock.

Magnetometer

According to Rizqi et al. (2011), Magnetometers are non-inertial sensors because they are used to estimate heading in respect to the magnetic north.
The earth's magnetic field has been used for navigation since thousands of years. Renaudin et al. (2010) state that the first compass, invented by the Chinese, was a simple bowl of water and a magnetic element put on a leaf floating on the surface. Nowadays, magnetometers are small devices that are build into nearly every smartphone. Magnetometers are very vulnerable to magnetic interferences, which are typically created by electronic devices, but they work well in outdoor environments. If magnetometers are interfered through other magnetic fields, they have to be calibrated. Bowditch (2002) describes a procedure where the compass is rotated to multiple known headings. This method is also used for calibrating the compass found on any iPhone device.

2.4 Localization methods

This section maps the previously described localization technologies to different localization methods, which are described in Bowditch (2002) and Holčík (2012).

2.4.1 Absolute vs. relative localization

According to Baranski and Strumillo (2012), absolute localization is performed by technologies like GPS, which are able to calculate an exact coordinate for a device, whereas relative localization methods are calculating locations *relative* to an initial coordinate.

2.4.2 Inertial localization

Inertial Localization is a relative localization method, which uses inertial sensors like accelerometers and gyroscopes. *Relative localization* is often referred to as *dead reckoning*, which is a term from the marine navigation. As stated in Bowditch (2002), marine navigators used a known initial position for determining a new position while advancing in terms of motion and direction. As described in Jin and Soh (2011), the

readings of the inertial sensors are used to calculate a displacement to the previous position by double-integrating. The main disadvantage of this technique is that errors of noisy sensors are accumulated quickly because of the double integration.

2.4.3 Satellite localization

Satellite localization uses GPS or similar systems to determine locations. The advantages and disadvantages of these methods rely on the GPS system.

2.4.4 Radio localization

Radio localization methods are using different radio waves to measure the distance to a base station by comparing the received signal strength. Technologies like GSM, Bluetooth, WiFi and NFC use this method.

2.4.5 Optical localization

Optical localization uses known image patterns, which are mapped to locations so they are recognizable again. Cameras are used to perform optical localization methods.

2.5 Mobile development platforms

Today's portable devices such as smartphones and tablets have operating systems offering developers easy to use interfaces for creating mobile applications. According to Gandhewar and Sheikh (2010), the biggest competitors provide operating systems based on *UNIX* and *Windows*. Windows is licensed by Microsoft, whereas UNIX is open source. The most popular development platforms built on UNIX are *iOS by Apple Inc.* and *Android by Google Inc.*, which will be explained in more detail in the following subsections.
For each platform there are extensive and well documented Software Development Kits (SDK) available to encourage developers to build their own applications.

2.5.1 iOS

iOS[2] is the name of the operating system developed by *Apple Inc.* for their mobile devices *iPhone*, *iPad* and *iPod Touch*. The first version was released in 2007, when the first iPhone was introduced. It is based on the Darwin UNIX platform. Contrary to other competitors, iOS is only used on Apple's own hardware.
As stated in Apple Inc. (2010) *iOS Software Development Kit* contains all tools required to develop native iOS applications. The programming languages that are used with the XCode IDE on Mac OS X are *Objective-C* and *Swift*. iOS is structured in different layers, which is indicated in Figure 2.3.

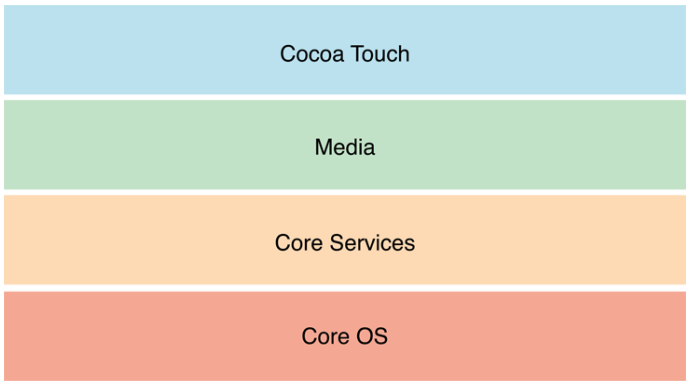

Figure 2.3: Layers of iOS (from Apple Inc. (2010))

Lower layers contain essential services and technologies, whereas higher layers build on lower layers and provide services and technologies that are easier to use and better documented. Developers are recommended to use high level frameworks whenever possible. There are different frameworks available to work with different technologies.
While Apple provides all necessary tools and interfaces for building native applications, Apple also helps developers distributing their apps using an open market called *App Store*. The App Store is maintained by Apple and each submitted application has to follow *Human Interface Guidelines* (Apple Inc. (2011)) in order to get into the store, otherwise the app gets rejected and has to be changed. The review process often takes a week or longer.
The disadvantage of the iOS platform is that the framework is not open source and

[2] `https://www.apple.com/at/ios/`

the operating system is restricted. For example it does not allow for real multitasking or gathering data from the user of the mobile device. In addition, the developer does not get full access to the hardware layer, which makes it difficult or sometimes impossible to implement specific sensor-related features.

2.5.2 Android

The *Android*[3] platform is an open source mobile operating systems developed and maintained by *Google Inc.* According to Gandhewar and Sheikh (2010), the Android SDK was first presented by Google in late 2007 and was first available in late 2008 to be used by developers. In contrast to their main competitor, Apple's iOS, Google partnered with different hardware vendors and Android is available on many different devices of different manufacturers. Because Android is open source, any vendor can use it as an SDK and is able to customise it.
The philosophy of the Android platform is to be open and expandable. So, beside an open source core, developers can easily get access to all raw sensor data, other application data and even user data on the device, which differs heavily from Apple's closed approach for iOS.
As said in Butler (2011) the Android SDK, similar to iOS, uses a layered architecture, which is called the *Android development stack*, which is shown in Figure 2.4.

The Android stack is built on a modified Linux kernel and system components written in Java, C, C++ and XML and runs Java applications in a virtual machine. As stated in Meier (2010), Android developers are encouraged to develop their native Android applications using Java. The recommended development IDEs are *Eclipse* and *Android Studio*. Because Java is platform independent, developers are allowed to choose their platform, in which they want to develop their apps. In contrast, iOS developers have to buy Apple hardware in order to be able to develop native apps legally.
Similar to Apple, Google assists developers in the distribution of their apps using the *Google Play Store*. Beside the Play Store, there are other stores available for obtaining Android applications, for example the *Amazon Store*. As stated in Gand-

[3] `https://www.android.com`

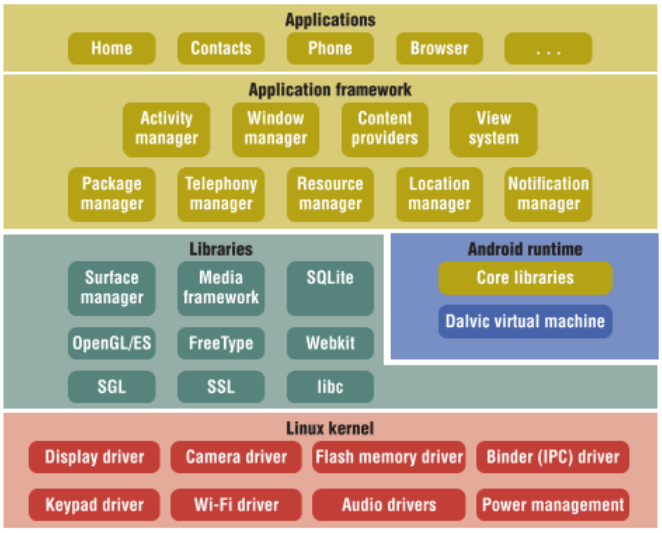

Figure 2.4: Android system architecture (from Butler (2011))

hewar and Sheikh (2010), there is no review process in these stores, so it is rather easy to submit apps and updates. The downside is, that there is no quality control in these stores.

According to Gandhewar and Sheikh (2010) the Android platform is very successful and still growing. Because it is available on a large number of devices, it has become the leading mobile platform according to market share.

3. Related Work

This chapter will first discuss the relevancy of different indoor localization methods for this thesis. Related work is presented for inertial and optical localization methods and trace visualization. In addition, approaches are presented that other applications have taken for locating lost items.

3.1 Relevant indoor localization methods

Different indoor localization methods were introduced in section 2.4, which are now investigated for their relevancy for this thesis.

Method	Relevant	Reason
Inertial localization	**yes**	
Satellite localization	no	not available indoors
Radio localization	no	requires additional hardware
Optical localization	**yes**	

Table 3.1: Relevancy of different localization methods for this thesis

As seen in table 3.1, only inertial and optical localization methods are relevant for this thesis, because these methods do not require any additional infrastructure and work indoors. Inertial sensors and cameras are build into most modern mobile device, so these methods are supported widely.

Based on this findings, the following sections will investigate different approaches for inertial and optical localization systems.

3.2 Inertial localization methods

Inertial methods refer to localization techniques using an *inertial measurement unit (IMU)*, which according to Jimenez et al. (2009) consists of several accelerometers, gyroscopes and sometimes magnetometers. As described earlier, inertial methods are always relative localization methods depending on initial positions.

3.2.1 Dead reckoning

As discussed in section 2.4.2, dead reckoning describes the process of estimating the location of an object by using velocity and direction based on an initial, absolute location.

The fact that sensors like accelerometers and gyroscopes are cheap and found in every modern mobile device, enables many applications to include dead reckoning approaches for localization tasks.

The disadvantages when using dead reckoning is that the accuracy decreases quickly due to the fact that errors are growing exponentially because new positions are always calculated relative to previously known positions. The task is even harder when using smartphone sensors, because they are cheaper and often inaccurate. As stated in Jimenez et al. (2009), the position where the sensors are mounted are important and affect the results, especially when dealing with pedestrian dead reckoning. Good positions are on the legs of a walking person, for example using the front pocket of the trousers.

While dead reckoning is less suited for long-term usage because of the error accumulation, it is often used as a valuable addition to absolute localization methods as presented in Baranski et al. (2009) and Kao (1991). It is able to support absolute methods that suffer from having information gaps or inaccuracies, for example when using GPS indoors or in urban areas.

Steinhoff and Schiele (2010) present a study of different dead reckoning solutions build for pedestrian localization. According to this study and many other related work, a common way to perform pedestrian dead reckoning is by combining *step detection algorithms* with *direction detection*. Additional steps increasing the accuracy are *stride length estimation* and *initial positioning*.

3.2.2 Step detection methods

Step detection has become a popular research topic because, as stated in Jin and Soh (2011) it provides valuable information for many different applications in healthcare, logistics and entertainment. The primary use cases are for activity recognition, pedometer applications and dead reckoning localization systems. These systems rely on the observation that humans produce very similar gait patterns. These patterns are clearly visible, when the raw signal of an accelerometer worn by a person while walking is visualized, as shown in Figure 3.1.

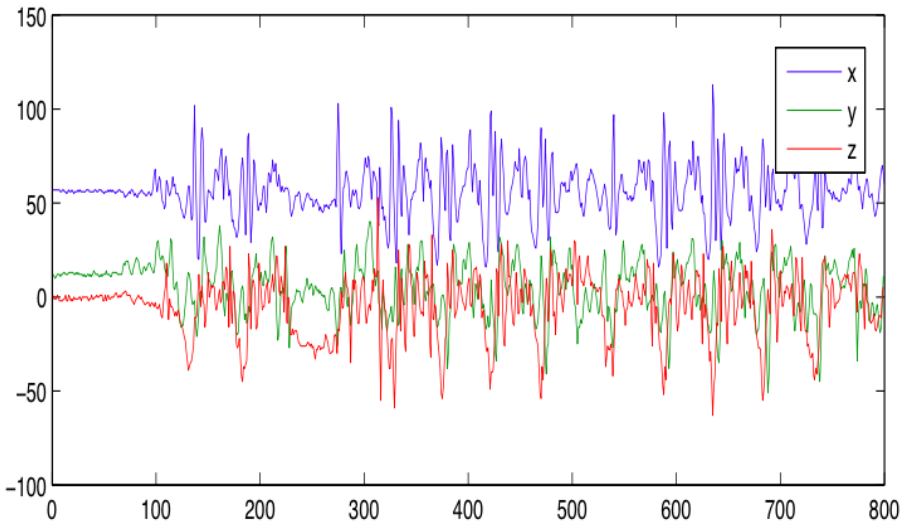

Figure 3.1: Raw accelerometer signal (from Mladenov and Mock (2009))

As stated in Mladenov and Mock (2009), the graph drawn from the raw signal shows the periodic up- and downwards motion of the human body as clearly indicated peaks while taking steps.

The human step

To be able to analyze gait patterns it is important to understand how a human step works. Kim et al. (2004) divide the activity of walking into the two phases: *walking* and *standing*. The walking phase is further divided into a *swing* and a *heel-touch-down* phase, as shown in Figure 3.2.

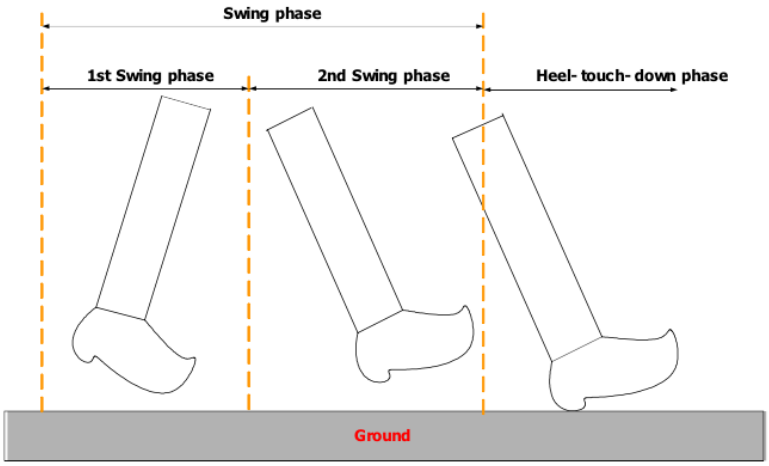

Figure 3.2: The human step (from Kim et al. (2004))

In the first swing phase the foot starts behind the human gravity center and is then accelerated in the second swing phase to end in the front of the human gravity center, before the foot is put back on the ground. The ground impact starts with the heel, before the sole and the toes have contact with the ground too.
The horizontal acceleration signal of one human step is shown in Figure 3.3.

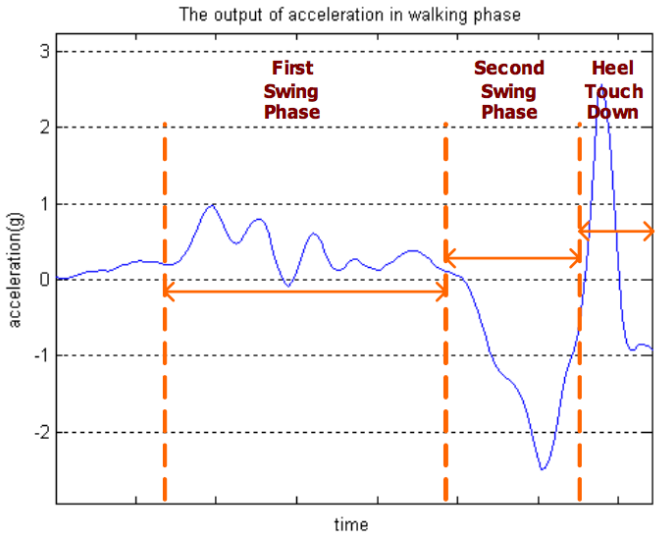

Figure 3.3: Horizontal acceleration signal of a human step (from Kim et al. (2004))

Step detection methods attempt to recognize these single step patterns in the raw signals to count the number of steps.

Step Detection using Accelerometers

Step detection performed with signals recorded from accelerometers is the most common method described in literature. As said before, it is not an easy task because accelerometers tend to drift and therefore to be inaccurate.
Raw signals of a three-axis accelerometer cannot directly be used for step detection. The signal has to go through multiple signal processing stages, depending on the chosen algorithm.
At least, each algorithm has to realign the signal to the user's frame of reference, because it is recorded in the sensor's frame of reference. According to Mladenov and Mock (2009), that means that none of the three axes of the accelerometer is assignable to the user's up and down motion produced while walking.

Mladenov and Mock (2009) describe an algorithm for performing reliable step counting using a three-axis accelerometer. Their first step is to calculate the magnitude of the acceleration vector and use it for further calculations. This is done by following formula.

$$a_i = \sqrt{a_{x_i}^2 + a_{y_i}^2 + a_{z_i}^2}$$

Using the magnitude instead of single axis values is a rather simple solution for getting useful movement information in the desired reference frame.
The second step is to flatten the signal curve by applying a *lowpass filter* on the data. This step is very important and is useful for every algorithm. Accelerometers are very subtle devices and recognize even the smallest movements like muscle reactions. These small movements are filtered out by using the lowpass filter. According to the authors, a frequency of 5Hz is suitable to detect walking or running steps.
After applying the filter, a peak-detection algorithm is performed. A threshold-based peak-detection is used, which has the disadvantage that thresholds can vary between different people. To mitigate this problem, they are performing two loops, where the output of the first is a mean value for a peak, which is used as threshold for the second loop. The limitation of this approach is, that the algorithm is not able to perform on-the-fly step detection.
For the experiment the authors tested various positions to carry the sensor like a

belt clip or different pockets of trousers. The result of the algorithm is shown in Figure 3.4.

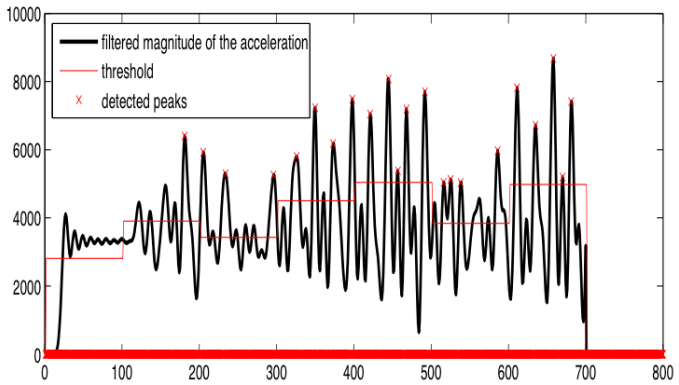

Figure 3.4: Detected peaks in the processed acceleration signal (from Mladenov and Mock (2009))

Accelerometer based step detection methods usually follow a similar principle as the method described above. Jin and Soh (2011) introduce a slightly different approach. Instead of calculating the magnitude of the input vector, they are using the measurements from the digital compass integrated in the device. The acceleration values are multiplied with the inverse of the rotation matrix in reversed order to obtain the acceleration values in world coordinate system.

Another approach is taken by Tumkur and Subbiah (2012), where the raw values are not transformed to world coordinates. They simply choose the axis with the largest acceleration change to detect steps on it. Minimal signal changes are discarded. Additionally to the peak threshold a time-threshold is added. A step has to occur between the time window of 0.2 and 2.0 seconds in order to count as valid step.

Horita et al. (2008) created a proposal to adopt step counting algorithms for elderly people. As also described in Marschollek and Goevercin (2008), step detection methods gain problems with elderly and impaired people, because they often have very uneven gait patterns, which makes it difficult to calculate proper threshold for the peak detection algorithm. Horita et al. (2008) propose a filter bank including seven different band-filters to process the raw accelerometer signal, then a step-cycle related signal is extracted to predict a threshold.

Step Detection using Gyroscopes

Accelerometer based step detection methods are working well, but as stated in Abhayasinghe and Murray (2012) they have drawbacks when the walking speed of the test subject is rather low. Thus, it is difficult to use this technique with elderly or impaired people.

Jayalath and Abhayasinghe (2013) present a novel approach using solely a gyroscope for detecting steps. Their findings are based on earlier work introduced in Abhayasinghe and Murray (2012).

The algorithm is build on the observation, that the human leg shows a sinusoidal behaviour while walking. Depending on the orientation of the device, one axis is showing the sinus curve when the data is visualized. In order for the algorithm to work properly, it is important how the device is placed. As described in the study, the recommended position to carry a mobile phone is to mount it vertically in the front pocket of the trousers. For other carrying positions, only the input axis of the sensor has to be changed in order to make to algorithm work again. Figure 3.5 shows a visualization of x-axis readings of a gyroscope carried in the front pocket, together with the thigh orientation of a person walking on flat land.

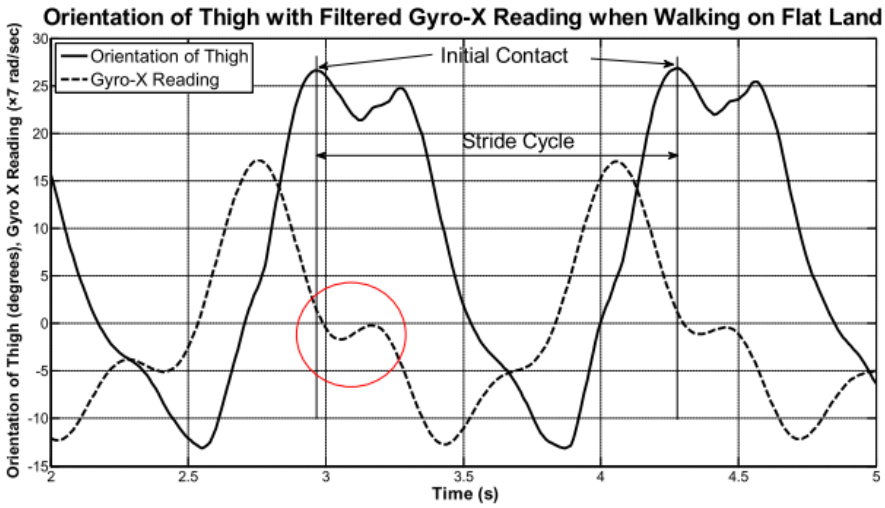

Figure 3.5: Thigh position recorded with a gyroscope (from Jayalath and Abhayasinghe (2013))

As indicated in the visualization, the sinus curve created by the gyroscope is clearly visible. The same results were found for people walking up stairs or hills.

The first step of the algorithm is to apply a low-pass filter on the raw sensor readings.

From the observation, that typical walking speed is between 1.5 to 3 steps a second, a filter with a cutoff frequency of 0.9Hz to 3Hz was chosen.
The identification of a valid step relies on the fact, that the thigh produces a sine-like curve, which crosses zero in each step. So the algorithm detects consecutive zero crossings, which are shown in Figure 3.6.

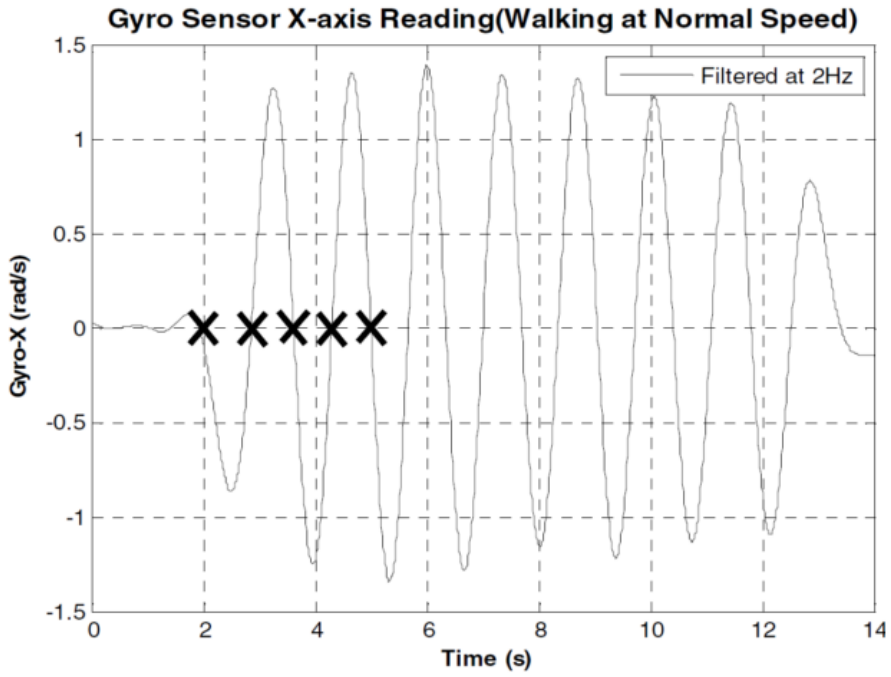

Figure 3.6: Zero crossing in the gyroscope signal (from Jayalath and Abhayasinghe (2013))

After a zero cross is detected, the algorithm uses a threshold for peak detection. Only if the peak is higher than the threshold, it will count as a valid step. The threshold is important to eliminate unwanted steps caused by small device movements. The threshold can vary between different test subjects and is depending on the position the device is placed. For example, it would be possible to change location of the device, if the threshold is adapted correspondingly.
A second threshold is used, which validates the duration of a step. A valid step only occurs between a time range of 0.4 to 1.2 seconds, depending on the walking speed. Everything which is detected as step outside the time range is discarded.
The algorithm was implemented as part of an iPhone application and tested with 5 female and 5 male users with varying age. The results where exceptionally good, with an accuracy of over 95%. The test subjects were instructed to perform different activities including walking on stairs or hills.

3.2.3 Direction detection

As stated in Smittle et al. (2010), direction detection, also referred to as *heading detection*, is the process of calculating or estimating the direction in which a device is heading. According to Bowditch (2002) in marine navigation this task is fulfilled by navigators using a traditional magnetic compass.
As stated in Kim et al. (2004), modern devices use either a magnetometer, or *digital compass*, or a gyroscope to perform direction detection. The characteristics of the two sensors are shown in Table 3.2.

Sensor	**Advantage**	**Disadvantage**
Magnetometer	absolute azimuth long term stable accuracy	unpredictable external disturbances
Gyroscope	no external disturbances short term accuracy	relative azimuth drift

Table 3.2: Comparison of magnetometer and gyroscope (adapted from Kim et al. (2004))

As seen in the comparison, the disadvantage of one sensor is the advantage of the other, so a reliable direction detection system might integrate both sensors. The described algorithm tries to detect interferences of the magnetometer by calculating the angular rate of the magnetometer and the gyroscope. If the difference is higher than a threshold, the compass readings are ignored.
A similar approach is shown in Klingbeil et al. (2010), where an extended Kalman Filter employs magnetometer and gyroscope readings. When the filter recognizes a disturbance of the magnetic field, the filter parameters are changed to trust more in the angular rate taken from the gyroscope.
Kothari et al. (2011) also present a similar solution for inertial direction detection. The solution consists of two complimentary methods. The first method uses accelerometer and magnetometer. While the magnetometer is able to give a reference to the magnetic north, an accelerometer can provide a reference direction for gravity. The benefit is that each measurement is externally referenced and errors are not accumulated.
The second method uses a gyroscope to measure the relative movement of the phone. Gyroscopes are less noisy than accelerometers and are not as vulnerable to external

interference than magnetometers.
The presented algorithm combines the two methods as shown in Figure 3.7, to mitigate each methods weakness. The algorithm detects gyroscope drifts and magnetometer inferences.

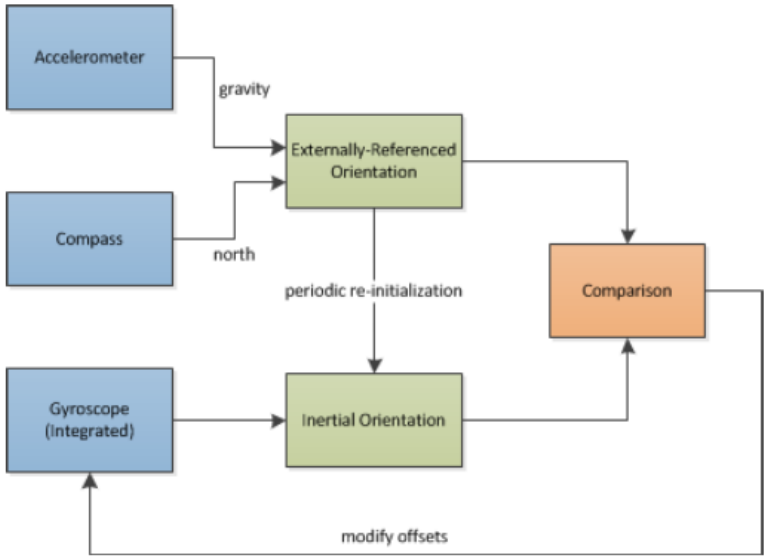

Figure 3.7: Robust heading determination (from Kothari et al. (2011))

Steinhoff and Schiele (2010) introduce a method to detect the direction a user is heading by determining the motion axis and the forward movement. This is done by applying the measurements of the gyroscope in the rotation matrix on the accelerometer readings.
Smittle et al. (2010) use an averaging method to weaken inaccuracies in the direction detection, caused by interferences magnetometers are vulnerable to. The average of the last n directions is taken into account, when a new direction is calculated.
A very simple method is described in Link et al. (2011), where the direction detection is performed as part of the step detection. When a step is detected, the current azimuth of the compass is taken and, together with the step information, passed to the path matching algorithms for further processing. While this approach is very simple to implement, it is vulnerable to external interferences disturbing the magnetometer.

3.2.4 Step length estimation

Step Length Estimation or *Stride Length Estimation* is a valuable addition to step detection algortihms, that is used as part of dead reckoning systems. As shown in Serra et al. (2010), it is sufficient for prototype-systems to use a fixed-value step length depending on the user's height and stride length, which gives reasonable results. An average step length of a person might be around 70 centimeters. Steinhoff and Schiele (2010) also emphasized that a fixed step length is sufficient, because step frequency and length hardly varies. As shown in their results, the angular error was the dominant error source, although a fixed step length was chosen.

According to Tumkur and Subbiah (2012), the accuracy of a dead reckoning system can be improved by employing a step length estimation algorithm, which is able to calculate the length of individual steps. Using fixed step length adds a cumulative error, which results in a displacement and wrong distance the user has covered. But, as stated in Renaudin et al. (2012), calculating a step length of a user out of inertial sensors is a very challenging task.

Zhao (2010) presents a solution depending on the walking speed and the body height of a person. The walking speed is measured in *steps per 2 seconds*. Table 3.3 shows, how the stride length is computed.

Speed (steps/2s)	**Stride Length (m/s)**
0-2	Height/5
2-3	Height/4
3-4	Height/3
4-5	Height/2
5-6	Height/1.2
6-8	Height
>8	Height*1.2

Table 3.3: Stride length depending on height and walking speed (adapted from Zhao (2010))

The disadvantage of this simple estimation method is that the algorithm needs the height parameter as input, which can not be calculated and thus has to be entered by the user.

Smittle et al. (2010) also present an approach, based on the user's height. Different methods were evaluated to calculate the stride length. The first method was to

let the user walk a distance of 9 meters for three times. The number of steps was counted and the average was taken. The second method used a simple formula to calculate the stride length. For females, the formula is *Height * 0.413*, for males *Height * 0.415*. Again, both methods depend on user input or calibration.
The algorithm presented in Weinberg (2002), relies on the observation the the stride length is proportional to the vertical movement, or *bounce*, of the human hip. The behaviour of the hip while walking is shown in Figure 3.8.

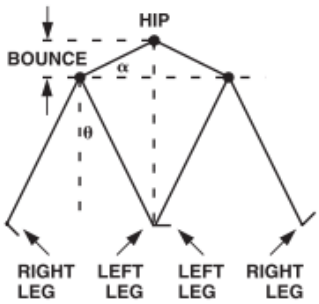

Figure 3.8: Vertical movement of the hip while walking (from Weinberg (2002))

The step length is calculated from the largest acceleration differences in each step. For retrieval, the following formula is used:

$$length = K \cdot n \cdot \sqrt[4]{a_{v_{max}} - a_{v_{min}}}$$

$a_k^{v_{min}}$ and $a_k^{v_{max}}$ are the minimum and maximum values of the vertical acceleration, calculated for n^{th} step. K is a constant applied for unit conversion.
Feliz et al. (2009) proposed an algorithm known as *Zero Velocity Update*. The algorithm uses the observation seen in Figure 3.2, that the human step is divided in a swing and a stand phase. When a stand phase is detected, any velocity higher than zero is an error produced by inertial measurements. These errors are removed by the algorithm, and a correction is applied to the current step. The corrected acceleration values are transformed to linear velocity through integration. The stride length is then computed from the position update using the horizontal cartesian distance.
Jimenez et al. (2009) compares the algorithms introduced by Weinberg (2002) and Feliz et al. (2009) as part of a dead reckoning system. An experiment was performed including tests with slow, normal and fast walking. A total distance of 360m was covered. The correct step length for every step is unknown, but the average was

taken as reference by dividing the total distance with the total number of steps. The results are shown in Table 3.4.

Walking Speed	Weinberg (2002)	Feliz et al. (2009)
Slow	-1.10m (0.3%)	-2.23m (0.62%)
Normal	-2.64m (0.73%)	4.15m (1.15%)
Fast	-2.81m (0.78%)	-3.47m (0.97%)

Table 3.4: Comparison of Weinberg (2002) and Feliz et al. (2009) (adapted from Jimenez et al. (2009))

When analyzing the findings of Table 3.4, both algorithms deliver very accurate results around 1% of the total distance, with a subtle advantage for the algorithm presented in Weinberg (2002).

3.2.5 Retrieving the initial location

As stated in Section 3.2.1, dead reckoning is a relative localization method. To be able to calculate locations relative to the previous one, an initial location has to be known before the algorithms are able to start. Because relative localization methods can not calculate absolute positions, these initial position have to be provided in order for the algorithms to work properly.

Baranski et al. (2009) propose to retrieve the initial position by combining GPS with dead reckoning methods. The same result can be accomplished by using other absolute localization methods.

Smittle et al. (2010) also describe the problem of location initialization, especially when GPS is unavailable, which might be a frequent problem in indoor environment. The suggestion is to allow the user to select a starting location. Issues that could arise with this approach is that users do not know their location, or to find a proper user interface for selecting the location. For example, maps are usually not available in indoor environments.

Serra et al. (2010) propose a fixed initial location, which is known by the test user and the application. For example the test route could always start at the front door of a building.

3.3 Optical localization methods

Optical localization methods are using images taken from cameras as input for the localization algorithm. According to Mulloni et al. (2009), optical localization methods are often marker-based and thus require an infrastructure to work properly. They are scanning QR-codes or barcodes containing unique identifiers, that are used for orientation inside a building. Markers are used because they are cheap and the implementation of a marker-based system is less complicated. Because marker-based systems require an infrastructure and do not work solely with a smartphone, they are considered as not relevant for this thesis.

Beside marker-based systems, there are many other approaches which are using the images taken from the camera for localization. Aubeck et al. (2011) present a solution that performs optical step detection. It is added as additional sensor element to a inertial localization system using dead reckoning methods. The step detection is accomplished by recognizing both feet appearing alternately in the camera image. This is done by capturing grayscale images and using template images to be able to apply a cross-correlation algorithm. The template generation is applied dynamically, to be able to react to different environment, light conditions and shoe forms. In order for the algorithm to work properly, the smartphone has to be carried in the hand of the user, pointing with the camera towards the ground. The evaluation of the camera based step detection shows promising results and can support inertial systems, especially when users are walking very slow. The system has difficulties when users are walking fast or standing still. Difficult light conditions are also a problem for camera based systems, because shadows are interfering the image recognition process.

The solution presented in Ruotsalainen et al. (2011) has a goal similar to Aubeck et al. (2011). The proposed system is two dimensional as well, adding an optical localization method as addition to an already existing indoor localization method. But instead of increasing the accuracy of step detection methods, the presented system aims to increase the accuracy of the heading detection. The proposed algorithm is based on the observation, that images are a projection of three-dimensional

object into two-dimensional figures. The depth information is lost when a picture is taken. Parallel lines in projections won't stay parallel, they all intersect in a point called *vanishing point*. The algorithm uses this vanishing point, to track its coordinate changes whenever the camera moves, by comparing consecutive images. According to the results of the evaluation, adding the additional optical phase to a WiFi based indoor navigation system, improved the global accuracy of the system.

The two presented systems applied an additional optical localization method to an already existing localization system with the goal to improve the accuracy. As stated in Ruotsalainen et al. (2011), integrating the measurements of different, independent sensors increases the accuracy, continuity, availability and integrity of a localization system. Both solutions are relative localization techniques, also having all disadvantages of relative systems like error drifts.

Optical localization methods also enable the possibility to develop absolute localization systems. Such an absolute system is presented in Elloumi et al. (2013). The solution consists of two phases. First, a learning phase is conducted, where a reference path is defined by selecting key frames along the track. Saliency extraction methods are used for accomplishing this task. With the extracted key frames it is possible to calculate the orientation of the camera. In the second phase, the localization phase, the recorded video stream is used to compare the current orientation with the orientation information gathered in the learning phase. In the localization phase, the algorithm is able to provide navigation and localization information in real time. For the evaluation of the system users had to carry their smartphone fixed on their chest, with the camera pointing in front of them. The localization and navigation results were promising, but the system is not able to estimate the walking speed using only optical methods. In a future version of the system, a pedometer should be added to accomplish this task.

Werner et al. (2011) also introduce a marker-less indoor navigation system using a smartphone camera. The presented solution combines an image recognition system with a distance estimation algorithm. The proposed algorithm works similar to WiFi fingerprinting. A database of images is created in a training phase, which is

done offline. The location is then estimated in the localization phase by comparing the scale of a reference point in the database with the scale in the actual image, as shown in figure 3.9.

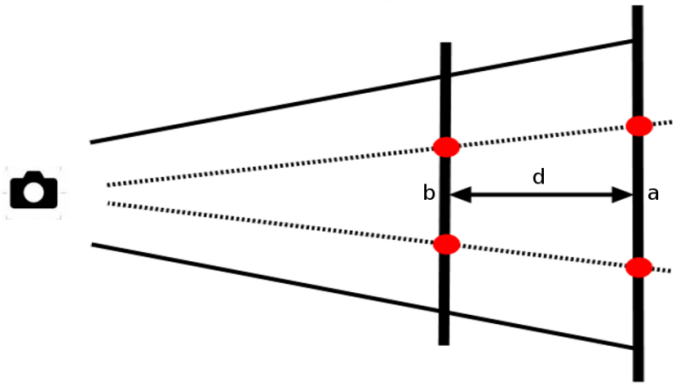

Figure 3.9: Distance estimation between reference image and actual image (from Werner et al. (2011))

A very interesting feature of the algorithm is, that beside video streams, it also supports static images as input. So users can retrieve their current location by just taking a picture, that is processable by the algorithm.

3.4 Trace visualization

To be able to analyze the results of an indoor localization system, it is important to add a proper visualization. Raw signal data, which is often the starting point of such localization systems, is usually visualized using basic line charts, as shown in figure 3.1. Creating a convincing visualization of localization information is a more complex task.

The foremost method for the visualization of trace data is to use line charts. Line charts are plots of sequences of single coordinates in cartesian coordinate systems. The chart is usually drawn two-dimensionally, on an x-axis and an y-axis. If a step detection based system is used, the single steps are often highlighted using bigger data points. A popular method to establish connections to real world locations is by adding background to the visualizations showing maps in outdoor environments and floor plans in indoor environments. Examples for this visualization method are

presented in Jimenez et al. (2009), which are shown in figure 3.10, and Steinhoff and Schiele (2010).

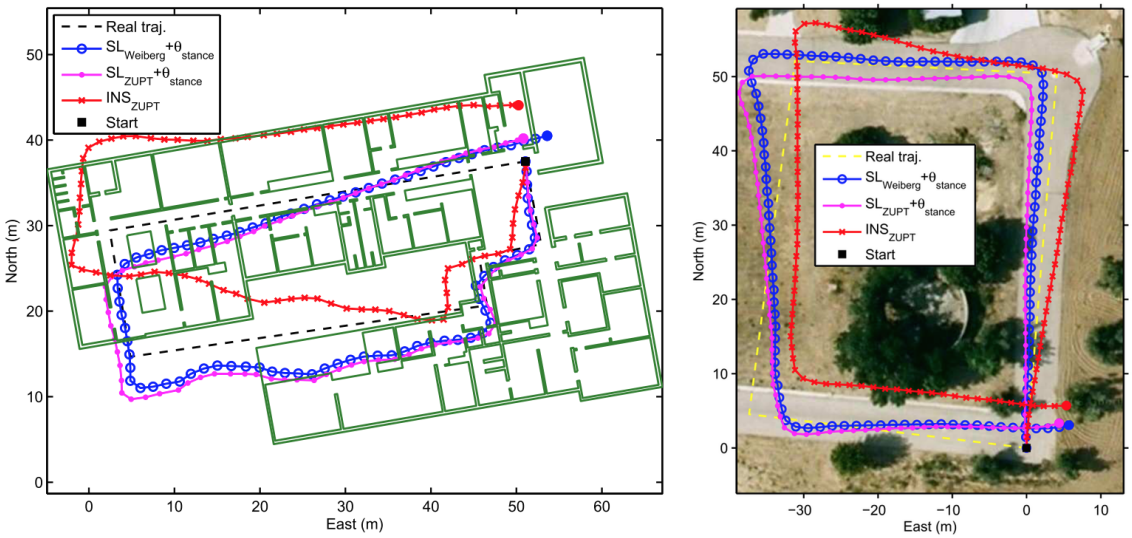

Figure 3.10: Visualizations of pedestrian traces (from Jimenez et al. (2009))

3.5 Apps for locating lost items

Locating lost items is a problem many people face in their daily life, thus many companies are engaged in finding solutions to this problem. Most of the applications are utilizing bluetooth technologies and require external markers attached to the personal items in order to retrieve them.

Find my iPhone

One of the first applications that was available for the iOS platform was called *Find My iPhone*[1] and was published by Apple. The app, which is available for free, is used to send the location of the current device to the server application. In the case that the iPhone is lost or misplaced, the website reveals the position of the device by showing an indicator on a map. The application is also available for iPad, iPod touch and Mac OS X. The tagging of the current position is accomplished by using GPS and WiFi technology.

[1] `https://itunes.apple.com/at/app/mein-iphone-suchen/id376101648?mt=8`

The same functionality is available for Android devices using the application *Android Device Manager*[2].

Tile

Tile[3] uses small bluetooth markers in the form of stickers. These stickers are attached to personal items to be able to detect them. Tile offers two options to find the items. The first is by using their smartphone app, which shows a visualization similar to a radar to navigate the user to a bluetooth marker. The second method is to trigger a squeaking sound on the markers, so the position is revealed through to noise. Tile is available for iOS and Android. The necessary bluetooth markers can be purchased for 25 USD each from the company's web shop.

Duet

A similar system is developed by ProTag and is called *Duet*[4], which also relies on bluetooth markers. Instead of using stickers, the bluetooth markers are manufactured as key pendants. As the name *Duet* indicates, the markers have two use cases. The first is, that the markers are located by using the freely available iOS or Android application. The application uses a very simple interface, which shows the strength of the signal, that is detected. The second use case pairs one of the markers to the smartphone application. Whenever the distance between the marker and the smartphone exceeds a defined threshold, the marker starts creating noises. Therefore the markers do not only protect items, they are used to protect the smartphone too. The markers are available in different colors from the company's web shop at a price of 29 USD each.

[2] `https://play.google.com/store/apps/details?id=com.google.android.apps.adm`
[3] `https://www.thetileapp.com`
[4] `http://theprotag.com/de/product/duet/`

Trackr

Trackr[5] is a successfully funded crowd-funding project, which offers the same feature set as Duet, but in addition establishes a crowd GPS network. The idea behind this concept is, that when an item is marked lost, every Trackr user that enters the monitored region of the marker triggers an event, so that the owner of the marker receives a notification including the location of the item. Besides a visualization that indicates the distance to markers, Trackr offers a map view including the last known positions of the items. The positions are tagged by the smartphone locations whenever a bluetooth marker is within range. The smartphone apps are available for iOS and Android for free. The bluetooth markers are available for pre-order in the company's web shop for 29 USD.

StickNFind

StickNFind[6] is a similar system using bluetooth markers and offering the same features as other competitors, but additionally it is using a specific hardware setup to build indoor navigation systems. The focus of the company is in the enterprise market, using the same concepts for inventory and assets management in large companies.

My StuffFinder

The iOS Application *My StuffFinder*[7] is available in the iOS App Store for 0,89 USD. The application allows users to create entries for all items they want to track. Whenever a user places an item, the app has to be opened and the button for the corresponding item has to be pressed. The application then automatically remembers the position of the item. The application is based on GPS and WiFi technology, so this solution will not work accurately for retrieving items indoors. For finding the item it offers a standard map view. Besides adding location information, the

[5] `https://www.thetrackr.com`
[6] `https://www.sticknfind.com`
[7] `https://itunes.apple.com/us/app/my-stufffinder/id542000800?mt=8`

app allows the user to attach images or voice notes to the item. The last position can also be retrieved by looking at the picture where the item was placed.

4. Prototype Application

Based on the research presented in chapters 2 and 3, a prototype application was implemented. This chapter describes the conception, design and implementation of the application. The used software components and technologies will be explained briefly and problems encountered during the work on the prototype will be discussed.

4.1 Concept

The concept of the application is based on the findings gained by the research presented in previous chapters. It includes the formulation of requirements and use cases for the prototype application.

4.1.1 Requirements

Before starting the development of an application it is important to define the requirements. During the implementation, these requirements help to stay focused on what is important. The application is designed as research prototype and does not implement all necessary features to represent a product ready for sale. The following requirements were defined for the prototype application:

- The purpose of the application is to verify that it is possible to track and retrieve the position of different items solely using data collected from inertial sensors.
- The dead reckoning system should work with proper accuracy to allow the retrieval of lost items by users.

- The application is very easy to use and should not disturb users in their daily routine.
- The tracing algorithms are expected to perform live tracking without mandatory training phases.
- The visualizations should be easily understandable and clearly indicate the covered route as well as the positions of the placed items.

4.1.2 Use cases

For the user, the system has two simple but important use cases.

- The user is able to tag the location of a personal item, when it is put down somewhere.
- The user is able to retrieve the location of a personal item, if it is lost.

Everything else needed to fulfill the use cases described above happens automatically and without further user interaction.

4.2 System design

To provide an application capable of locating lost items solely using sensors build into common smartphones, it was decided to design a dead reckoning system using inertial sensors. Inertial methods were preferred, because optical methods were either added as addition to existing localization methods, or were dependent on a training phase. Beside the stated disadvantages of optical methods, inertial methods are better documented in literature and have a more active research community, as well as better existing software libraries.

The decision was made to design the application using a *client-server architecture*, because the primarily use will be to act as a prototype for evaluating the chosen algorithms and methodologies. Beside that, analyzing large datasets is much easier on desktop computers than on mobile devices, because of display size and computation power.

It is important to notice, that all information necessary for the system to operate is retrieved by the client, running directly on a common smartphone.
The client will be primary used for reading sensor data. Users also have the opportunity to tag the location of different items.
The server-part will implement all signal processing functionalities, including the dead reckoning system, and is responsible for creating the data visualizations.
Both applications will strictly follow all concepts suggested and described in Apple Inc. (2014).

4.3 Technical Requirements

As explained in sections 2.5.1 and 2.5.2, iOS and Android are both widely used and powerful development platforms. It was decided to implement the prototype as an iOS application. There was no technical reason for it, because as seen from the research, both platforms provide the needed libraries and accompanying documentation in order to be able to develop the prototype application. The decision was based on the fact, that more development experience and all necessary devices were available for the iOS platform.
Besides, as stated in section 2.5, the iOS platform is more restrictive than the Android platform. If the prototype is working on the more restrictive platform, the switch to other platforms should be easier.
The following technical requirements apply for the prototype:

- iPhone 4 or higher with iOS 7+ installed for the client
- the client application needs the operating system to provide an API to retrieve location and motion updates
- a Macintosh device running Mac OS X 10.9 or higher for the server application
- client and server are connected to the same WiFi in order to be able to exchange data
- Bluetooth is available and activated on both devices to establish the connection

4.4 Development environment

This section gives a brief overview of the hardware and software tools, that were used for the implementation of the prototype application.

4.4.1 Hardware

For the implementation and testing of the server application an 27-inch Apple iMac and a 13-inch Apple MacBook Air were used. For testing the client application, an Apple iPhone 5s and an Apple iPhone 6 were used.

4.4.2 XCode

XCode is the name of the integrated development environment (IDE) provided by Apple for developers to implement their applications. It is available free of charge in the Mac App Store. Apple[1] describes XCode as following:

"The Xcode IDE is at the center of the Apple development experience. Tightly integrated with the Cocoa and Cocoa Touch frameworks, Xcode is an incredibly productive environment for building amazing apps for Mac, iPhone, and iPad."

XCode has a lot of useful features like a powerful source code editor and corresponding build system, integrated interface builder, version management, test environment, documentation, debugger and much more.
The implementation of the prototype application started with XCode 5, but at soon as it was available, XCode was updated and used in version 6.

4.4.3 CocoaPods

CocoaPods[2] is an open source dependency manager for Objective-C. It is written in Ruby and is available as a Ruby Gem. It solved a huge issue with third party framework integration in XCode projects, which prior to CocoaPods required a lot

[1] `https://developer.apple.com/xcode`
[2] `http://cocoapods.org`

of changes to the build settings of the project, which was often connected with a serious time effort. CocoaPods moves the configuration of third party frameworks to a *Podfile*. The Podfile of the server application is shown in listing 4.1.

```
1  source 'https://github.com/CocoaPods/Specs.git'
2  platform :osx, '10.9'
3
4  inhibit_all_warnings!
5
6  pod 'CocoaAsyncSocket'
7  pod 'CorePlot'
```

Listing 4.1: Podfile configuration

After the Podfile is configured, typing "pod install" in the terminal is everything that is necessary to do, the rest is done automatically by CocoaPods. Instead of the XCode project file, it is necessary to use the generated workspace file, when CocoaPods are used.

4.4.4 Git

Git was used as version management system to be able to have a solid development workflow including versioning, branching and backup. The two repositories for client and server were hosted on *BitBucket*[3].

4.5 Implementation

As stated before, the prototype application was implemented using a client-server-architecture including a native iOS app for the client and a native Mac OS X application for the server part. The programming language used for the whole project was *Objective-C 2.0* and all interface elements were created using the build-in interface builder. The application is build on base of the Cocoa framework and the corresponding APIs. Only two third party frameworks were used, *CocoaAsyncSocket* for client-server communication and *CorePlot* for the visualizations. These frameworks will be described in more detail in the following subsections.

[3] `http://www.bitbucket.org`

Figure 4.1 provides an overview of the application's architecture and the most important classes.

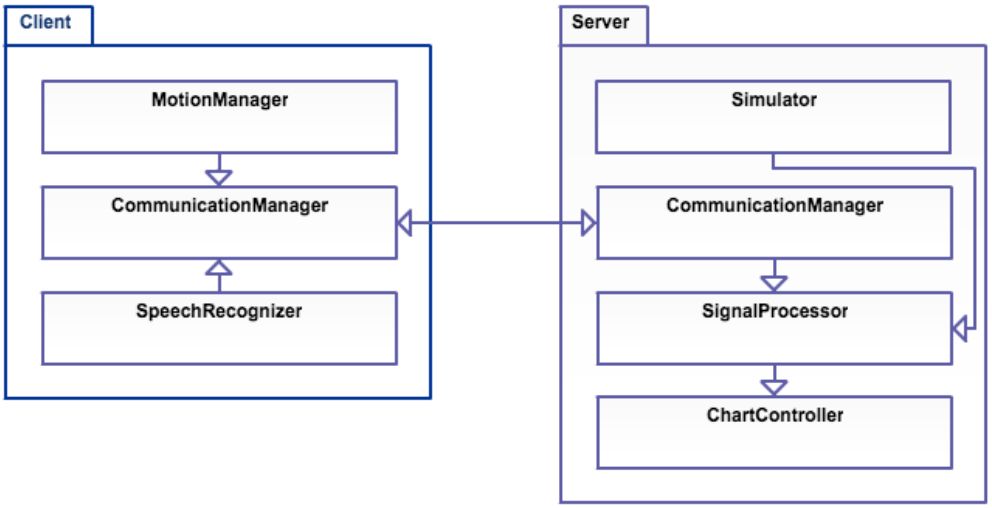

Figure 4.1: Architecture of the prototype application

4.5.1 Reading sensor data

The reading of sensor data is solely implemented on the client, which acts as data logger for the server. The *MotionTracker* class is responsible for performing this task. It basically acts as wrapper class for two libraries provided by the iOS SDK, *Core Motion*[4] and *Core Location*[5].
For analyzing and working with sensor data it is important to understand the concepts of the frameworks and the underlying classes, which deliver the necessary information.

CoreMotion

The Core Motion framework allows applications to access motion data from the device. It delivers the data containing motion and orientation of the device by observing changes of accelerometer, gyroscope and magnetometer. The data is accessible in raw form as well as in processed form, often by applying sensor fusion.

[4] `https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CoreMotion_Reference/index.html`
[5] `https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/`

The suggested way of retrieving motion data is by using an instance of *CMMotionManager*. This API class provides a block-based interface for retrieving motion data. The initialization and configuration of this class is shown in listing 4.2

```
1  ...
2  double updateInterval = 1.0/50.0; //50 Hz
3
4  CMMotionManager *motionManager = [CMMotionManager new];
5  motionManager.accelerometerUpdateInterval = updateInterval;
6  motionManager.gyroUpdateInterval = updateInterval;
7  motionManager.magnetometerUpdateInterval = updateInterval;
8  motionManager.deviceMotionUpdateInterval = updateInterval;
9  ...
```

Listing 4.2: CMMotionManager configuration

The update interval is set to a frequency of 50Hz or 0.02 seconds. As shown in listing 4.2, the CMMotionManager provides access to accelerometer, gyroscope, magnetometer, altimeter and device motion. While the first four options provide a way to obtain raw data from the corresponding sensor, the device motion delivers processed data using sensor fusion.

Core Motion uses several model classes to provide sensor data. The classes containing the raw data are *CMAccelerometerData*, *CMGyroData*, *CMMagnetometerData* and *CMAltitudeData*. The altimeter and the corresponding class CMAltitudeData are only available when using iOS 8 and the iPhone 6 or iPhone 6+. Because some of the features of Core Motion are only available on newer iOS devices, before starting the sensor updates the methods checking for the sensors availability should be invoked.

While the other model classes only provide the raw sensor data, including a struct of the three device axes, *CMDeviceMotion* delivers much more information. It includes processed data through sensor fusion of all available motion sensors. As stated in the documentation, this processed data includes attitude and rotation rate, gravity and user acceleration as well as the magnetic field, each as a separate class.

As described in Cook (2014), *CMAttitude* contains three different representations for the orientation of the device: Euler Angles including roll, pitch and yaw, a rotation matrix and a quaternion. *CMRotationRate* contains the rotation rates of the three axes as vector.

The raw accelerometer values of the device are a sum of gravity and user acceleration.

With the help of the gyroscope, the acceleration data available in CMDeviceMotion separates these two parts and makes them separately available in the *CMAcceleration* classes, by providing the values as gravity and userAcceleration.
Last but not least CMDeviceMotion offers the calibrated magnetic field, which is provided in respect to the device.
For all of the values described above, it is important to find the correct frame of reference. The attribute *CMAttitudeReferenceFrame* is used to assign the desired reference frame. The reference frame defines the orientation from which the device's attitude is calculated. All four possible values describe a device laying flat on the surface, where the specificity, concerning the direction the device is pointing, is increasing.

- *CMAttitudeReferenceFrameXArbitraryZVertical* is the default value and describes a device having an arbitrary x-axis, which is fixed to the orientation the device pointed to when motion updates were started.

- *CMAttitudeReferenceFrameXArbitraryCorrectedZVertical* has the same behaviour as CMAttitudeReferenceFrameXArbitraryZVertical, but uses the magnetometer to prevent from long-term inaccuracy due to gyroscope drift, which results in increased CPU usage and reduced battery life.

- *CMAttitudeReferenceFrameXMagneticNorthZVertical* describes a device with an an x-axis pointing to magnetic north. Using this reference frame may require magnetometer calibration.

- *CMAttitudeReferenceFrameXTrueNorthZVertical* describes a reference frame where the x-axis of the device is pointing in the direction of the true north. Beside calibration, it requires location data to calculate the difference between magnetic and true north.

For the implementation of the prototype, the processed sensor data provided by CMDeviceMotion is used in the reference frame CMAttitudeReferenceFrameXMagneticNorthZVertical. This reference frame is selected, because the attitude is provided relative to the magnetic north, which is used for the direction detection. The configuration is shown in listing 4.3.

```
1  ...
2  CMAttitudeReferenceFrame frame = CMAttitudeReferenceFrameXMagneticNorthZVertical
       ;
3  NSOperationQueue *queue = [[NSOperationQueue alloc] init];
4  [self.motionManager startDeviceMotionUpdatesUsingReferenceFrame:frame
5                                                          toQueue:queue
6                                                      withHandler:^(CMDeviceMotion *
                                                           motion, NSError *error) {
7                                                        // process motion
8                                                      }];
9  }
10 ...
```

Listing 4.3: Device motion updates

Every time the update handler is called by the CMMotionManager, the CMDeviceMotion object is serialized and sent to the server for further processing.

Core Location

The Core Location framework allows applications to determine the current location and heading of the device. It utilizes GPS, WiFi, Bluetooth and motion sensors. It is also useful for performing tasks like region monitoring, either geographical or using iBeacons.
Core Location is used similarly to Core Motion. A CLLocationManager instance is used for observing location or heading updates. The configuration of the CLLocationManager is shown in listing 4.4

```
1  ...
2  CLLocationManager *locationManager = [CLLocationManager new];
3  locationManager.desiredAccuracy = kCLLocationAccuracyBest;
4  locationManager.headingFilter = 1.0;
5  locationManager.delegate = self;
6
7  [locationManager requestWhenInUseAuthorization];
8  [locationManager startUpdatingHeading];
9  ...
```

Listing 4.4: CLLocationManager configuration

The interface of the CLLocationManager is not block based, thus the updates are received by implementing a delegate protocol. Since the introduction of iOS 8 it is mandatory to invoke the *requestWhenInUseAuthorization* method first, otherwise

Core Location will not fire any updates. The most important model class associated with the CLLocationManager is probably the *CLLocation* class, which contains information of the current location of the device. For the prototype, only the heading information is needed, which is provided as *CLHeading*. The class contains the *magneticHeading* and the *trueHeading*. The magneticHeading represents the heading relative to the magnetic North Pole, which is different from the geographic North Pole, which is represented in trueHeading. Retrieving trueHeading requires additional location data, which is retrieved from GPS or WiFi. Beside the processed heading values, CLHeading contains heading accuracy values and raw heading values.

```
1  ...
2  - (void)locationManager:(CLLocationManager *)manager didUpdateHeading:(CLHeading
       *)newHeading
3  {
4    // process newHeading
5  }
6  ...
```

Listing 4.5: CLHeading retrieval

Listing 4.5 shows the implementation of the delegate method which is called by Core Location when new heading information is available. Similar to CMDeviceMotion, the CLHeading data is serialized and sent to the server as soon as it is retrieved.

4.5.2 Client-server communication

The logging of motion and location events generates a lot of data, which is sent from the client to the server. To be able to handle the large amount of traffic, it was decided to build on socket connections. For this task, the third party framework *CocoaAsyncSocket*[6] was added to the project, which provides powerful TCP/UDP sockets. The big advantage of the library is, that it provides non-blocking, asynchronous sockets. The framework provides a high-level Objective-C API.

In the prototype application, the client and server part each consist of a *CommunicationController*, which acts as wrapper class around the socket library. For

[6] `https://github.com/robbiehanson/CocoaAsyncSocket`

establishing the connection, the server socket is bound to a port and waits for clients to connect. In order to do that, the client needs to know the IP-address of the server. A simple solution would be to enter it manually in the client application, but this would require user interaction. The better solution is to integrate *Bonjour*[7], which allows to establish of a network connection with zero configuration. The server starts to broadcast a Bonjour service, which is discoverable by clients. The service allows the client to retrieve the IP-address of the server, which makes it possible to connect to the server without user interaction.
Once the connection is established, the two applications can use the socket for communication. The prototype application uses an asynchronous UDP socket. To be able to dispatch the received data on the server, a command system was implemented. The following commands are supported by the protocol:

- *connected* is sent from client to server once, immediately after the client has established the connection.
- *start-tracking* is sent when the user starts the tracking.
- *stop-tracking* is sent when the user stops the tracking.
- *motion-data* is sent whenever an update of motion data is available. The data is sent together with the command.
- *placed-item* is sent after the user has placed an item. The identifier of the placed item is sent together with the command. Valid identifiers for the prototype application are:
  - *keychain*
  - *wallet*
  - *watch*
- *heartbeat* is used to verify that the connection between client and server is still alive. This command is sent by the client each second.

Once a command is recognized on the server, the data is handed to the delegate of the CommunicationController for further processing.

[7] `https://developer.apple.com/bonjour/index.html`

The handling of network errors is very simple for the prototype. If the network connection fails, the current tracking session is interrupted and has to be started again.
During the implementation of the system it was verified, that the UDP socket can handle the large amount of traffic, produced by reading motion data every 0.02 seconds, without any problems.

4.5.3 Simulator

The *Simulator* was a highly important class to develop the dead reckoning system. The use case of this component is very simple. It allows to replay previously recorded tracking sessions. Every tracking session, that is recorded by the server application, is written to a file containing all received commands. The simulator can read these files and processes the contained data. The data is read with the same interval the client uses for logging, in case of the prototype 50HZ. The data is handed in the same way to the SignalProcessor, as if the data was just received from the client.
Without the simulator it would be very time consuming and difficult to increase the performance of the dead reckoning system, because it would require a lot of human testing. Using the simulator, the development was performed only with few test files filled with data recorded by tests from different people using the client application. An advantage of using the same test data is, that the results of changes to the algorithms could be compared easily.
Beside improving the dead reckoning system, the simulator was also important for drawing the visualizations. The simulator was written to process on file at a time, to be able to analyze the results.

4.5.4 Dead reckoning

The *heart* of the system is located in the *SignalProcessor* class. The purpose of this class is to transform the sensor data received from the client into meaningful information, which is suitable for locating pedestrians using the application while carrying the device. As discussed in chapter 3.2.1, a dead reckoning system consists of multiple steps. The individually steps are described in the following subsections.

Getting the initial position

As explained in section 3.2.5, retrieving the initial position for a dead reckoning system is a tricky task. Because the prototype application is expected to work indoors, the usage of GPS was not an option. Other absolute methods like WiFi fingerprinting require additional hardware, which was not an option either. The simple solution for this issue was, to require users to start tracking on a fixed location, as it is suggested in Serra et al. (2010).
During the implementation of the dead reckoning system, the initial position was not important. Tracking was implemented relative to a starting location, which was unknown. The results of the step detection and direction estimation algorithms did not depend on a valid initial location. However, in the field experiment and for users to know where they placed their items, it is important to use a fixed initial location, to be able to recognize the location inside a building.

Signal processing

Processing the signal in order to use it for step detection was one of the most challenging tasks of the development process. To be able to detect step patterns in the signal, it has to be smoothened. As explained in section 3.2.2, raw signals contain data, which is produced even from the smallest movements like muscle reactions. Unfortunately, these small movements could have a very similar pattern compared to a typical human step and would therefore be recognized falsely, which would lead to inaccurate results.
To counteract this issue, a low-pass filter was implemented and applied to the signal. According to Shanklin et al. (2011), a low-pass-filter has the characteristic to let frequencies below the cutoff frequency pass through undiminished, while frequencies above the cutoff frequency are subdued. This behaviour leads to a delay in the growth of the sinus curve of the signal. The following formula is used for describing the filter.

$$K = K_i * \epsilon + K * (1 - \epsilon)$$

K refers to a single signal value, for example the acceleration on the x-axis. The filter has to be applied for each axis individually by using the same formula. The ϵ

describes the filtering factor. This factor has to be adapted by observation, unless it is set to a value that works well for the provided signal and the desired use case. For the prototype application, it was discovered that a value of *0.2* works best, which corresponds to a cutoff frequency of 5Hz. It was confirmed by Abhayasinghe and Murray (2012) and Jayalath and Abhayasinghe (2013), that cutoff frequencies of 5 Hz are suitable for low-pass filters. Figure 4.2 shows the result of the lowpass filter applied to the x-axis reading of the accelerometer.

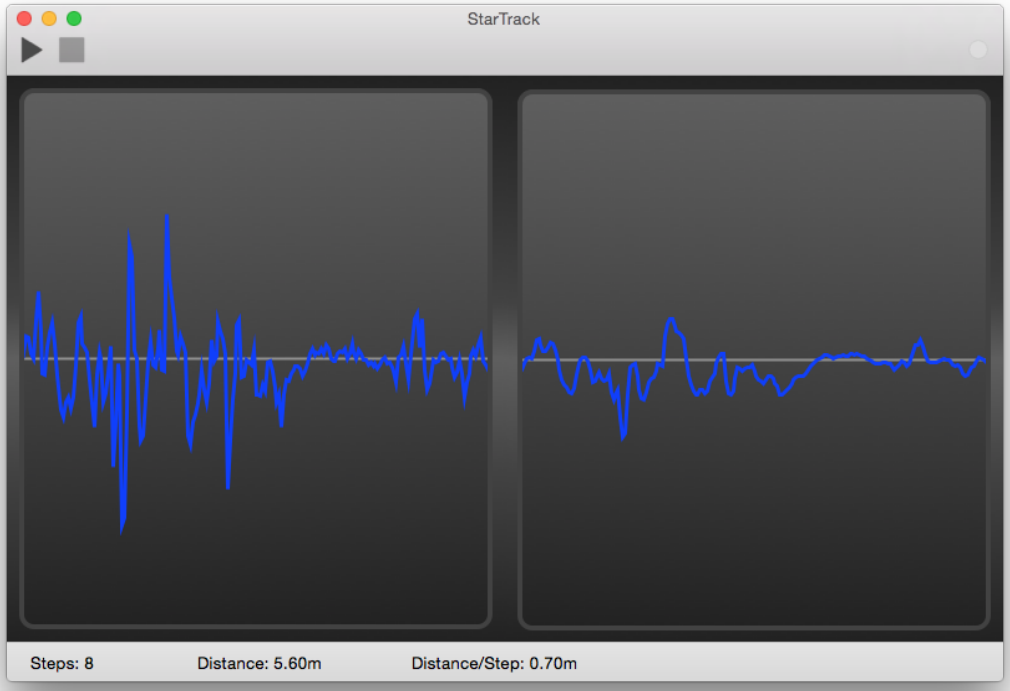

Figure 4.2: Comparison of signals with or without low-pass filter

The chart on the left side shows the original signal without processing, where a low-pass filter with a cutoff frequency of 5 Hz is applied to the signal visualized in the right chart. As clearly noticeable in the comparison, the processed signal is much smoother and has less small movements.

Step detection

As discussed in section 3.2.2, there are different approaches for implementing step detection from sensor signals. The two approaches explained in Mladenov and Mock

(2009) and Jayalath and Abhayasinghe (2013) were inspected in more detail. The first algorithm uses accelerometer readings, whereas the second depends on gyroscope readings.
The first step was to visualize the signals to be able to compare them and recognize step patterns. The result is shown in figure 4.3.

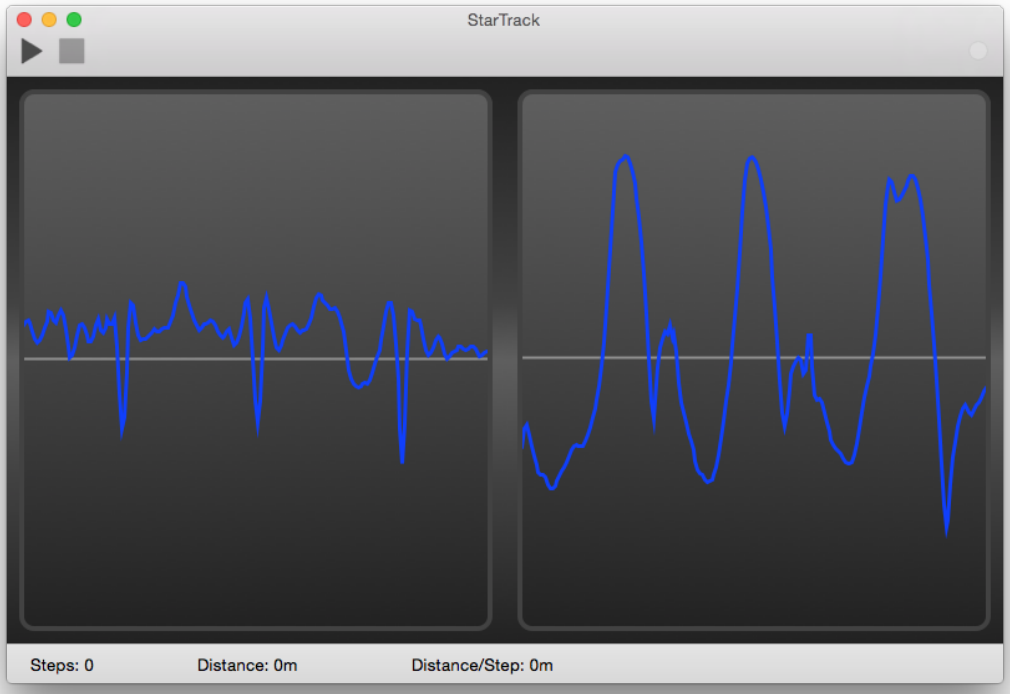

Figure 4.3: Comparison of accelerator and gyroscope signals

The two signals show the same walking sequence. As noticeable from the visualization, the step pattern is clearly recognizable with both sensors. Because it was highlighted in Jayalath and Abhayasinghe (2013), that the presented algorithm is more reliable and accurate than algorithms using an accelerometer, it was decided to continue work based on gyroscope signals.
A requirement for the algorithm to work is, that users will fix their smartphone somewhere near around the hip. According to Jayalath and Abhayasinghe (2013), a good position is the front pocket of the trousers. The smartphone is then placed vertically in the pocket, having a horizontal x-axis and z-axis and a vertical y-axis. Figure 4.4 shows the rotation of the device according to the swing phases of a single step.

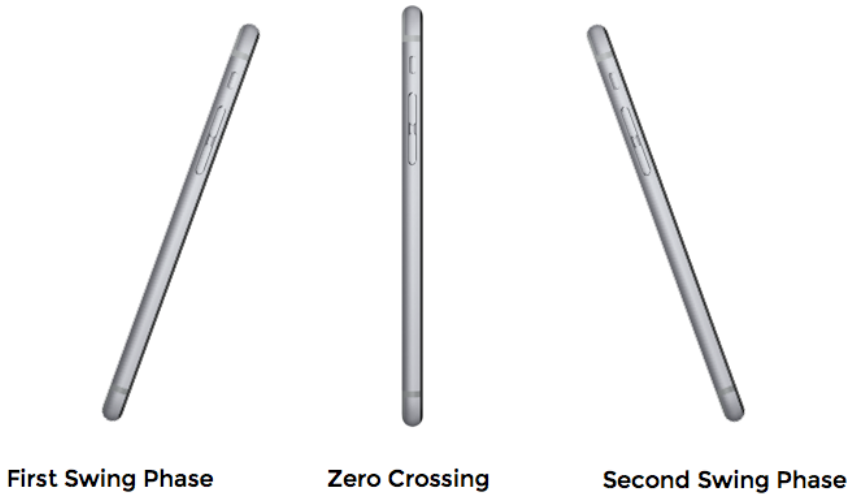

Figure 4.4: Rotation of the phone while walking)

The x-axis is the important axis, which is rotated during a step. It shows a negative amplitude in the first swing phase and crosses the zero line before the second swing phase starts, resulting in a positive amplitude. Using this observations and the explanations from Jayalath and Abhayasinghe (2013), algorithm 1 was designed.

The algorithm depends on two thresholds. The threshold deciding if the detected peak was large enough to be a valid step is applied dynamically by using an average value of all detected peak. The average is multiplied by 0,85. The detected peak has to be larger than this threshold, but at minimum 0.08. The initial threshold before averaging is 0.4. These values were collected during the tests of the algorithm by observation. These thresholds are collected individually for each swing phase of the foot, to be able to react better on arhythmic gait patterns. Additionally, a time threshold is applied. In order to count as a valid step, the duration of the step has to be longer than 0.3 seconds.

The algorithm first updates the local minimum and maximum if necessary, to be able to compare them against the thresholds later. Subsequently, the current sensor value is compared to the previous one, to check if the zero line was crossed. Whenever the zero line is crossed, the algorithm tries to detect if a valid step was performed. In order to verify this, the local minimum and maximum are compared to the thresholds. If it is within the threshold, the time difference to the previous step is calculated. If the difference is more than the time threshold, a valid step was found and the number of detected steps is updated. The data of the found step is

Algorithm 1 Step Detection

Require: $zeroCrossing = false, withinThreshold = false, withinTime = false$
  if $x_i < localMin$ **then**
    $localMin \leftarrow x_i$
  else if $x_i > localMax$ **then**
    $localMax \leftarrow x_i$
  end if
  if $x_{i-1} > 0$ **and** $x < 0$ **or** $x_{i-1} < 0$ **and** $x > 0$ **then**
    $zeroCrossing \leftarrow true$
  end if
  if $localMin < minThreshold$ **or** $localMax > maxThreshold$ **then**
    $withinThreshold \leftarrow true$
  end if
  if $timeDifferenceOfSteps >= minTimeDifference$ **then**
    $withinTime \leftarrow true$
  end if
  if $zeroCrossing = true$ **and** $withinThreshold = true$ **and** $withinTime = true$
  then
    $detectedSteps \leftarrow detectedSteps + 1$
  end if

used for further processing.

The advantage of the used algorithm is, that it allows live step detection without a training phase or the whole signal data. The implementation using the zero crossings was simple, the challenging task was to find suitable thresholds, which required a lot of testing and observing.

Step length estimation

For the prototype application the step length estimation problem was solved by using fixed step length based on the height and gender of the user, as presented in Smittle et al. (2010). If the person is male, the formula *Height * 0,415* is used, otherwise the formula *Height * 0,413* is used for retrieving the length of a single step. As seen from the research in section 3.2.4, many authors stated that a fixed step length gives sufficient accuracy.

Direction detection

As seen in section 3.2.3, detecting the direction a user is heading using relative localization systems is a complex task, because there is no absolute geographical

reference available. The magnetometer and the gyroscope deliver sensor data, that is transformable into heading information.
Core Motion and Core Location provide useful heading data. The calibrated magnetic field found in CMDeviceMotion, removes the device bias and reflects the values of the earth's magnetic field including surrounding fields. The value is provided as vector data including values of all three axes.
The magneticHeading found in Core Location not only corrects the device bias, it also filters out all local external magnetic fields. If the external fields are moving with the device, they are ignored, otherwise they are measured. Beside the elimination of possible interferences, the value retrieved from Core Location is provided in degrees. In return, the raw data of the individual axis is not available through Core Location.
As discussed before, Core Location also provides a trueHeading property, which corrects the offset gap the magnetic and geographical north pole. But in order to do this correction, GPS is required.
A simple iOS application[8] is suitable for a comparison of the raw and calibrated magnetometer data from Core Motion and the magneticField property from Core Location. Testing the application in different indoor and outdoor environments confirmed, that the values provided by Core Location are far more stable and accurate than others. Based on these findings it was decided to use the magneticHeading for further calculations.

During the implementation and testing of the prototype it became clear, that finding the correct direction the user is heading to, is the most difficult part of the application. The magnetometer is very vulnerable to external interferences. Today's indoor environments are full of electric devices creating their own magnetic fields. Thus the magnetometer was the main source of error.

4.5.5 Placing items

Users are asked to track their items using the prototype application. Due to the limitations of the dead reckoning system, it was difficult to find a proper user ex-

[8] `https://github.com/foundry/MagnetoMeter/`

perience for accomplishing this task. While using the app for tracking, users have to carry their smartphone in their front pocket. Using simple buttons for placing items was the first approach taken, but soon it was clear, that taking the phone out of the pocket and putting it back in after the item was tagged led to unexpected results. By performing these motions, often steps were detected falsely.
To counteract the problem, a speech recognition library was added to allow the users to give voice commands to the application. There are many speech recognition libraries available, most of them are commercial. It was decided to use the freely available *OpenEars*[9] iOS framework. The library supports speech recognition for the languages English and Spanish and text-to-speech for the English language. The *SpeechRecognizer* class was implemented for the prototype application to handle voice commands from the user. As shown in listing 4.6, the SpeechRecognizer supports three commands for placing a wallet, a keychain or a watch.

```
1  ...
2  NSArray *commands = @[@"wallet", @"keychain", @"watch"];
3  self.speechRecognizer = [[SpeechRecognizer alloc] initWithCommands:commands];
4  self.speechRecognizer.delegate = self;
5  ...
```

Listing 4.6: SpeechRecognizer configuration

The commands are prefixed with the command indicator "Placed". So a correct command would be "Placed keychain". Since OpenEars needs to know the dictionary of supported words for speech recognition, only these three commands will be recognized by the prototype application and every other words spoken by the user will be ignored.
Using speech recognition for the tagging of the items showed quite promising results. If the users speak slow, clear and loud the commands are recognized without problems.

4.5.6 Visualization

The visualization of the gathered information was very important to be able to verify and analyze results. As discussed in section 3.4, line charts are the favorite way to visualize sensor data. The basic idea was to have a visualization which is able

[9] `http://www.politepix.com/openears/`

to draw raw or processed sensor data along with another visualization showing the trace of the user.

After comparing different plotting libraries available for iOS, it was decided to go with a very popular framework, called *CorePlot*[10]. The framework is a highly customizable 2D plotting library for iOS, which provides many different chart types. Thus, CorePlot should provide everything that is needed for creating the visualizations.

The *ChartController* was implemented to work together with CorePlot. It is responsible for drawing the sensor and trace charts to the main window of the server application. The two chart types are both basic line charts, which are available in CorePlot by default. Internally they are two different instances of the ChartController.

The ChartController has different methods, which accept data to be plotted. The interface of the controller is shown in listing 4.7.

```
1  ...
2  typedef enum _HostedChartType {
3      HostedChartTypeLineChart,
4      HostedChartTypeTraceChart
5  }
6  HostedChartType;
7
8  @interface ChartController : NSObject<CPTPlotDataSource, CPTPlotSpaceDelegate>
9
10 @property (weak) IBOutlet CPTGraphHostingView *graphHostingView;
11 @property (nonatomic, assign) HostedChartType hostedChartType;
12
13 - (id)initWithType:(HostedChartType)hostedChartType;
14
15 - (void)placeItem:(NSString *)item withColor:(NSColor *)color;
16 - (void)addPointWithValue:(NSValue *)point isStep:(BOOL)isStep;
17 - (void)addTracePointWithValue:(NSValue *)val;
18 - (void)reset;
19
20 @end
21 ...
```

Listing 4.7: ChartController interface

The *WindowController* passes all information to the ChartController. For displaying raw sensor values, it uses the same key-path that is configurable in the SignalPro-

[10] https://github.com/core-plot/core-plot

cessor class.

The final visualization the prototype application produces is shown in figure 4.5. On the left side, the sensor data is shown, while the trace is drawn on the right half of the window.

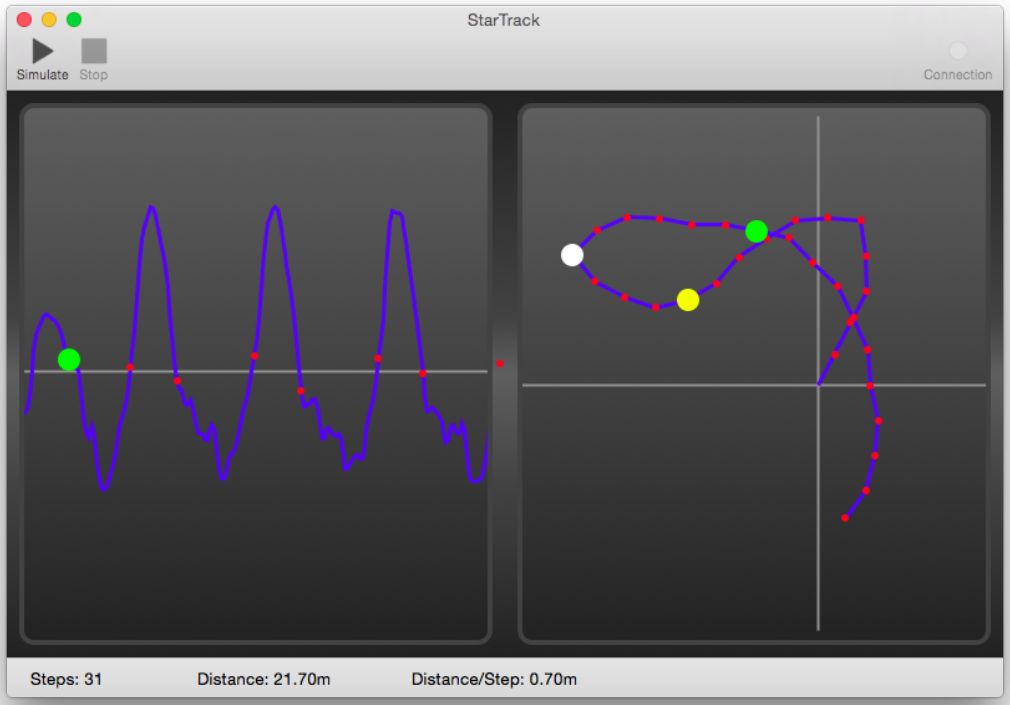

Figure 4.5: Data visualization in the prototype application

The generation of the individual visualizations and the indication of steps and placed items is explained in more detail in the following subsections.

Visualizing sensor data

The visualization of raw or processed sensor data was simple to implement using the basic line chart from CorePlot. On the x-axis, the index of the current element in the array is used, where on the y-axis the read sensor data is applied. CorePlot's charts are using the *DataSource* pattern to retrieve the necessary data from the controller.

The chart always shows a fixed number of data points in its frame. For the prototype application 200 data points were visible. Because the logging of motion events

produces a lot of data points, the chart has to scroll horizontally when new data is available. To achieve this behaviour, the plot space of the chart has to be adapted every time new data is added that is displayed. The source code that is necessary to achieve this behaviour is shown in listing 4.8.

```
1  ...
2  #define VISIBLE_POINTS 200
3  ...
4  if (self.points.count > VISIBLE_POINTS)
5  {
6    NSInteger start = self.points.count - VISIBLE_POINTS;
7    NSInteger length = VISIBLE_POINTS;
8
9    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) self.graph.defaultPlotSpace;
10   [plotSpace setXRange:[CPTPlotRange plotRangeWithLocation:CPTDecimalFromInteger
         (start)
11                 length:CPTDecimalFromInteger(length)]];
12   [plotSpace setGlobalXRange:[CPTPlotRange plotRangeWithLocation:
         CPTDecimalFromInteger(0)
13                       length:CPTDecimalFromInteger(self.points.count)]];
14 }
15 ...
```

Listing 4.8: Dynamic plot space

The charts support local and global x-axis ranges. The local range always defines the visible range of data points, whereas the global range includes all data points. After recording or simulating has stopped, it is possible to scroll horizontally to be able to analyze the complete data set.

Visualizing pedestrian traces

Drawing the trace of a user was far more challenging than visualizing sensor data. As discussed in section 3.4, a line chart is also the preferred method for generating trace visualizations.

After applying dead reckoning algorithms to the collected sensor data, the resulting data was not directly usable for drawing in visualizations. The result of these algorithms was a list of detected steps including step length and heading. So this data had to be transformed into coordinate data in order to use it in coordinate systems. The basic observation that is used for defining the algorithm is shown in figure 4.6. The cartesian coordinate system is divided into four quadrants, starting on the top

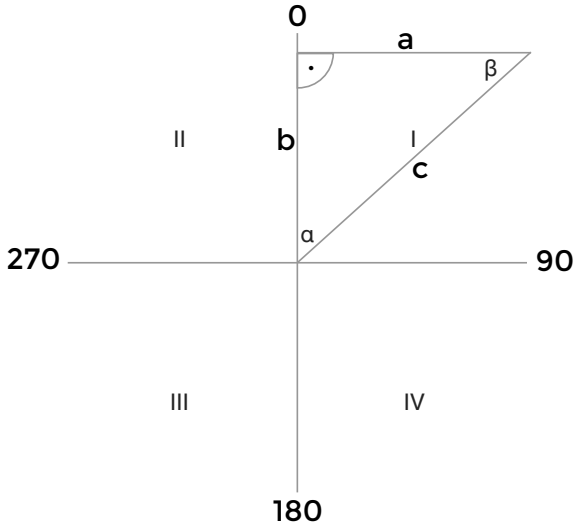

Figure 4.6: The cartesian coordinate system

right and continuing in direction against the clock. These quadrants have to be considered separately. The influence of the quadrant to the coordinate values for the x-axis and the y-axis is shown in table 4.1.

	I	II	III	IV
x-axis	positive	negative	negative	positive
y-axis	positive	positive	negative	negative

Table 4.1: Values for x-axis and y-axis in the four quadrants of the cartesian coordinate system

The length of the step is known, which represents c, or the hypotenuse of the triangle. The angle α is also known, because it represents the heading information. So the sides a and b have to be calculated to get a coordinate. The following trigonometric formulas are used to find the necessary values.

$$a = cos(\alpha * \tfrac{\Pi}{180}) * c$$
$$b = sin(\alpha * \tfrac{\Pi}{180}) * c$$

Without modification, these formulas only work for the first quadrant. For the other quadrants the heading angle has to be subtracted by the angle of the quadrant, to normalize it between 0 and 90 degrees. The values of a or b are multiplied by -1,

if the value in the quadrant has to be negative, according to table 4.1.

Taking these findings, algorithm 2 was defined for mapping step data into a coordinate system.

Algorithm 2 Mapping step data to a coordinate system

Require: $aMultiplier = 1, bMultiplier = 1$
  if $heading >= 0$ **and** $heading < 90$ **then**
    $\alpha = heading$
  else if $heading >= 270$ **and** $heading < 360$ **then**
    $\alpha \leftarrow heading - 270$
    $aMultiplier \leftarrow -1$
  else if $heading >= 180$ **and** $heading < 270$ **then**
    $\alpha \leftarrow heading - 180$
    $aMultiplier \leftarrow -1$
    $bMultiplier \leftarrow -1$
  else if $heading >= 90$ **and** $heading < 180$ **then**
    $\alpha \leftarrow heading - 90$
    $bMultiplier \leftarrow -1$
  end if
  $a \leftarrow (cos(\alpha * \Pi/180) * stepLength) * aMultiplier$
  $b \leftarrow (sin(\alpha * \Pi/180) * stepLength) * aMultiplier$
  $coordinate \leftarrow coordinate(a, b)$

The result of the algorithm is a coordinate with x-axis and y-axis values, which is easily drawable using a basic line chart. The initial position is always the origin of the cartesian coordinate system, a coordinate with the value 0 for the x-axis and the y-axis. Every other coordinate received by the mapping algorithm is always drawn relative to the previous coordinate. This means the values of the x-axis and the y-axis of the current coordinate are added to the values of the previous coordinate. This is a typical behaviour for dead reckoning, which indicates the relative localization.
Similar to the sensor visualization, the plot space of the chart was adapted to fit the whole trace. But unlike the sensor chart, this chart only uses the local plot ranges because the plot space is expanded if the minimum or maximum of an axis changes. There is no scrolling in the result, because the whole trace is always visible.

Visualizing steps

For better analysis, the location where a step was detected was highlighted in the sensor and the trace visualization. CorePlot supports *annotations*, which were used to implement this feature. Whenever a zero crossing was detected and the step detection algorithm has found a valid step, an annotation is added. Listing 4.9 shows the configuration of an annotation and how they are added to the chart.

```
1  ...
2  CPTBorderedLayer *stepLayer = [[CPTBorderedLayer alloc] initWithFrame:CGRectMake
       (0, 0, radius, radius)];
3  stepLayer.cornerRadius = radius/2;
4  CPTFill *fill = [CPTFill fillWithColor:color];
5  stepLayer.fill = fill;
6
7  NSNumber *x = @(point.pointValue.x);
8  NSNumber *y = @(point.pointValue.y);
9  CPTPlotSpaceAnnotation *annotation = [[CPTPlotSpaceAnnotation alloc]
       initWithPlotSpace:self.graph.defaultPlotSpace anchorPlotPoint:@[x,y]];
10 annotation.contentLayer = stepLayer;
11 [self.graph addAnnotation:annotation];
12 ...
```

Listing 4.9: Annotations in CorePlot

The annotations for the steps were added in both visualizations.

Visualizing item positions

Similar to the visualization of a step in the charts, the location of the items are added as annotations too. Whenever the user placed an item, a circle in a specific color for the item is drawn in both charts. A legend is used to map the colors to the identifier of the placed item.

5. Field Experiment

To be able to verify and analyze the methods and algorithms implemented in the prototype application, a field experiment was conducted. The experiment was performed by testing the application with twelve participants of different gender, age and height. The testers were asked to walk a predefined and measured track and place one or more items in different locations. After the walk, they were asked to retrieve the position from the placed items from the trace visualization and give feedback.
This chapter is dedicated to presenting the methodology and execution of the field experiment that was conducted with the goal of measuring the performance of the prototype application in terms of acceptance, usability and accuracy. The results of this evaluation are presented in chapter 6.

5.1 Goals

The field experiment was conducted to collect data to fulfill two different goals. First, the performance and accuracy of the indoor positioning of the application should be evaluated. For that purpose, data collected during the usage of the application is compared to reference data that is collected using other methods. The results are gathered using mathematical methods.
The second goal is to be able to evaluate the application in terms of Human Computer Interaction. The data is collected through user feedback and observation how people are using the application. The fulfillment of the second goal strongly depends on the first goal, because if the application is inaccurate users are not able to work properly with the application.
The second goal also strongly relates on the use cases defined in section 4.1.2, be-

cause it evaluates if users are able to tag the positions of the lost items and if they are able to recognize these positions later.

5.2 Experiment setup

The experiment was performed by testing users sequentially on a single day using the prototype application described in chapter 4. The used hardware setup and the selected test track are explained in more detail in the following sections.

5.2.1 Hardware setup

The hardware used for the experiment was an Apple MacBook Air 13" with Mac OS X Yosemite 10.10.1 installed, which hosted the server application. The client application was installed on an Apple iPhone 6 running iOS version 8.1.2. An Apple Airport Extreme Base Station was used for providing the WiFi connection client and server used to communicate with each other.

5.2.2 Test track

It was decided to select a test track with a distance someone would usually cover when walking inside a building. Tape was used for marking the test track, so users could follow it easily. The distance was measured on site using a measuring tape. The measurements were verified using Google Maps' feature to measure distances and GPS measurements with the app *Runtastic*. The measured distance of the test track was approximately 45 meters.
On different locations, small tables were provided were the users could place their items. The track was defined as a lap to test if the starting location matches the end location, to be able to analyze the accuracy of the application.
The layout of the test track is illustrated in figure 5.1, including the possible locations where the users could place their items.

The track was located on flat surface and didn't include any obstacles as stairs or doors users had to open.

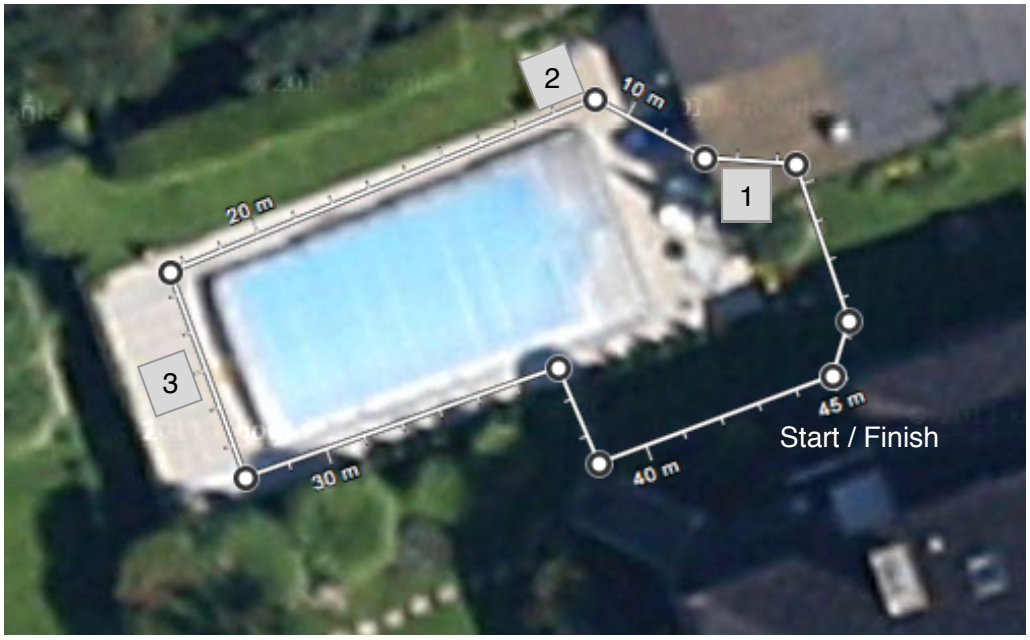

Figure 5.1: Layout of the test track

5.3 Test participants

Twelve test participants were recruited for the field experiment. The fact that different people of different height, age and gender produce different gait patterns was respected when choosing the test users. Six male and six female users of different age groups participated in the experiment.

Especially the performance of the application with elderly people was of great importance, because they often have asymmetric gait patterns through to different disorders. Five of the test users were at the age of 60 years or older, three of them where around 80 years.

Table 5.1 shows the demographic data of all twelve test participants.

The *ID* column applies a unique identifier to each test participants in order to refer to this person in later sections.

5.4 Walking test

The walking test was performed in order to evaluate the accuracy of the implemented dead reckoning system. Users were instructed to walk at normal and steady speed along the marked path and place one or more items during the test.

ID	Gender	Age	Height
P01	male	65y	1,86m
P02	male	31y	1,87m
P03	female	84y	1,60m
P04	female	63y	1,69m
P05	male	23y	1,73m
P06	male	51y	1,78m
P07	male	26y	1,82m
P08	female	27y	1,62m
P09	female	79y	1,64m
P10	male	80y	1,72m
P11	female	55y	1,65m
P12	male	56y	1,77m

Table 5.1: Demographical data of the test users

In the case of technical interruptions, like network errors or phone calls, or a misunderstanding of a test person in the test procedure the current test was restarted.

5.4.1 Prerequisites

Users were first asked to provide information about their age and body height, so the server application could be pre-configured in order to provide more accurate step length information, which results in a better trace visualization.
All test users were asked to start their walk at a fixed location, which was marked as start. In order to best support the step detection algorithm, the smartphone was mounted in the front pocket of the trousers and the display was heading towards the test user. The client application was the active application and it was connected to the server. The users were instructed how to use voice commands for placing the items. Subsequently, each tester received a wallet, a keychain and a watch as items they could place during the test.

5.4.2 Procedure

After the recording of the sensor data was initiated by the test supervisor, the participants were allowed to start their walk. Additionally they were asked to count the number of steps while walking, and tell the result to the supervisor after the end location was reached. To ensure that the number of steps taken was correct,

a second person, which was observing the experiment, was also asked to count the steps of the currently walking test person.
After the users had ended their test lap, the recording of the sensor data was stopped and the gathered data was saved to the file system to be able to analyze it with the simulator later. The number of the counted steps was entered in the test protocol, to compare them with the number of detected steps later.
At this point the walking test was finished and the test user was requested to view the results and give feedback in an interview.
The results of the walking test are presented in chapter 6.

5.5 Interview

The interview was performed immediately after the participant had successfully finished the walking test. The users were offered to examine the results delivered by the server application in form of visualizations. Beside the personal gait pattern they were able to see the trace of their walk together with the number of detected steps, the calculated total distance that was covered and the measured duration they required for their lap.
Every user was asked to answer eight questions about the trace visualization and the prototype application in general. For easier evaluation, the questions were formulated similar to a multiple choice test, asking participants to answer the questions by choosing *yes*, *rather yes*, *rather no* and *no*.

1. Do you use a smartphone?
2. Do you use smartphone applications in your daily routine?
3. Do you often waste time searching lost items?
4. Do you use an existing app for retrieving lost items?
5. Would you use this application?
6. Does the trace visualized in the prototype application reflect the test track?

7. Were you able to recognize the position of the placed items from the visualization of the trace?

8. Do you think the application could be helpful for locating lost items?

For easier understanding, the questions were asked in German language. The participants of the interview were also asked to provide general feedback and ideas for improving the application. The results of the interviews are presented together with the results of the walking test in chapter 6.

6. Results

This chapter presents the results that were derived from the field experiment explained in the previous chapter. First, the performance and accuracy of the prototype application is analyzed by evaluating the walking test.
The second part is dedicated to analyzing the feedback of the test participants retrieved through the interview, in terms of acceptance and usability of the prototype application.

6.1 Reference data

This reference data was gathered while conducting the walking test. It refers to the actually number of steps taken, the actual distance covered and the measured time participants needed to complete the test lap. The data was collected with the following methods:

- The total number of steps taken was counted by the participant and a second person observing the experiment.
- The total distance covered was derived from measuring the test track with a measuring tape and Google Maps.
- The time users needed to complete the track was retrieved by calculating the time difference between the start and end of the walk.

Additionally, the number of steps per second and the average step length was calculated in order to compare it with the results derived from the dead reckoning module of the prototype application.
The data gathered and calculated individually for each participant is shown in table

6.1. The unique ID is the same as in table 5.1 in order to retrieve the demographic data of the person. The same unique ID will be used in the whole chapter.

ID	Total Distance	Total Steps	Total Time	Steps/s	Avg. Step Length
P01	45m	57	35,44s	1,61	0,79m
P02	45m	56	36,98s	1,51	0,80m
P03	45m	96	59,04s	1,63	0,47m
P04	45m	68	37,70s	1,80	0,66m
P05	45m	63	35,58s	1,77	0,71m
P06	45m	54	33,64s	1,61	0,83m
P07	45m	52	31,34s	1,66	0,87m
P08	45m	63	34,68s	1,82	0,71m
P09	45m	66	43,66s	1,51	0,68m
P10	45m	74	49,14s	1,51	0,61m
P11	45m	60	35,64s	1,68	0,75m
P12	45m	47	28,12s	1,67	0,96m

Table 6.1: Reference data

As shown in table 6.1, the values of the total steps taken and the total time are varying strongly between different participants.
Elderly people obviously require a significant longer time to complete the track and tend to take shorter steps. Participant P12 was able to complete the track with 47 steps. That is less than the half of P03, who required 96 steps to walk the same distance.

The calculation of the steps per second produces similar results for all participants. According to the results, people tend to perform larger steps if they want to increase their walking speed rather than increasing the step frequency.

The number of steps taken is related to the height of a person. Large people need to perform fewer steps, because they are able to perform larger steps, which is shown in the calculated average step length of each participant.

6.2 Step detection

To analyze the accuracy of the step detection algorithm explained in section 4.5.4, the number of steps detected by the prototype application was compared to the number of steps that were counted by the participants during the walking test. The results are shown in table 6.3.

ID	Total Steps	Detected Steps	Deviation	Error	Accuracy
P01	57	58	1	1,72%	98,28%
P02	56	56	0	0%	100%
P03	96	94	2	2,04%	97,96%
P04	68	70	2	2,86%	97,14%
P05	63	64	1	1,56%	98,44%
P06	54	55	1	1,82%	98,18%
P07	52	53	1	1,89%	98,11%
P08	63	62	1	1,61%	98,39%
P09	66	64	2	3,13%	96,88%
P10	74	79	5	6,33%	93,67%
P11	60	60	0	0%	100%
P12	47	49	2	4,08%	95,92%
Sum	756	768	18	-	-
Avg	63	64	1,50	2,25%	97,75%
SD	12,78	13,36	1,31	1,73%	1,73%

Table 6.2: Accuracy of the step detection algorithm

As the results indicate, the algorithm gave very satisfying results for all of the twelve test participants. The overall accuracy was 97,75%, with only 18 steps that were detected falsely from a total of 768 detected steps. The highest error value for an individual participant was 6,33%. The dynamic threshold generation was a key feature for achieving this high accuracy, because the peaks in the gait patterns of the participants were very different. Adapting the thresholds for minimum movements and the time threshold could slightly influence the results. Small changes on the time threshold improved results for some participants, but decreased the accuracy for others. The threshold for 0,3 seconds as minimum time for a detected steps was discovered to work suitable for all data sets.

The analyzation of the individual gait patterns revealed, that the patterns differ strongly between participants. Particularly elderly people produce very arhythmic

gait patterns, which is often a result of a disease because they put more weight onto one leg. As the results show, the algorithm has slightly decreased accuracy for elderly people (P03, P09, P10). Figure 6.1 shows the differences of the gait pattern for the oldest and youngest participants (P03 and P05). The coordinate system is the same for both signals.

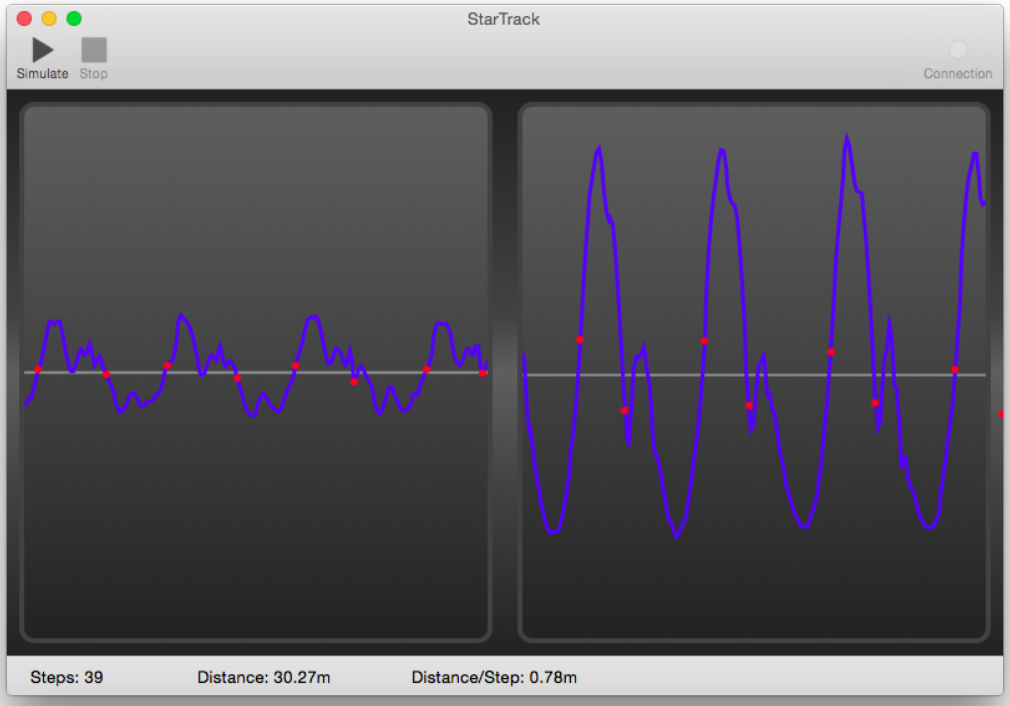

Figure 6.1: Differences in the gait patterns of the oldest and youngest participants

As the comparison of the two signal curves indicates, the swing motion the leg produces in each phase of the step is much more pronounced for younger people. For the step detection algorithm it is more difficult to differentiate the smaller peaks from unintentional movements.

The circumstance that participant P10 had the highest error rate could correlate with the fact, that this participant was the only one wearing suit trousers, which are fitted wider than the jeans or leggings the other participants were wearing. Thus the smartphone was not that fixed as for other participants, which could lead to an imprecise gait pattern.

A common error found during the signal analyzation, was that participants tend to perform unusual movements while they are placing items. Figure 6.2 shows two examples of this particular situation.

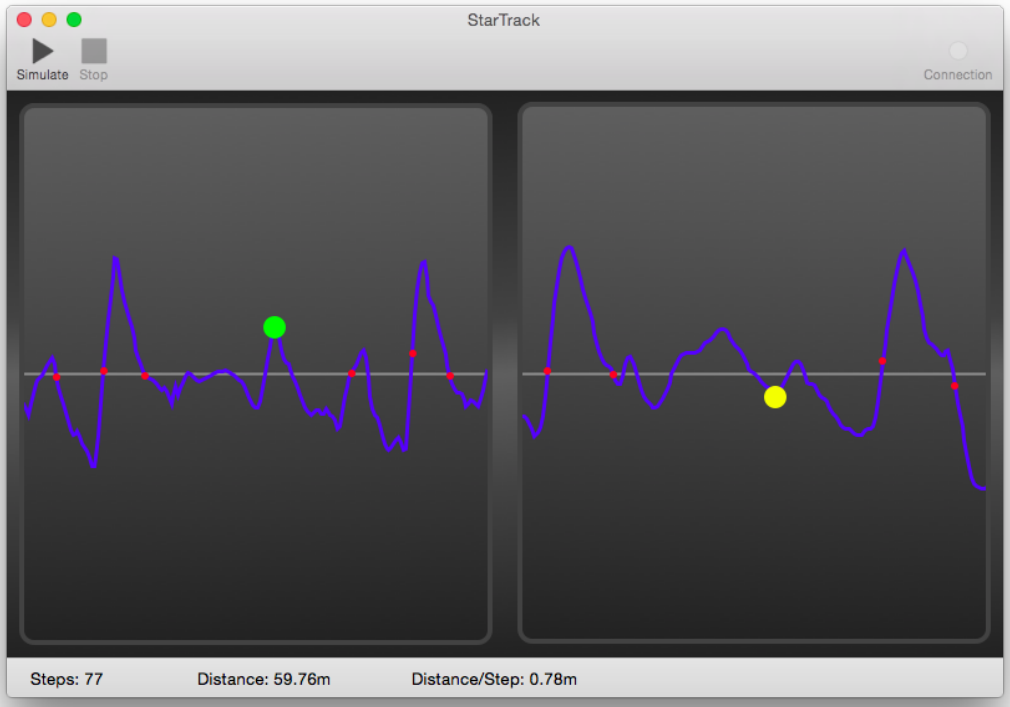

Figure 6.2: Unsual movements while placing an item

As seen in the signal, the rhythmic step pattern is broken while the movement is performed to place the item. Therefore the step detection algorithm failed to detect a step or falsely detected a step.

6.3 Step length estimation

As explained in section 4.5.4, the calculation of the length of single steps is correlated to the gender and the height of a participant. If the person is male, the formula *Height * 0,415* is used, otherwise the formula *Height * 0,413* is used.
The formulas are used to provide a rough estimation of the step length rather than calculating them based on the sensor readings. The results when the formula is

applied to the individual test participants is presented in table 6.3.

ID	Avg. Step Length	Calc. Step Length	Abs. Deviation
P01	0,79m	0,77m	0,02m
P02	0,80m	0,78m	0,02m
P03	0,47m	0,66m	0,19m
P04	0,66m	0,70m	0,04m
P05	0,71m	0,72m	0,01m
P06	0,83m	0,74m	0,09m
P07	0,87m	0,76m	0,11m
P08	0,71m	0,67m	0,04m
P09	0,68m	0,68m	0,00m
P10	0,61m	0,71m	0,10m
P11	0,75m	0,68m	0,07m
P12	0,96m	0,73m	0,22m
Avg	0,74m	0,72m	0,08m
SD	0,13m	0,04m	0,07m

Table 6.3: Step Length Deviation

The formulas seem to provide reasonable accuracy if all conditions are met. If the participants take shorter or langer steps than expected, the fixed formula is not able to react and thus delivers wrong results. For participants P03, P07, P10 and P12 the deviation is equal ore greater than 0,10m per step. For participants P01, P02, P05 and P09 the formulas seem to work well because the deviation is below 0,08m per step. Taking the average over all participants, the formulas are producing a deviation of 0,02m. The calculated step length is slightly too short compared to the actual step length. Table 6.4 shows the impact of the deviations from the single steps on the total distance.

The total deviation is excessively high for participants P03 and P12, having differences of 18,44m and 10,48m. These participants completed the track with the fastest and slowest time, respectively. So the success of the step estimation formulas is correlated to the walking speed. If the pedestrian is walking at normal pace, the step length estimation formulas give reasonable results.
The average deviation of all participants is 5,01m. An interesting observation is, that the deviation of the average of the calculated distances compared to the actual total distance is only 0,18m.

ID	Total Distance	Calc. Distance	Abs. Deviation
P01	45,00m	44,00m	1,00m
P02	45,00m	43,46m	1,54m
P03	45,00m	63,44m	18,44m
P04	45,00m	47,46m	2,46m
P05	45,00m	45,23m	0,23m
P06	45,00m	39,89m	5,11m
P07	45,00m	39,28m	5,72m
P08	45,00m	42,15m	2,85m
P09	45,00m	44,70m	0,30m
P10	45,00m	52,82m	7,82m
P11	45,00m	40,89m	4,11m
P12	45,00m	34,52m	10,48m
Sum	540,00m	537,84m	60,06m
Avg	45,00m	44,82m	5,01m
SD	0,00m	7,41m	5,25m

Table 6.4: Total Distance Deviation

6.4 Direction detection

Verifying that the detected directions are correct is the most difficult task, because there are no actual values to compare against. Two methods were chosen for analyzing the results. First, all traces of the test participants are compared using a single visualization, to see if the results are similar. The second method compares the first and last location of the trace. Because the test lap starts and end at the same position, these positions should overlap in the visualization too.
Figure 6.3 shows all traces of the participants, that were drawn during the walking test by the prototype application, together with the shape of the test track.

As seen in the visualization, the traces differ strongly. There are two circumstances responsible for the deviations. First, the step length calculation produces an error, as previously presented in section 6.3. For participants P01, P02, P03, P04, P05, P06, P07 and P12 the generated traces are similar. For P03, the trace is much larger, because of the falsely calculated total distance of 63,44m.
For P08, P09, P10 and P11 the heading was different than for other participants. During the test it was verified before each test lap of a participant, that the device was mounted the same way. The orientation of the device, that is logged with the sensor data is also equal for all participants. So the different heading has to be the

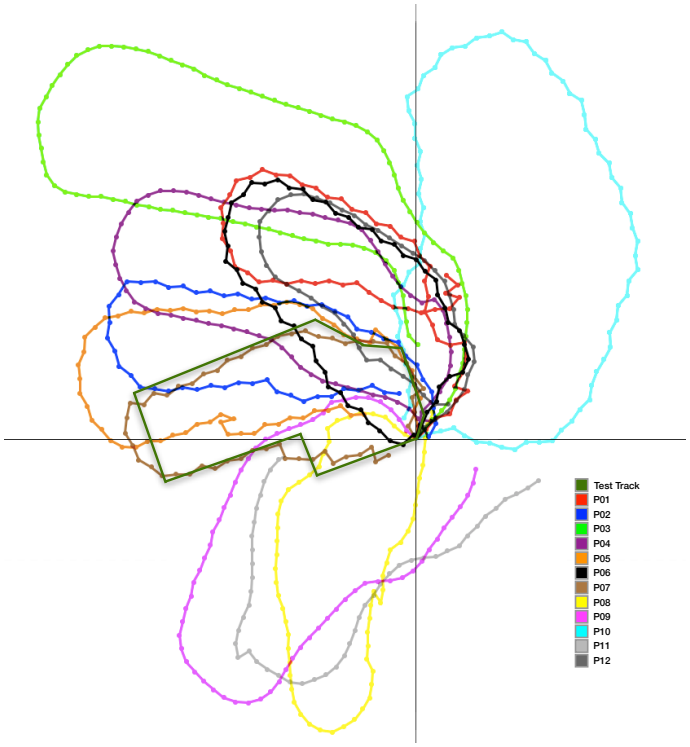

Figure 6.3: Visualized traces of all test participants

result of a miscalibration of the magnetometer.
In general the shapes of the generated traces reveal the same characteristics as the test track seen in figure 5.1 and test participants were able to recognize the test track from the visualization.

Table 6.5 shows the deviation from the initial location, or the start of the test lap, which was the origin of the cartesian coordinate system and the last location, which was recorded after the last step was taken.
As the scale of the coordinate system is one meter, the distance is retrieved directly from the coordinates without converting.

The distance between the start and the end coordinate was calculated using the following formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

As seen from the results, the deviation calculated between the start and end lo-

ID	Start Coordinate	End Coordinate	Deviation
P01	0; 0	1,66; 8,05	8,22m
P02	0; 0	-0,92; 2,23	2,41m
P03	0; 0	0,09; 4,63	4,63m
P04	0; 0	0,84; 0,90	1,23m
P05	0; 0	-2,80; 1,23	3,06m
P06	0; 0	-0,12; 0,24	0,27m
P07	0; 0	-1,47; -0,76	1,66m
P08	0; 0	0,93; 1,35	1,63m
P09	0; 0	3,26; -1,45	3,57m
P10	0; 0	-0,64; 1,05	1,23m
P11	0; 0	6,66; -1,98	6,95m
P12	0; 0	1,72; 1,72	2,44m
Sum	-	-	37,31m
Avg	-	-	3,11m
SD	-	-	2,41m

Table 6.5: Total Distance Deviation

cation is not related to any of the previous results. P03 and P12, which had the highest deviation concerning total distance, do not reveal unusual results now. The highest deviations were calculated for P01 and P11. P06 was the only user, where the distance between the two positions was below one meter. The average deviation was 3,11m, which is 6,91% of the total length of the test track.

The main error sources, that are responsible for the deviation between the coordinates were the inaccuracy of the magnetometer and erratic step lengths of the participants. People tend to perform shorter steps while performing a turn, so an error is produced because only a fixed step length is concerned when calculating the trace.

6.5 Locating lost items

To verify if the prototype application is suitable for its dedicated use case, an interview was conducted, as explained in section 5.5.
The participants were presented a visualization of their trace, where the position of the placed items was indicated using a bigger circle in a different color. According to the test participants, the highlighting of the placed items worked well and the items were located in the position in the trace, where the location of the tables on the test track could be estimated.
Figure 6.4 shows the prototype displaying the resulting visualization of the walking test of participant P06, as it was visible for the test participants.

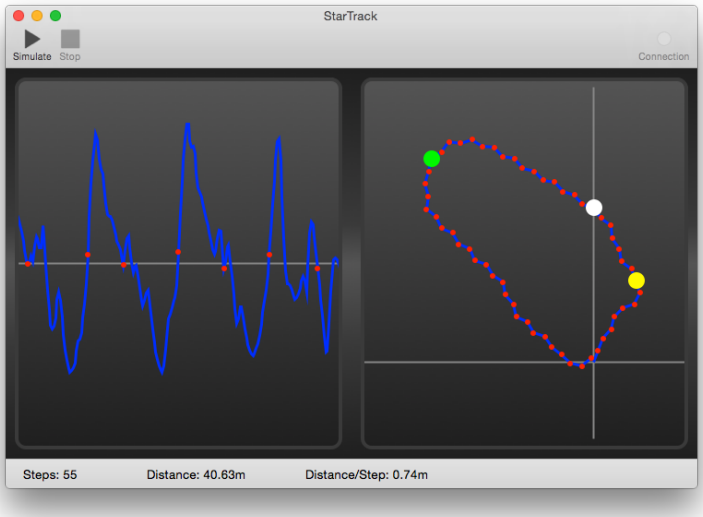

Figure 6.4: Item positions highlighted in the visualization

Figure 6.5 uses the same visualization, but this time the generated trace is mapped to the static image of the test track. The visualization of the trace was rotated, so that it matches the orientation of the test track as it appears on Google Maps. As seen in the overlay, the actual position of the tables on the test track is very close to the highlighted positions of the placed items in the walking test.

The questions from the interview were evaluated in order to be able to make a statement if the application is valuable for users. Table 6.6 shows the frequencies of the four possible answers that were given by the participants.

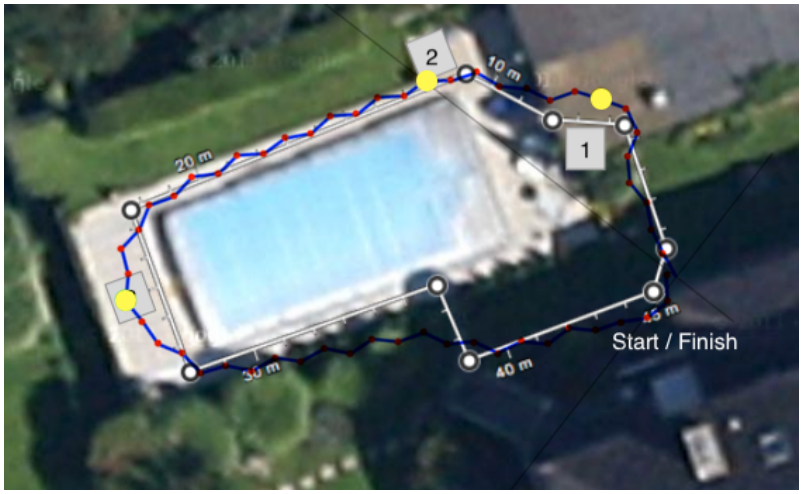

Figure 6.5: Positions of the items mapped to the test track

Question	Yes	Rather Yes	Rather No	No
1. Do you use a smartphone?	9	0	0	3
2. Do you use smartphone applications in your daily routine?	6	2	1	3
3. Do you often waste time searching lost items?	7	4	1	0
4. Do you use existing apps for retrieving lost items?	0	1	1	10
5. Would you use this application?	1	3	5	3
6. Does the trace visualized in the prototype application reflect the test track?	5	6	1	0
7. Were you able to recognize the position of the placed items from the visualization of the trace?	4	4	3	1
8. Do you think the application could be helpful for locating lost items?	4	7	1	0

Table 6.6: Results of the interview

For easier understanding, the distribution of the answers for each question is illustrated in figure 6.6.

Interpreting the results, it shows that especially the elderly generation is not used to smartphones. The three oldest participants do not own a smartphone, thus they are not using any apps to support their daily routine. On the other hand, just one

person, who owns a smartphone is not regularly using apps.
Seven people stated that they often face the problem that they have lost personal items. Four people occasionally find themselves in this situation. Only one out of twelve participant to not have the problem of loosing items.
Only two participants are using apps for retrieving items. Both are using the app *Find my iPhone* from Apple, which is available for free. None of the participants uses third party apps to get support in these situations.
Only four people answered that they would use this kind of application for retrieving lost items. The reasons for that is that they are not using smartphones, have privacy concerns, do not want an application to always require motion sensor readings through to battery life or simply don't want to carry the device with them while they are at home.
Comparing the visualized trace to the real test track, eleven out of twelve participants said that the visualization does reflect the test track and they would recognize it. Eight users were able to easily identify the position of their placed items from the visualization. In addition, they stated that mappings to floor plans or maps as well as corresponding icons for the items would be helpful.
In general, eleven people think that the concept of the application could be valuable for supporting people to retrieve their lost items.

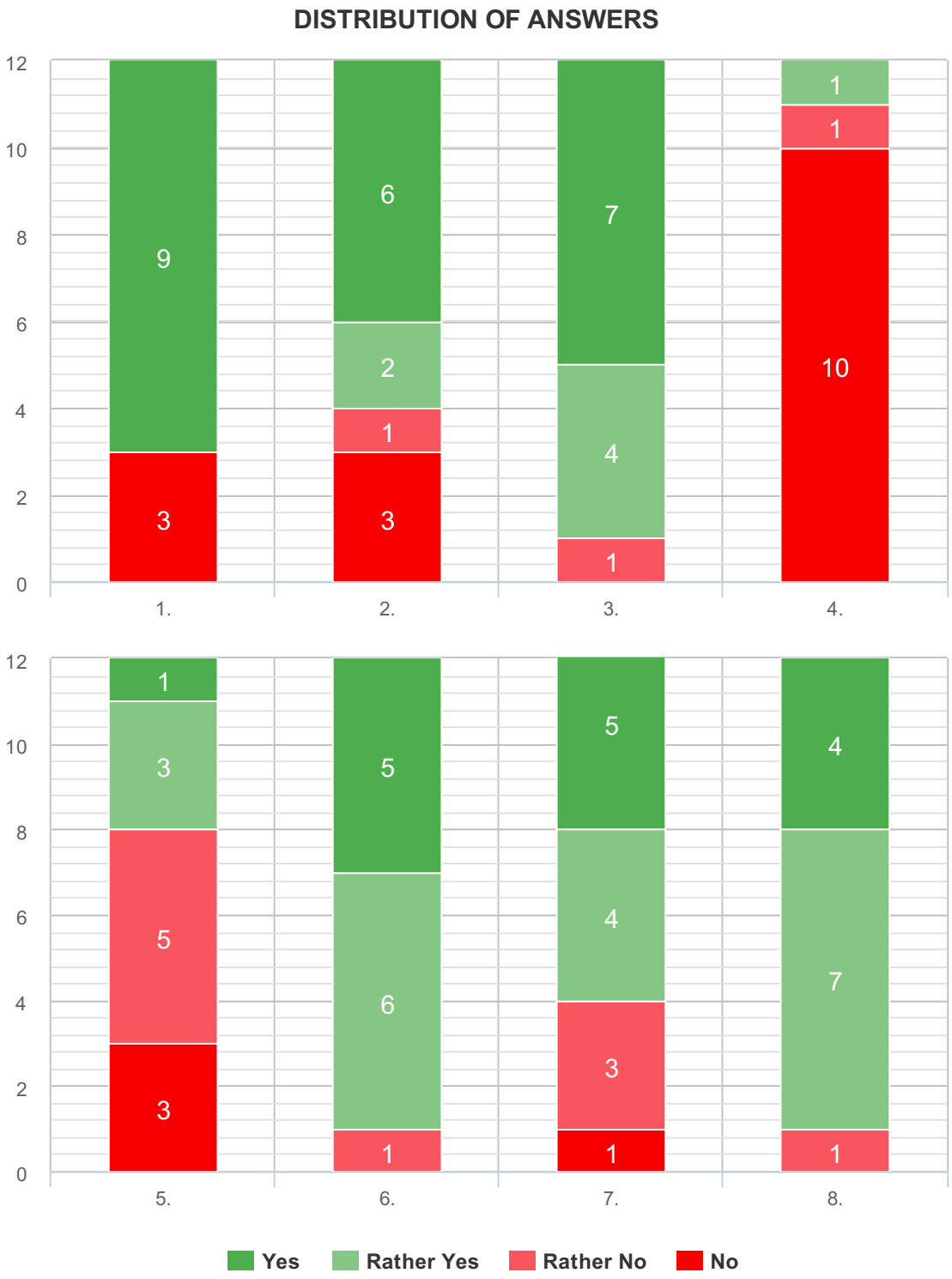

Figure 6.6: Results of the interview

7. Discussion and Lessons Learned

The results of the field experiment show that the implemented prototype application was able to fulfil its primary use case, to help users to retrieve the location of their lost items. The application only uses sensors built into common smartphones without the need of additional infrastructure.

The implemented step detection algorithm showed an overall accuracy of 97,75%, which is similar to the results of the algorithms presented in the related work in section 3.2.2. The decision to use the gyroscope as primary sensor for applying step detection was supported through the high accuracy. The results of the step length estimation in section 6.3 revealed, that there is much possibility for improvements. Due to the fixed step length based on gender and height of the participant, the deviations between actual and calculated step lengths and total distance were very high in some instances. The results showed a strong variance if the walking speed was very slow or fast, i.e. when participants performed much shorter or longer steps as usual for their body height. As Steinhoff and Schiele (2010) stated, the angular error produced by the inaccuracy of the magnetometer due to the influence of external magnetic fields is higher than the error produced by wrong step lengths.
As the results in section 6.4 showed, the implemented dead reckoning system was very promising, but had to fight the same issues regarding direction detection. The readings of the magnetometer, which are already corrected with the help of the gyroscope and averaged to avoid unintentional peaks, showed strongly varying results for the individual participants. Thus the deviation between start and end position of the test track was very high in some instances. To minimize the heading error, it would help to use absolute localization methods in addition to dead reckoning, as suggested by Baranski et al. (2009) and Kao (1991). These absolute methods like

GPS outdoors or WiFi indoors, could help to correct the position from time to time, so that errors are not accumulated.
For the purpose of the system, to act as prototype to verify it supports users to retrieve lost items, the accuracy of the dead reckoning system was sufficient. The visualizations of the traces, presented in section 6.5, could reflect the characteristics of the actual test track as defined in section 5.2.2. Probably, the system would not be suitable for long term tests, because of the error accumulation of dead reckoning systems. So having more complex test tracks and longer evaluations would decrease the accuracy of the application.

The feedback of the users was derived through an interview. The evaluation of the answers in section 6.5 revealed that most of the people face the problem of losing personal items regularly, so apps addressing this problem have a target group. None of the participants was using third party apps as presented in section 3.5. These apps provide a large feature set, but most of them require additional infrastructure like Bluetooth markers. These markers are rather expensive, costing more than 20 USD per marker. Due to the fact that a number of markers would be required to tag all of the personal items, such a solution quickly becomes expensive. An app that could fulfill the use case without additional infrastructure would be an interesting alternative to existing apps, as the distribution of question five shows, where at least four participants stated that they would be interested to try this kind of application. The downside of not using additional infrastructure is, beside accuracy, that people always have to carry their smartphone mounted in their front pocket in order to record their current location and to be able to tag the positions of their personal items to retrieve them if necessary.

Concerning the usability of the application, most of the participants were able to recognize the test track and discover the position of their placed items within the trace. Some of the participants mentioned, that discovering the position of the items would be easier if the application would use maps or floor plans as background image of the visualization, to have a visual mapping to real world objects. At this point it is important to state the the results are strongly depending on the test track and the duration of the application usage. If the application is used for a whole day in a

more complex environment, like a large building with multiple rooms, the results are certainly worse. The current prototype application would not be sufficient for such situations and users would need a features that navigates them back to the lost item or a floor mapping rather than only looking at trace visualizations, which would be large and confusing Therefore, these features would need to be implemented in order to transform the prototype application into a product.

In general most of the participants stated that they think that this application could be useful for retrieving lost items. The level of interest of the individual users during the walking test and the interview was very high and most of them were interested and able to use the trace visualization.

8. Conclusions

This thesis was concerned with the question, if it is possible to use an indoor localization system for retrieving data about the position of misplaced items. A requirement for this application was, that it uses no external infrastructure for the localization task, so only sensors build into modern smartphones were allowed.

The research of the theoretical background, as presented in chapter 2, showed that different technologies are suitable for the development of an indoor localization system, but due to the requirements, only inertial and optical localization methods were found relevant for this work. The research on this topics was done in more detail and several approaches were presented in chapter 3.
Finally, it was decided to develop a pedestrian dead reckoning system to perform the task of indoor localization. One requirement of the application was, that it performs live tracking of pedestrians. Optical methods require a learning phase, as explained in Werner et al. (2011), where reference images are taken to be able to compare them to actual images later. Inertial methods allow to perform live tracking using step based methods.

Based on the research done in section 3, a prototype application was designed and implemented. The system was designed using client-server-architecture, using an iOS app as mobile client which reads sensor data, and a Mac OS X server application, which was used to do the signal processing and the visualizations for the users. The client-server-architecture approach was chosen because it allows easier testing and verification of the prototype.
The server application included the dead reckoning system, which contained a step detection algorithm based on gyroscope readings as presented in Jayalath and Ab-

hayasinghe (2013). The step length estimation was done using a formula based on the gender and the height of the user. The direction detection was done through magnetometer and gyroscope readings.

To visualize the trace of the user, an algorithm was implemented to transform the step data into coordinates, which were mapped to a cartesian coordinate system. Visualizations were made for the gait pattern and the trace of a user. To allow users to tag the position of personal items, speech commands were used. The positions of these items were highlighted in the visualization.

The results presented in chapter 6 were collected by the execution of a field experiment, as described in chapter 5. The evaluation was conducted by asking twelve participants of different ages, gender and height to perform a walking test and an interview. The walking test included the completion of a 45 meters test lap wile using the prototype application. During their walk, users were asked to place three personal items in different locations on the test track. In the interview, the participants provided feedback about the accuracy of the vizualizations of their walking test presented by the server application, before they were asked to answer a questionnaire consisting of eight multiple choice questions.

The performance of the dead reckoning system as well as the feedback of the test participants were very promising. Particularly the results of the step detection algorithm presented in 6.2 were very satisfying, showing an overall accuracy of 97,75%. The step length estimation based on the height and gender of the user and the direction detection showed room for improvements. The deviations of the average step length and the corresponding total distance as well as the deviations of the start and end location of the test track were higher than expected. As indicated in Steinhoff and Schiele (2010), it was confirmed by the results that the direction detection using the magnetometer readings is a source of inaccuracies of indoor localization systems. The magnetometer is very vulnerable against magnetic interferences, which are very common in indoor environments or urban areas. The disadvantage of dead reckoning systems is that errors accumulate over time. In order to increase the accuracy, a dead reckoning system should be used as addition to absolute localization systems,

to minimize the errors, as suggested by Baranski et al. (2009) and Kao (1991).
The results of the dead reckoning algorithms are dependent on the test track, the local environment and the duration of the experiment. The system was only verified within this particular setup, so a general statement on the accuracy and performance of the dead reckoning system is not possible with the executed field experiment. In addition, the results were only compared to reference data, that was gathered by measurements, so it is very likely that the reference data will include minor errors too.

The feedback of the participants, which was collected through the interview was mainly positive and motivating, as the results in section 6.5 showed. Due to the fact that only twelve people participated in the field experiment, the results are not very convincing. The results are not expected to be the same for a larger number of participants, especially because not all possible target groups were covered within the twelve participants.

The prototype application was able to satisfy most of the requirements defined in section 4.1.1, with some limitations.

- *The purpose of the application is to verify that it is possible to track and retrieve the position of different items solely using data collected from inertial sensors.*
  In general, the application could fulfill that requirement. Again it is important to state, the the implemented prototype would possibly not work so well in a different environment.

- *The dead reckoning system should work with proper accuracy to allow the retrieval of lost items by users.*
  While the step detection part of the dead reckoning system was working well, the step length estimation and direction detection was not as accurate as expected.

- *The application is very easy to use and should not disturb users in their daily routine.*
  While the application itself does not disturb users, the downside is that they always have to carry their device with them.

- *The tracing algorithms are expected to perform live tracking without mandatory training phases.*
  The system was able to provide live information of the position of personal items for the users, without the support of any external infrastructure, but in the current application the data is not always visible for users because of the client-server architecture.

- *The visualizations should be easily understandable and clearly indicate the covered route as well as the positions of the placed items.*
  For the chosen test track that requirement could be fulfilled. For larger and more complex test tracks, a mapping to floor plans would be necessary.

Beside the requirements, the uses cases defined in section 4.1.2 were covered by the prototype application.

- *The user is able to tag the location of a personal item, when it is put down somewhere.*
  Users were able to tag the position of the items using speech commands.

- *The user is able to retrieve the location of a personal item, if it is lost.*
  Users were able to retrieve the location by looking at the generated visualizations. For the current prototype, the limitation is that users have to use the server part of the application to see the visualizations.

In conclusion, the implemented application is a valuable research prototype and a solid basis for future work. As expected, it has not reached the level of a product ready for sale. The known limitations of dead reckoning systems in terms of direction detection, step length estimation and error accumulation would very likely not allow to implement a product solely based on dead reckoning systems.

While the hypothesis of this work, if an application is able to retrieve the position of lost items solely using sensor built into smartphones, could be verified for a research prototype, the hypothesis would not hold for a market-level product. The long-term accuracy would be too low without the usage of absolute localization systems or other external infrastructure.

9. Outlook and Future Work

As presented in chapter 6, the developed prototype showed satisfying results. The participants of the field experiment said that the application was helpful for retrieving lost items. The dead reckoning system also showed reasonable results in terms of performance and accuracy.
The collected results through the walking test and the interview showed chances for improvements and additional features.

The dead reckoning system showed high accuracy for the step detection algorithm, but the step length estimation and direction detection could be improved in future versions of the application. At the moment a fixed step length is used, which produces an error because pedestrians perform steps of different length. An algorithm as presented in Weinberg (2002) could be used, to add a dependency on the peaks detected in the accelerometer signal, which is used to calculate steps of varying length.
For the problem of inaccuracies through wrong magnetometer readings, additional absolute localization systems could be used to minimize the error as suggested in Kao (1991). Absolute localization systems like GPS and WiFi would also be a valuable addition to provide the initial location for the dead reckoning system, as proposed in Baranski et al. (2009).

Additional features could use the readings of altimeter sensors, which are available in the iOS frameworks since iOS 8.0 on the iPhone 6 and iPhone 6+. The readings of these sensors could be utilized to detect the current floor inside a building. Step detection algorithms as explained in Jayalath and Abhayasinghe (2013) are capable of detecting if the user climbs down- or upstairs. Using this capability together with

Altimeter readings could be used for monitoring floor changes.

An important improvement in terms of usability is determining the current motion axis of the accelerometer or the gyroscope, to be flexible regarding the carrying position of the smartphone, as proposed in Tumkur and Subbiah (2012). However, the user would still need to carry the device in order to be able to generate the trace information.

As many participants of the field experiment mentioned, using maps or floor plans as background of the trace visualization would help to retrieve the position of the personal items, because it would allow the user to map the trace to real world objects. Instead of using circles highlighted through different colors, icons illustrating the personal items would allow the user to determine the position of a specific item more easily.

The architecture of the application could be changed to consist only of a mobile application without a server part. The methods used on the server application are all available in the mobile frameworks, so this switch would be very easy and less source code would be required, because the client-server communication becomes obsolete. Users could analyze their visualizations directly on their smartphone, if they want to retrieve an item.

An additional feature on top of the localization system could establish an indoor navigation system, which navigates the user directly to the desired personal item. A simple version could navigate the user backwards on the recorded trace. If the application still requires the user to carry the device in the pocket, text-to-speech functionality could be used to provide speech commands. A more sophisticated approach could analyze the collected trace information and calculate a shorter path to the personal items, like traditional car navigation systems do.

As described in this section, the basic idea of the application provides numerous suggestions for improvements and additional features. Indoor localization systems are a very popular research topic, so new possibilities for this application will arise

very likely. Not only research will advance, also devices will. Nowadays, modern smartphones contain all necessary sensors to implement indoor localization systems. In the near future smart watches will also become interesting devices for the development of these systems, because they will include all necessary sensors and are designed to be worn all the time by users, which could solve the problem that the users always have to carry the tracking device.

List of Figures

2.1 Linear three-axis accelerometer (adapted from Aviv et al. (2012)) . . 8

2.2 Gyroscope measurement (adapted from Rizqi et al. (2011)) 9

2.3 Layers of iOS (from Apple Inc. (2010)) 12

2.4 Android system architecture (from Butler (2011)) 14

3.1 Raw accelerometer signal (from Mladenov and Mock (2009)) 17

3.2 The human step (from Kim et al. (2004)) 18

3.3 Horizontal acceleration signal of a human step (from Kim et al. (2004)) 18

3.4 Detected peaks in the processed acceleration signal (from Mladenov and Mock (2009)) 20

3.5 Thigh position recorded with a gyroscope (from Jayalath and Abhayasinghe (2013)) 21

3.6 Zero crossing in the gyroscope signal (from Jayalath and Abhayasinghe (2013)) 22

3.7 Robust heading determination (from Kothari et al. (2011)) 24

3.8 Vertical movement of the hip while walking (from Weinberg (2002)) . 26

3.9 Distance estimation between reference image and actual image (from Werner et al. (2011)) 30

3.10 Visualizations of pedestrian traces (from Jimenez et al. (2009)) 31

4.1 Architecture of the prototype application 40

4.2 Comparison of signals with or without low-pass filter 48

4.3 Comparison of accelerator and gyroscope signals 49

4.4 Rotation of the phone while walking) 50
4.5 Data visualization in the prototype application 55
4.6 The cartesian coordinate system 57

5.1 Layout of the test track 63

6.1 Differences in the gait patterns of the oldest and youngest participants 70
6.2 Unsual movements while placing an item 71
6.3 Visualized traces of all test participants 74
6.4 Item positions highlighted in the visualization 76
6.5 Positions of the items mapped to the test track 77
6.6 Results of the interview 79

List of Tables

3.1 Relevancy of different localization methods for this thesis 15

3.2 Comparison of magnetometer and gyroscope (adapted from Kim et al. (2004)) 23

3.3 Stride length depending on height and walking speed (adapted from Zhao (2010)) 25

3.4 Comparison of Weinberg (2002) and Feliz et al. (2009) (adapted from Jimenez et al. (2009)) 27

4.1 Values for x-axis and y-axis in the four quadrants of the cartesian coordinate system 57

5.1 Demographical data of the test users 64

6.1 Reference data 68

6.2 Accuracy of the step detection algorithm 69

6.3 Step Length Deviation 72

6.4 Total Distance Deviation 73

6.5 Total Distance Deviation 75

6.6 Results of the interview 77

References

Abhayasinghe, Nimsiri and Iain Murray [2012]. *A Novel Approach for Indoor Localization Using Human Gait Analysis with Gyroscopic Data. International Conference on Indoor Positioning and Indoor Navigation.*

Ahn, Hyo-Sung and Wonpi Yu [2009]. *Indoor localization techniques based on wireless sensor networks. Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions*, pages 277–302. doi:10.5772/6998. `http://www.intechopen.com/source/pdfs/6617/InTech-Indoor_localization_techniques_based_on_wireless_sensor_networks.pdf`.

Anastasi, Giuseppe, Renata Bandelloni, Marco Conti, Franca Delmastro, Enrico Gregori, and Giovanni Mainetto [2003]. *Experimenting an indoor bluetooth-based positioning service. 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 480–483. doi:10.1109/ICDCSW.2003.1203598.

Apple Inc. [2010]. *iOS Technology Overview.* (December 2014). `http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html`.

Apple Inc. [2011]. *iOS Human Interface Guidelines.* (January 2015). `http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html`.

Apple Inc. [2014]. *iOS Application Programming Guide.* (December 2014). `https://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html`.

Au, Antea Wain Sy, Chen Feng, Shahrokh Valaee, Sophia Reyes, Sameh Sorour, Samuel N. Markowitz, Deborah Gold, Keith Gordon, and Moshe Eizenman [2012]. *Indoor Tracking and Navigation Using Received Signal Strength and Compressive Sensing on a Mobile Device.* (ii), pages 1–14. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6261513.

Aubeck, Ferenc, Carsten Isert, and Dominik Gusenbauer [2011]. *Camera based step detection on mobile phones.* doi:10.1109/IPIN.2011.6071910. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6071910.

Aviv, Adam, Benjamin Sapp, Matt Blaze, and Jonathan Smith [2012]. *Practicality of accelerometer side channels on smartphones.* In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 41–50. ACSAC '12, ACM. ISBN 978-1-4503-1312-4. doi:10.1145/2420950.2420957.

Baranski, Przemyslaw, Michal Bujacz, and Pawel Strumillo [2009]. *Dead reckoning navigation - supplementing pedestrian GPS with an accelerometer-based pedometer and an electronic compass.* 7502, pages 750216–750216–6. doi:10.1117/12.837522. http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.837522.

Baranski, Przemyslaw and Pawel Strumillo [2012]. *Enhancing Positioning Accuracy in Urban Terrain by Fusing Data from a GPS Receiver, Inertial Sensors, Stereo-Camera and Digital Maps for Pedestrian Navigation. Sensors (Peterboroug*, 12(6), pages 6764–6801. ISSN 14248220. doi:10.3390/s120606764. http://www.mdpi.com/1424-8220/12/6/6764/.

Bihler, Pascal, Paul Imhoff, and Armin B. Cremers [2011]. *SmartGuide – A Smartphone Museum Guide with Ultrasound Control. Procedia Computer Science*, 5, pages 586–592. ISSN 18770509. doi:10.1016/j.procs.2011.07.076. http://linkinghub.elsevier.com/retrieve/pii/S1877050911004017.

Bill, Ralf, Clemens Cap, Martin Kofahl, and Thomas Mundt [2004]. *Indoor and Outdoor Positioning in Mobile Environments—a Review and some Investigations on WLAN-Positioning. Geographic Information Sciences*, 10(2), pages 91–98. ISSN 1947-5683. doi:10.1080/10824000409480660.

Bleser, Gabriele and Didier Stricker [2009]. *Advanced tracking through efficient image processing and visual–inertial sensor fusion. Computers & Graphics*, pages 137–144. doi:10.1016/j.cag.2008.11.004. http://www.sciencedirect.com/science/article/pii/S0097849308001465.

Bowditch, Nathaniel [2002]. *The American Practical Navigator*. 9, National Imagery and Mapping Agency. ISBN 1-57785-271-0.

Butikov, Eugene [2006]. *Precession and nutation of a gyroscope*. In *European Journal of Physics*, volume 27, pages 1071–1081. ISSN 0143-0807. doi:10.1088/0143-0807/27/5/006.

Butler, Margaret [2011]. *Android: Changing the mobile landscape. IEEE Pervasive Computing*, 10(1), pages 4–7. ISSN 15361268.

Cook, Nate [2014]. *CMDeviceMotion. NSHipster*, (December 2014). http://nshipster.com/cmdevicemotion.

Elloumi, Wael, Kamel Guissous, Aladine Chetouani, Raphael Canals, Remy Leconge, and Bruno Emile [2013]. *Indoor navigation assistance with a Smartphone camera based on vanishing points. International Conference on Indoor Positioning and Indoor Navigation 2013*, pages 28–31.

Evennou, Frédéric and François Marx [2006]. *Advanced Integration of WiFi and Inertial Navigation Systems for Indoor Mobile Positioning. EURASIP Journal on Advances in Signal Processing*, 2006, pages 1–12. ISSN 1687-6172. doi:10.1155/ASP/2006/86706. http://asp.eurasipjournals.com/content/2006/1/086706.

Feliz, Raúl, Eduardo Zalama, and Jaime Gómez García-Bermejo [2009]. *Pedestrian tracking using inertial sensors. Journal of Physical Agents, Vol. 3 No. 1 January 2009*, 3(1), pages 35–43.

Gandhewar, Nisarg and Rahila Sheikh [2010]. *Google Android : An Emerging Software Platform For Mobile Devices. International Journal on Computer Science and Engineering*, pages 12–17. ISSN 09753397.

Gomez, Carles, Joaquim Oller, and Josep Paradells [2012]. *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology*. In *Sensors*, volume 12, pages 11734–11753. doi:10.3390/s120911734.

Hol, Jeroen, Thomas Schön, Fredrick Gustaffson, and Per Slycke [2006]. *Sensor fusion for augmented reality. 9th International Conference on Information Fusion*, pages 1–6. doi:10.1109/ICIF.2006.301604. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4085890http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4085890`.

Holčík, Michal [2012]. *Indoor Navigation for Android. Master's Thesis*, (Masaryk University, Faculty of Informatics).

Horita, Yousuke, Masaki Sekine, Toshiyo Tamura, Yutaka Kuwae, Yuji Highashi, and Toshiro Fujimoto [2008]. *New attempt of proposing the pedometer algorithm in the elderly. Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks*, pages 111–112. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4575030`.

Jayalath, Sampath and Nimsiri Abhayasinghe [2013]. *A Gyroscopic Data based Pedometer Algorithm. djks123.cn*, pages 551–555. doi:10.1109/ICCSE.2013.6553971. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6553971http://www.djks123.cn/ieee_cd/data/papers/12838.pdf`.

Jimenez, Antonio, Fernando Seco, Carlos Prieto, and Jorge Guevara [2009]. *A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. IEEE International Symposium on Intelligent Signal Processing*, pages 37–42. doi:10.1109/WISP.2009.5286542. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5286542`.

Jin, Yunye and Wee-Seng Soh [2011]. *A robust dead-reckoning pedestrian tracking system with low cost sensors. 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 222–230. doi:10.1109/PERCOM.2011.5767590. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5767590`.

Kao, Wei-Wen Kao Wei-Wen [1991]. *Integration of GPS and dead-reckoning navigation systems. Vehicle Navigation and Information Systems Conference 1991*, pages 635–643. doi:10.1109/VNIS.1991.205808. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1623672.

Kim, Jeon Won, Han Jin Jang, Dong-Hwan Hwang, and Chansik Park [2004]. *A step, stride and heading determination for the pedestrian navigation system. Journal of Global Positioning Systems*, 3(1-2), pages 273–279. http://www.gnss.com.au/JoGPS/v3n12/v3n12p34.pdf.

Klingbeil, Lasse, Michailas Romanovas, Patrick Schneider, Martin Traechtler, and Yiannos Manoli [2010]. *A modular and mobile system for indoor localization. International Conference on Indoor Positioning and Indoor Navigation IPIN*, pages 15–17. doi:10.1109/IPIN.2010.5646700. http://urd.vgtu.lt/media/files/12/justinas-trapnauskasthepaper.pdf.

Kothari, Nisarg, Balajee Kannan, and M Bernardine Dias [2011]. *Robust Indoor Localization on a Commercial Smart-Phone*. Technical Report CMU-RI-TR-11-27, Robotics Institute, Pittsburgh, PA.

Link, Jó Ágila Bitsch, Paul Smith, Nicolai Viol, and Klaus Wehrle [2011]. *Footpath: Accurate map-based indoor navigation using smartphones. International Conference on Indoor Positioning and Indoor Navigation*, pages 1–8. doi:10.1109/IPIN.2011.6071934. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6071934http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6071934.

Liu, Hui, Houshang Darabi, Pat Banerjee, and Jing Liu [2007]. *Survey of Wireless Indoor Positioning Techniques and Systems. IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 37(6), pages 1067–1080. ISSN 10946977. doi:10.1109/TSMCC.2007.905750. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4343996.

Marschollek, Michael and Mehmet Goevercin [2008]. *A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons. 30th Annual International Conference*

of the IEEE on Engineering in Medicine and Biology Society, pages 1319–1322. ISSN 1557-170X. doi:10.1109/IEMBS.2008.4649407. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4649407.

Meier, Reto [2010]. *Professional Android Application Development*. Wiley Publishing, Inc. ISBN 978-0-470-34471-2, 580 pages. http://books.google.com/books?id=ewMdnKNYfOsC&pgis=1.

Mladenov, Martin and Michael Mock [2009]. *A Step Counter Service for Java-Enabled Devices Using a. CAMS '09 Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*, pages 1–5.

Mulloni, Alessandro, Daniel Wagner, Dieter Schmalstieg, and Istvan Barakonyi [2009]. *Indoor positioning and navigation with camera phones. Pervasive Computing*, 8(2), pages 22–31. ISSN 1536-1268. doi:10.1109/MPRV.2009.30. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4814934http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4814934.

Otsason, Veljo, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara [2005]. *Accurate gsm indoor localization. UbiComp 2005: Ubiquitous Computing*, (iv), pages 141–158. doi:10.1007/11551201_9. http://www.springerlink.com/index/B6TBWEUG7QM91BWK.pdf.

Ozdenizci, Busra, Kerem Ok, Vedat Coskun, and Mehmet N Aydin [2011]. *Development of an Indoor Navigation System Using NFC Technology*. doi:10.1109/ICIC.2011.53. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5954491.

Renaudin, Valérie, Muhammad Haris Afzal, and Gérard Lachapelle [2010]. *New method for magnetometers based orientation estimation. Position Location and Navigation Symposium*, pages 348–356. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5507301.

Renaudin, Valérie, Melania Susi, and Gérard Lachapelle [2012]. *Step length estimation using handheld inertial sensors. Sensors (Basel, Switzer-*

land), 12(7), pages 8507–25. ISSN 1424-8220. doi:10.3390/s120708507. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3444061&tool=pmcentrez&rendertype=abstract.

Rizqi, Maulana, M Hariadi, and Markus Haid [2011]. *Implementation of Inertial Sensors in an Interactive User Interface for Mobile Devices (iOS)*. pages 1–4. http://digilib.its.ac.id/public/ITS-Master-17992-2209205019-Paper.pdf.

Ruotsalainen, Laura, Heidi Kuusniemi, and Ruizhi Chen [2011]. *Visual-aided Two-dimensional Pedestrian Indoor Navigation with a Smartphone. Journal of Global Positioning Systems*, 10(1), pages 11–18. ISSN 14463156. doi:10.5081/jgps.10.1.11. http://www.gnss.com.au/JoGPS/v10n1/JoGPS_v10n1p11-18.pdf.

Serra, Alberto, Tiziana Dessì, Davide Carboni, Vlad Popescu, and Luigi Atzori [2010]. *Inertial Navigation Systems for User - Centric Indoor Applications. Screen*, pages 2–6. http://nem-summit.eu/wp-content/plugins/alcyonis-event-agenda/files/NEM2010-Paper675_camera-ready.pdf.

Shanklin, Teresa A, Benjamin Loulier, and Eric T Matson [2011]. *Embedded sensors for indoor positioning. Sensors Applications Symposium*, pages 149–154. doi:10.1109/SAS.2011.5739822.

Smittle, Devin, Veselin Georgiev, Yi Shang, and Wang [2010]. *Indoor localization on mobile phone platforms using adaptive dead reckoning.*

Steinhoff, Ulrich and Bernt Schiele [2010]. *Dead reckoning from the pocket-An experimental study. Pervasive Computing and Communications*, pages 162–170. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5466978.

Tumkur, Kashyap and Suneeth Subbiah [2012]. *Modeling Human Walking for Step Detection and Stride Determination by 3-Axis Accelerometer Readings in Pedometer. 2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation*, pages 199–204. doi:10.1109/CIMSim.2012.65. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6338075.

Weinberg, Harvey [2002]. *Using the ADXL202 in pedometer and personal navigation applications. Analog Devices AN-602 application note*, pages 1–8. http://www.bdtic.com/DownLoad/ADI/AN-602.pdf.

Werner, Martin, Moritz Kessel, and Chadly Marouane [2011]. *Indoor positioning using smartphone camera. 2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6. doi:10.1109/IPIN.2011.6071954. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6071954.

Zhao, Neil [2010]. *Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer. Analog Dialogue*, pages 1–5. http://www.analog.com/library/AnalogDialogue/archives/44-06/pedometer.pdf.

Zirari, Soumaya, Philippe Canalda, and Francois Spies [2010]. *WiFi GPS based combined positioning algorithm. International Conference on Wireless Communications, Networking and Information Security*. doi:10.1109/WCINS.2010.5544653. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5544653.