COM SYS Communication & Distributed Systems

RWTH AACHEN UNIVERSITY

Attitude Independent Movement Determination using MEMS for Screen-less Pedestrian Navigation

Master Thesis

Jens Helge Reelfs

RWTH Aachen University, Germany

Chair for Communication and Distributed Systems

Advisors:

Dipl.-Inform.	Jó Ágila Bitsch Link
Prof. Dr.-Ing.	Klaus Wehrle
Prof. Dr.	Bernhard Rumpe

Registration date: 2012-11-19
Submission date: 2013-03-28

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 28. März 2013

Abstract

While indoor navigation systems attract more and more attention, smartphone-based dead reckoning systems heavily depend on the user holding the device in a specific way. As soon as a user puts the device in her trouser pocket or jacket, they fail. In this thesis, we enable smartphones to accurately determine steps and the current bearing of a user even in these conditions.

We build a general model of the walking motion with specific instances for different device locations (trousers pocket, jacket pocket). Slicing the measured data within a step to only extract the segment which provides the most information on the user's walking direction, our approach has a median absolute error of only 12° (mean: 22°, q75: 25°) on a total of 15 participants completing a total of 49 test runs. Therefore, together with proposed feedback generation methods, this thesis lies the foundation of true hands-free indoor navigation without the need for any infrastructure.

Kurzfassung

Während das Gebiet der Gebäudenavigation mehr und mehr an Bedeutung gewinnt, hängt Smartphone-basierte Schritterkennung in großem Maße von der genauen Lage des Gerätes ab. Diese Methoden versagen, sobald ein Nutzer sein Gerät in seiner Hosen- oder Jackentasche trägt. In dieser Arbeit wird ein Ansatz vorgestellt, der es ermöglichts auch unter solchen Bedingungen Schritte und die aktuelle Richtung des Nutzers präzise bestimmen zu können.

Ein generelles Modell der Schrittbewegung dient als Basis für verschiedene Erkennungsinstanzen bei unterschiedlicher Lage des Gerätes (Hosentasche, Jackentasche). Indem aus den gemessenen Daten eines Schrittes nur genau das Segment berücksichtigt wird, welches die genauesten Informationen zur Bewegungsrichtung eines Nutzers liefert, zeigt unser Ansatz nur einen durchschnittlichen absoluten Fehler von 12° (Median: 22°, q75: 25°) bei 15 Versuchsteilnehmern über insgesamt 49 Testläufe. Zusammen mit vorgeschlagenen Navigationsausgabemechanismen, liefert diese Arbeit somit die Grundlage einer Gebäudenavigation bei der die Hände frei bleiben ohne den Rückgriff auf jegliche Infrastruktur.

"You mean this isn't woodshop class?"
— Emmanuel Goldstein
a.k.a. Cereal Killer

Contents

1

Introduction

Nearly everybody uses car navigation systems in everyday life, whereas we are walking most of the time - so why do we restrict this application to the car?

Imagine Bob wants to vist his wife Alice at the University Hospital Aachen as she had a surgery this morning. Due to the complexity of the building, he has no clue how to find her patient's room, but the only available information is, that she is on the intensive care unit IM18 room 4 which is located at corridor 11B. Luckily, he downloaded the building map before, such that he can use FootPath - an indoor pedestrian navigation system for smartphones - in order to find Alice as quick as possible. However, he does not want to stick to his smartphone display as otherwise, he might disturb other patients on his way through the building. Without any further local knowledge of the clinic, he finally found his cheerful wife.

Typical navigation application suffers from the non-availability of the Global Positioning System (GPS) indoors, which leads to the need of estimating the user's movement based on internal sensor data of the used device. Because of the desire to simply put the device anywhere (e.g. trousers or jacket pocket) as usual, the movement estimation is non-trivial. First of all, the measured sensor data is represented relative to the device. Furthermore, it may show high variance caused by the pedestrian walking motion in combination with the device location.

Moreover, all non-handheld locations do not allow the user to look at the screen which restricts feedback to non-visual media like audio and vibration. In comparison to passively showing the current user position on a map, the task of giving precise environment dependent feedback hints is more complex than simple turn-by-turn left/right instructions.

In this thesis, we present a new technique to determine the current user's heading at the trousers and jacket pocket location. Our method bases on a model of typical human walking which allows to segment the sensor data by single steps. We process the sensor data from each step-segment and calculate a heading estimation. From this, we have determined best fitting model parameters by optimization on a set of training data.

In addition to the heading estimation, we provide a flexible framework in order to give non-visual feedback. In general, users prefer to only get important abstracted and aggregated information only. This highly depends on the current scenario of the user, i.e., his position inside a building. We further present a Metric Linear Temporal Logic (MLTL) in order to programmatically determine the current user scenario more easily. As a proof of concept, we implemented two navigation feedback types: one providing verbal audio feedback and a second giving vibrational feedback.

1.1 Contributions

The main contributions of this thesis are:

- A method to distinguish between trousers pocket and jacket pocket device location,
- A new peak-based step detection depending on determined device location,
- A generalized model for heading estimation while walking,
- A method for model instance parameter determination on given training data,
- A detailed evaluation of our new method compared to other commonly used techniques like the android-internal fused *Rotation Vector* or using the magnetic field sensor in combination with the acceleration sensor,
- A comparison between our offline and online implementation of the presented new algorithm,
- A detailed investigation on the behavior of our method at different sensor data sampling rates,
- A flexible approach for generating usable navigation feedback and two prototypes using audio and vibration.

1.2 Structure of the Thesis

We present background information in Chapter 2. This includes used sensors, the used platform, pedestrian navigation applications and the current FootPath implementation.

Due to put this thesis intro context, we continue with related work in chapter 3. This includes Wireless Fidelity (WiFi) localization, pedestrian dead reckoning and navigation interface types. Because of this thesis' topic, we focus on dead reckoning and present most common techniques for step detection, step length estimation and heading estimation.

One main part of this thesis is our design decisions in Chapter 4. We introduce a generalized model for heading estimation. Based on a survey we conducted about smartphone carrying locations, we introduce two distinct model instances (trousers

pocket and jacket pocket). For this reason, we show how a device location detection technique and show the model parameter determination via optimization on training data. We further introduce a general framework for navigation feedback and two feedback type: verbal audio and vibration.

Chapter 5 discusses implementation decisions and encountered issues. We introduce an offline MATLAB and an online Java variant, wheras we focus on the online implementation. We finish the chapter by presenting the navigation feedback module.

Afterwards, we focus on providing a very detailed evaluation on several issues regarding the heading estimation. We compare our new approach against other available techniques.

We finish this thesis by giving a brief overview of future work in Chapter 7 and a conlusion in Chapter 8.

2

Background

This chapter will introduce several topics which are necessary to understand this thesis work. We explain basics on Inertial Navigation Systems (INSs) and Dead Reckoning (DR). After providing a brief introduction on the used sensors of typical smartphones, we present the Android platform. To understand the context, we introduce FootPath, its modules and how they generally work.

2.1 Inertial Navigation Systems

INS describe self-contained navigation systems which use an Inertial Measurement Unit (IMU) in order to estimate a moving system's position, orientation and velocity. Such IMUs typically contain motion and rotation sensors. Self-contained means, that there is no external reference. Such systems are widely used for navigation and navigation support e.g. in ships, submarines, aircrafts, guided missiles and spacecraft [10, 17].

DR is the main used concept which deduces inductively the current position from the last position and estimated movement. This works well in theory, but is hardly effected by errors of the measuring entities which get cumulated. Especially the usage of current low-cost Microelectromechanical systems (MEMS) sensors is not feasible for this task.

We show this approach in a simple example shown in Figure 2.1. We denote the current position as (x, y) and the current acceleration as a tuple for the x- and y-direction $(\ddot{x}, \ddot{y})$. Assuming constant acceleration, the double integrated acceleration for a timespan t will lead to the distances Δx and Δy, whereas the resulting direction angle Θ will be the arctangent of both. This leads to a new position $(x', y') = (x + \Delta x, y + \Delta y)$.

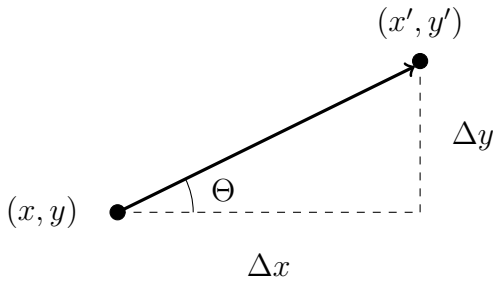

Figure 2.1 Dead Reckoning. DR deducts from the last position and current sensor information the new estimated position. This figure shows a simple 2D case with the origin position (x, y), the measured movements Δx and Δy which leads to the new position (x', y'). The movement-direction Θ will be the arctangent of both.

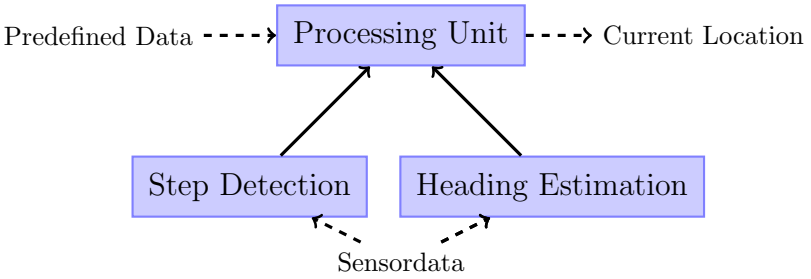

Figure 2.2 Pedestrian Dead Reckoning. The typical modules of a PDR system. The system uses internal sensors to determine the current movement and direction. A processing unit uses external data and parameters to merge information for computing a new estimated location.

2.1.1 Pedestrian Dead Reckoning

A specialized application of dead reckoning is the Pedestrian Dead Reckoning (PDR). At PDR, two different approaches are widely established. These use either foot mounted devices or some hand held device like a smartphone.

The typical PDR running on hand held devices consist of at least two different modules: step detection and heading estimation. The system may be further improved by adding more sophisticated filtering techniques or map data. However, Figure 2.2 depicts a generalized overview of such a system.

2.2 MEMS

Microelectromechanical systems (MEMS) systems are ambient light and temperature sensors, barometers and humidity-sensors. Such MEMS sensors are typically made up of components ranging in the size of micrometers. Moreover, they typically have a low weight, low power consumption and a short start-up-time. Thus, these sensors are feasible for embedding into all kinds of everyday devices.

However, such low cost MEMS devices have a big drawback as well. They are error prone, typically introducing drift and noise.

2.2.1 Gyroscope

A MEMS Gyroscope makes use of the Coriolis effect which states that a mass moving inside a rotating system experiences a force. The MEMS implementation pushes a tiny mass back and forth along one axis (vibrating), whereas the Coriolis force will make this mass veer away from this specified vibration direction and keeps vibrating in another axis, whenever the device is rotated [37]. Such gyroscope implementations have the main disadvantage that they are far less accurate than e.g. optical ones.

They cannot directly measure an angle, but provide an angular speed. Usually, there are three different gyroscopes put together into a single device for full 3D-rotation.

2.2.2 Accelerometer

MEMS Accelerometers typically measure the displacement of a supported mass against a reference frame or changes in the frequency of a vibrating element. This is realized by small springs which bend at acceleration. They also introduce errors such as bias and other external influences like temperature and pressure.

The typical output gives an acceleration against the gravity. Usually, there are three different accelerometers combined on different axes for 3D-acceleration measurements. In particular an accelerometer measures the force towards the sensor itself is denoted as F_s, where A_d describes the acceleration applied to the device. Furthermore the gravity always has impact on the acceleration applied to the device, which leads to another relation with the force towards the device denoted as F, while assuming constant earth gravity as $\mathrm{g} = 9.80665\mathrm{m/s}^2$:

$$A_d = \sum \frac{F_s}{\text{mass}}$$
$$A_d = \mathrm{g} + \sum \frac{F}{\text{mass}}$$

2.2.3 Magnetic Field

There are several MEMS approaches for creating a magnetic field sensor depending on the manufacturer and architecture. The most prominent implementation makes use of the Hall effect. To measure the magnetic field, a current is passed through a wire. Because of influences of the magnetic field, electrons show a higher density at one side of the wire compared to the other one, which introduces a voltage across the wire being proportional to the magnetic field [37].

However, due to external influences, it is not always possible to determine a reliable orientation towards north (by estimating the orientation towards the magnetic north pole) from the given magnetic field data. Especially inside buildings, there are often disturbances in the magnetic field because of permanent magnets, large iron bodies and electric current.

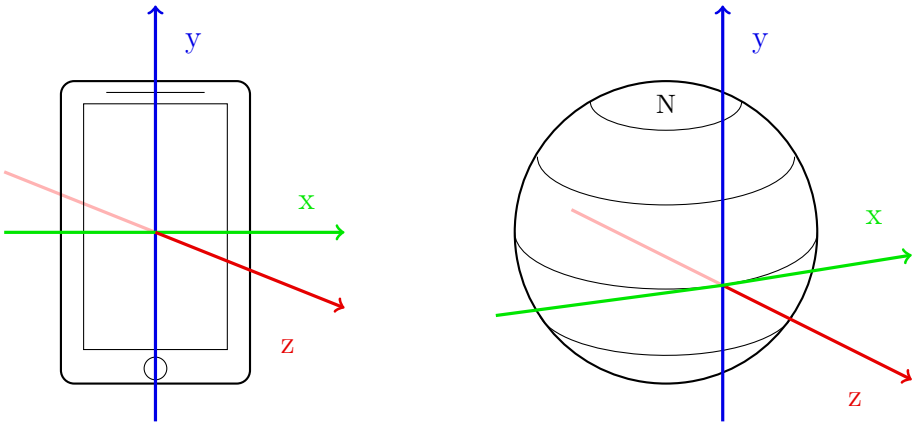

Figure 2.3 Instrinsic and Extrinsic Coordinates.
Left: The measured sensor data is typically expressed as intrinsic coordinates, i.e. relative to the device.
Right: By using the fusioned device orientation, we can transform intrinsic to extrinsic coordinates, i.e. relative the to world. Note that this is only one example how to define the axes [1].

2.2.4 Sensor Fusion

The physical sensors in general give useful information, but we can do further filtering and combine several sensors to either improve results or just for convenience. We will call this combining procedure Sensor Fusion. In fact, e.g. the Android platform provides the following fused "virtual" sensors:

Linear Acceleration. The acceleration sensor returns the raw acceleration values. This means, that these values include earth's gravity. By removing the gravity, we get pure linear acceleration, i.e., the relative change:

$$A_d^{rel} = \mathrm{g} + A_d$$

Orientation. Each compass uses the measured magnetic field of earth to get the direction towards north which means, that the magnetic field enables determining the actual orientation.

Rotation Vector. We have seen, that the gyroscope introduces drift and will lead to long-term errors, but has a good short-term behavior. For stabilizing the orientation estimation, it is possible to use the magnetic field and accelerations as well [37].

We can use the orientation values to transform the device's relative, *intrinsic*, coordinates into the world, *extrinsic*, coordinates. Figure 2.3 shows an example of the different coordinate systems. As the position of the device is not fixed, the transformation from intrinsic to extrinsic coordinates only consists of a three-dimensional rotation. Thus, we can set up a rotation matrix $m \in \mathbb{R}^{3\times3}$ for transformation.
With this matrix, we can easily convert e.g. acceleration values via a simple matrix multiplication into accelerations relative to the world:

$$A_{\text{extrinsic}} = m \cdot A_{\text{intrinsic}}.$$

We will call the special case, that the z-axis of intrinsic and extrinsic coordinates already match, *null-orientation*. In this case we can already use the magnetic field values without rotation to determine the orientation towards north.

2.3 Android

Android is an open source software stack, that includes the operating system, middleware and a set of key mobile applications which have originally been designed for mobile phones. Moreover, Android also provides a wide range of libraries. The first version has been released in late 2008 by the Open Handset Alliance (OHA) under the lead of Google. The source code is released under the Apache Software License at version 2 [36].

Later on, a version for tablets (Honeycomb) has been published followed by the current version family Android 4 (Ice Cream Sandwich (ICS) and Jelly Bean). The operating system is based on a linux kernel, whereas applications are executed in a specialized Java virtual machine, the Dalvik VM. Applications ship as single package files and can be easily installed. Today, a big variety of devices use Android - Looking into market reveals that Android is in 2012 the most successful Operating System (OS) for mobile devices with over 68% market share, followed by iOS with about 17% which is only available on Apple devices. Table 2.1 shows a complete market overview.

OS	Q2 2012 Shipments	Q2 2012 Market share	Q2 2011 Shipments	Q2 2011 Market share
Android	104.8	68.8%	50.8	46.9%
iOS	26.0	16.9%	20.4	18.8%
BlackBarry OS	7.4	4.8%	12.5	11.5%
Symbian	6.8	4.4%	18.3	16.9%
Windows Phone 7/ Windows Mobile	5.4	3.5%	2.5	2.3%
Others	3.6	5.8%	3.9	3.5

Table 2.1 Smartphone Operating Systems (OSs) Q2 2012 [Unit in Millions] [5]. Android is by far the market leader with an incresing shipment rate from 2011 to 2012.

Freely available source code, easy application development and distribution (e.g. via Google Play) is seen as the key factor for its success today. The main language for all Android applications is Java, whereas the Java Native Interface (JNI) also supports C and C++ code. Moreover, Android is considered as the main reason for the new blossom and popularity of Java [38].

Due to the openness, Android is adequate for rapid prototyping and development in general. The Application Programming Interface (API) does not always allow deep access into the system, but rooting[1] the device gives full access. This also enables overcoming limitations set by carriers or hardware manufacturers.

[1]Rooting describes the process of giving a user privileged access to the Android subsystem.

2.4 Navigation Applications

Navigation systems in general have become very popular especially for cars over the last decade. Although the same navigation techniques mostly using Global Positioning System (GPS) work for pedestrians well, indoor navigation introduces some serious issues. On the one hand, all navigation systems normally rely on map data. On the other hand, GPS may not be accurate enough at private usage for pedestrian navigation or even may not be available - at least indoors. Furthermore, having pure GPS-based systems and some indoor counterpart, the next step will be combining both techniques into a hybrid navigation system.

However, indoor navigation has some serious applications. First of all, such systems may aid blind or visually impaired people through buildings. Moreover typical larger buildings like airports, hospitals or shopping malls may be very complex. Due to complexity, the typical larger buildings provide floor plans or "you-are-here" maps. Apparently, such aids introduce time overhead. Besides classical navigation, it is furthermore possible to extend such indoor positioning system to provide location aware services.

2.5 FootPath

The context of this work is FootPath which is a project of the Chair of Communication and Distributed Systems at Rheinisch Westfälische Technische Hochschule Aachen (RWTH) which already started in 2010. The main task of FootPath is currently pure indoor pedestrian navigation on Android smartphones. The main reason for using smartphones is the wide and cheap availability of such devices. Moreover, some specialized e.g. foot mounted sensor devices are not practical in everyday life.

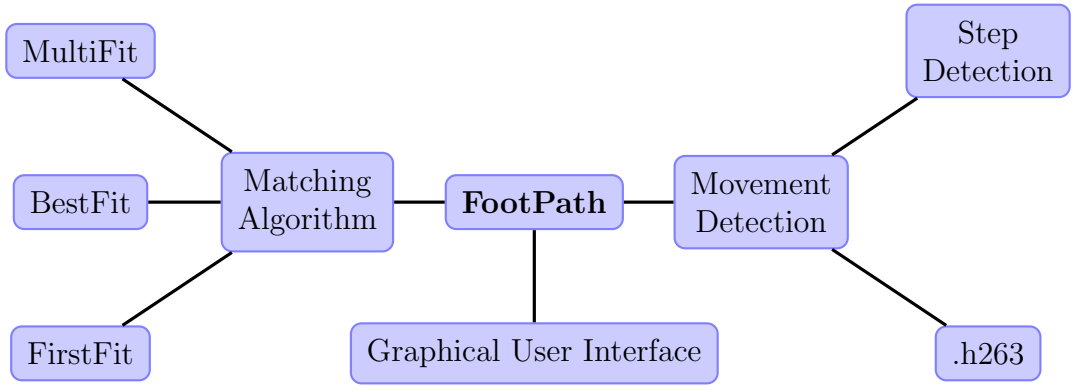

Figure 2.4 FootPath: Major Functional Building Blocks. The location estimation uses two major parts: the movement determination and matching algorithm. Feedback is provided via a GUI.

We can split the navigation process into three functional building blocks. We illustrate an overview in Figure 2.4. First of all, FootPath uses acceleration sensor data to detect steps that the user makes - or the device's camera to detect movement from its video stream. Furthermore, it calculates a heading estimation out of magnetic field sensor data. After having detected movement and direction, a matching

algorithm finally estimates the user's current position along a predefined path on the map by using sequence aligning, which compares expected heading values towards the measured ones.

Moreover, while navigating, the system shows a graphical user interface displaying the current map with the estimated user's position, the selected path and the currently covered path. However, we will dive into the functionalities in much more detail in Section 4.1.

2.5.1 Current Movement Detection

To detect movement, FoothPath currently supports two different techniques. The **simple step detection** observes accelerometer data and tries to detect changes in the z-acceleration. For this reason, the data will be smoothed by a simple Low-Pass (LP)-filter first. Afterwards a sliding window analyzes the acceleration changes between successive values. We detect a new step, whenever this change is greater than a predefined threshold. After detection, there will be a timeout interval to prevent false detections. We show an overview in more detail in Figure 2.5. The current user's bearing will simply use the internal magnetic compass. A predefined step length together with the step count allows to calculate the covered distance.

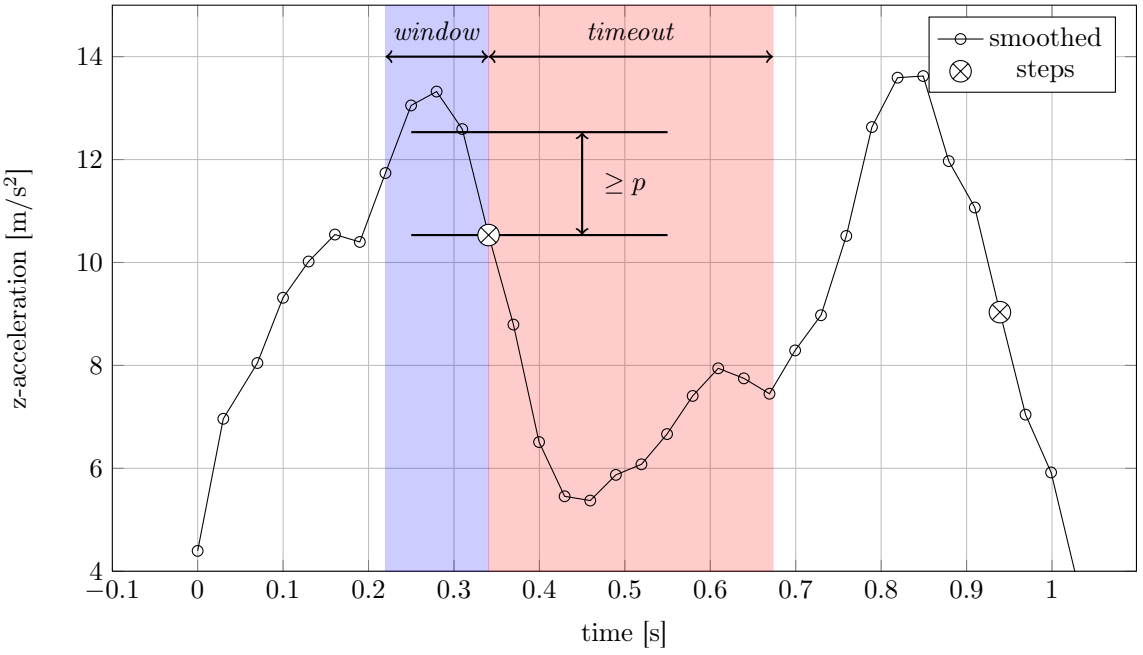

Figure 2.5 Simple Step Detection. The simple step detection finds a rapid decreasing z-acceleration to detect a new step by searching a window of five sensor values. We detect a new step whenever the decrease is greater than $p = 2\text{m/s}^2$. Afterwards there will be a timeout interval (here 333ms) to prevent false positives.

Another method, **FlowPath**, to determine movement uses the front-camera's video stream. The main idea is to use the optical flow to calculate the device's movement. FootPath does not calculate the optical flow explicitly as this would be by far too costly in terms of calculation, but exploits the video stream's compression technique.

The used compression (H.263) separates the image into certain macro blocks which will each move from frame to frame due to the similarity of sequential images [22]. As this actually happens with hardware accelerated video-compression, there is no need to calculate image features and to match them. In order to determine the overall movement (amount and direction) of the device, the system extracts macro block movement vectors [8].

2.5.2 Matching Algorithm

The most crucial part of FootPath is the matching algorithm. The step detection gives step-wise movement information with a certain distance and direction. To track the user's position on a given map, or more precisely a route, the matching algorithm tries to correlate the route's expected directions with the measured ones from the step detection. In fact, the more a route involves bearing changes, the better the matching typically works. There are currently three different matching algorithms implemented: FirstFit, BestFit and MultiFit, wheras we show an overview of matching techniques in Figure 2.6.

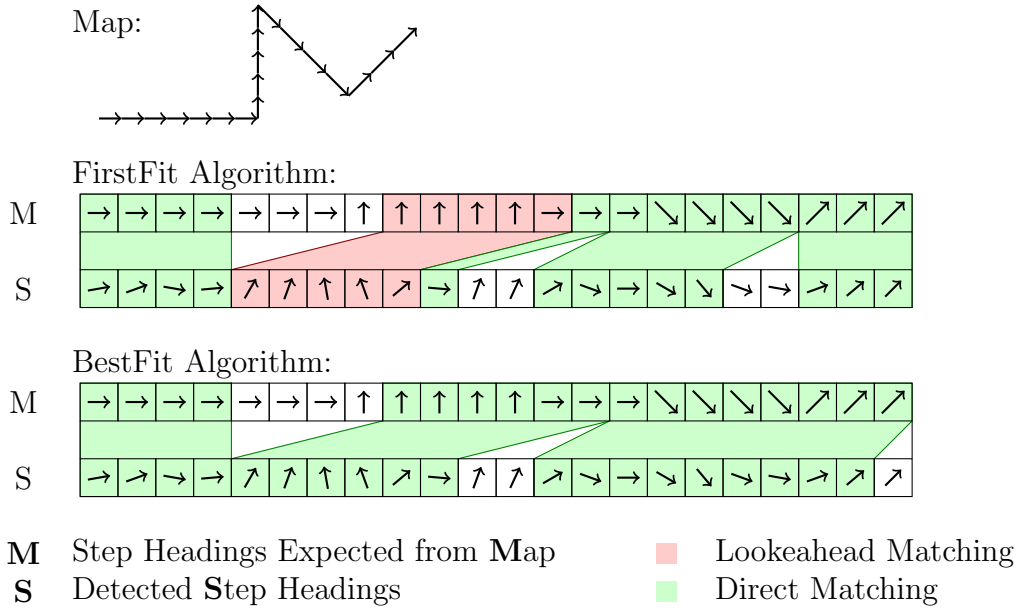

Figure 2.6 Matching Algorithmsin Comparison.
Top: The given path on the map with the expected bearings.
Middle: FirstFit alignment.
Bottom: BestFit alignment [9].

FirstFit. FirstFit consists of two matching modes. The *Direct Matching Mode (DMM)* is awaiting the next expected step bearing α given a predefined acceptance distance, typically $\sphericalangle(\alpha, \beta) \leq 42°$. The user's position updates if the measured step bearing β corresponds to the expected one. If this is not the case for a defined amount successive steps (typically $k = 5$), it is very likely, that the system has underestimated the step length. Thus, the matching method changes to *Lookahead Matching Mode (LMM).*

At first, the *LMM* tries to find (inside the range of the next k expected bearings) a matching expected bearing angle inside the given acceptance range. The algorithm

resumes in *DMM* if it finds one. Otherwise the algorithm remains in the lookahead mode and tries to find a segment of $k = 5$ consecutively matching headings on the path. The next upcoming steps are now collected until this segment meets the last unmatching expected step. After having found this connection, the algorithm can proceed in *DMM* [9].

BestFit. BestFit uses a sequence aligning algorithm which is also prominent in bioinformatics and similar fields. The idea is to reduce the measured bearing to expected bearing mapping to a string matching problem. This introduces a scoring function which gives a penalty to higher differences between expected and measured bearings. A dynamic programming approach which spans a matrix representing the scored relation between the expected and the measured step bearings can then solve this problem.
The most probable covered path, i.e. estimated user position, is the matrix entry with the smallest penalty value [9].

MultiFit. MultiFit is a natural extension of the BestFit idea. It follows multiple possible paths instead of a single path. The algorithm correlates expected to measured bearings with a scoring function similar to the one of BestFit for each path. The most probable estimated user position will then be the scoring entry with the smallest penalty score. For efficiency, a tree datastructure, which contains parts of the BestFit matrix (a partial penalty matrix), exploits typical shared prefixes of several routes. Breadth-first traversing of the complete tree allows the efficient calculation of the estimated user position [8].

2.5.3 Current Map Modeling

The maps used in FootPath use the OpenStreetMap Extensible Markup Language (XML) format which is simple, portable and an established standard. The Java OpenStreetMap Editor (JOSM) tool allows creation and modification conveniently. Although there are properties defined by the OpenStreetMap Community, we have to model some more parameters additionally. The Open Street Map (OSM) file format supports nodes each with an absolute location (latitude and longitude) and a set of key-value pairs. Moreover the format supports directed edge-sequences between such nodes which also may have a set of key-value pairs.

For modeling buildings and their interior, all nodes and edges have a key for the floor-`level` in general. Moreover, they have a value `indoor`={`yes`, `no`} which indicates being indoor or outdoor. The format represents walls as edges witch the key `buildingpart=wall`; Note that outer building walls are defined as non-indoor. Special areas like elevators or steps will have a corresponding key-value pair `buildingpart`={`steps`, `elevator`}.

For actual navigation, we represent possible ways through the building by edges each connecting two nodes which may have extra information like the key `name`=*Char** for specific room-names or `building`=`entrance` for building entries.

The connecting edges introduce more information as they indicate whether the region is accessible by wheelchair (`wheelchair`=`yes`) or not. Especially stairs and steps are not usable by wheelchair and provide additional data like the number of steps (`steps`=$n \in \mathbb{N}$) as well as the tag `highway`=`steps` by OSM convention [2].

2.5.4 User Interface

The user interface consists of two different main screens, the main menu and navigation. We provide examples in Figure 2.7. The menu allows to load map data and to select a starting location as well as a destination for navigation. Furthermore, it allows to select the desired matching algorithm and movement detection type. The simple step detection relies on an estimation of the user's step length which can also be configured here. While navigating, the application displays the current map with the user's current estimated location. The map includes map data like walls, stairs and elevators.

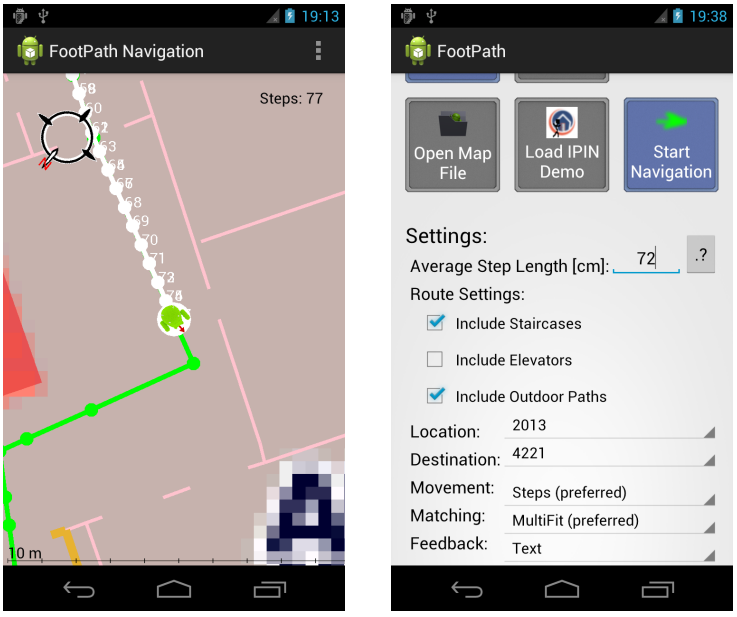

Figure 2.7 Navigation User Inferface.
Left: The FootPath navigation interface. It shows the current map and calculated route (green). Moreover, it shows information about the covered path (white) and the number of detected steps. A compass shows the current direction towards north. Moreover, the interface also gives the opportunity to pan and zoom with common gestures.
Right: The main menu allows to load map data, configure the step length, select current location and destination as well as used modules for navigation.

2.6 Summary

We have introduced a basic background of inertial navigation and dead reckoning (2.1). Furthermore, we have presented the different types of MEMS sensors and typical sensor fusion applications (2.2). This is followed by a short introduction to Android (2.3) and navigation applications in general (2.4). This chapter will finish with introducing FootPath (2.5). We explain and discuss the overall structure and different modules which includes the current movement detection and matching algorithm as well as the user interface.

3

Related Work

In this chapter, we will give an overview on inertial navigation. Although this is not a new topic, in recent years, it has become an actively studied research area since Inertial Measurement Unit (IMU)-enabled devices (e.g. smartphones) are widespread and cheap. After discussing Wireless Fidelity (WiFi)-based localization, we elaborate on Dead Reckoning with a special focus on heading estimation and step detection as well as step length estimation. We furthermore show a comparison of Pedestrian Dead Reckoning (PDR) results from contributions of several authors. Finally, we introduce different types of navigation interfaces which are visual audio and vibration feedback.

3.1 WiFi-Localization

Besides Dead Reckoning (DR), there have been solutions published which only use WiFi fingerprinting for localization and navigation. While, the main disadvantage is the need for infrastructure. Nevertheless, such approaches may be useful and are often combined with PDR.

Generally, providing a WiFi-localization system consists of two phases - (1) information retrieval and (2) usage afterwards. The information retrieval phase is responsible for creating and filling a fingerprint database. Such fingerprints usually contain RSSI values towards available Access Points (APs) in a certain geographic location. Then, estimating the current device location results in a matching towards possible locations in the database. Bahl et al. present an implementation of this method in [6].

But this simple approach has a drawback as well. The Received Signal Strength Indicator (RSSI) values suffer from a heavy variance - e.g. different device types, directions and carrying locations as well as environmental changes lead to heavy changes in the receiving power. However, there are approaches to overcome this issue, like Kim et al. in [30].

3.2 Dead Reckoning

DR, i.e. PDR, describes the DR approach (see 2.1) for pedestrians, where the displacement is represented by an amount of steps which each have a direction and step length. The main reason for dead reckoning is the unavailability - or at least the degraded signal - of Global Positioning System (GPS) indoors. Most of the solutions describe two different locations for the sensors, foot mounted devices and hand-held ones. Foot mounted devices enable better analysis of walking behavior than others, thus we will first introduce these approaches. In general, we can separate the movement determination into two different functional blocks. The heading estimation tries to detect the current user orientation, whereas the movement determination tries to estimate actual movement which is typically done via step detection.

3.2.1 Foot-Mounted Devices

Foot-mounted PDR systems enable sophisticated analysis and modeling of the user's walking behavior. Human walking can be split into different phases of the foot movement. Kim et al. present a model which consists of two swing phases as well as the Heel-touch down [29].

Such devices typically use accelerometers and gyroscopes as well as sometimes force sensitive capacitors to deduct walking. A main advantage of the positioning on the foot (or under the shoe) are Zero Velocity Update (ZUPT). As discussed before (2.2), the used IMU sensors introduce a high amount of errors which get accumulated over time yielding unfeasible results already after a few meters of walking. Recognition of a stationary attitude while a stance phase of the IMU can eliminate this issue by assuming having zero velocity. With this technique, the sensor data can be periodically recalibrated which resets the error each time.
Most research on this type of IMU positioning typically use unaided DR approaches [33, 39, 50].

3.2.2 Heading Estimation

The typical data source for heading estimation is the magnetic compass or gyroscope. Some approaches even combine both sensor data or include acceleration data. Most approaches concentrate on a well defined attitude of the IMU. In order to support other locations, a projection from intrinsic to extrinsic coordinates for the heading estimation is usually used to have a "virtually" well defined attitude.

In general, the magnetic field sensor performs well outdoors, but because of magnetic disturbances indoors, it gives false data in some cases. E.g. Liu et al. [35] show that the magnetic compass itself has a Root Mean Squared Error (RMSE) of $\approx 10°$ while not moving, whereas this error rises up to about 27° while walking.

Park et al. [40] present more information on the error characteristics at different device locations. Depending on their scenario, while walking, the error variation changes between about 10° up to 25°. They investigated on e.g. the trouser pocket and two hand held positions (calling and messaging). Looking into their statistics

reveals, that apparently the trouser location shows the biggest sensitivity to errors in the measured heading values.

In contrast to this, the gyroscope works unaffected by such influences and gives good results. However, the gyroscope introduces a high amount of drift, such that errors get accumulated. This means that the magnetic compass gives good long term accuracy, whereas the gyroscope is only valid for a very limited time span. For that reason, there are several approaches to combine both sensors. These either compare the different calculated headings by a threshold, e.g. by Kothari et al. [31], or use a correlation measurement between both sensor outputs e.g. Kang et al. [27].

Nevertheless, the error of the magnetic compass is usually disregarded [31,32,35,40, 42–44,49]. Beauregard and Haas [7] compensate this via including GPS data which of course restricts the usage to outdoors. For example Fink et al. [12] as well as Hong et al. [21] only use gyroscope data for heading estimation anyway.

3.2.2.1 More sophisticated approaches

Rai et al. [43] show another method to determine the user's heading. First, they introduce a magnetic offset which shows in tests an error of about 15° for different locations in a building. Moreover they investigate on the Fourier spectrum of a typical walking pattern. The spectrum shows a central frequency at about 2Hz which represents both strides - of the left and right leg. Furthermore, they use a second harmonic to determine the user's moving direction. However, as they only use the unsigned magnitude Fourier transform, they cannot distinguish between two valid solutions which are in opposite to each other. They state, that their heading estimation shows an error in a range of about 60°.

3.2.3 Movement and Step Detection

Besides the heading estimation, we need to detect movement in general. This is usually done by a method of step detection via e.g. acceleration sensor data. Other approaches are e.g. vision-aided using camera input to determine or at least support the movement determination by optical flow.

In this section, we will explain and discuss different common approaches for actual step detection briefly.

3.2.3.1 Zero-Crossing

One of the most prominent methods is, to determine zero crossings (ZCs) in the acceleration, e.g. used by Beauregard et al. [7]. For this reason, the acceleration values get first smoothed by a Low-Pass (LP) filter and then step counting uses either the acceleration magnitude $a = \sqrt{(a_x^2 + a_y^2 + a_z^2)} - g$ or the (linear) z-acceleration. Usually, developers introduce a threshold after having found a certain crossing event (step) to prevent false positives. This has the disadvantage that the defined threshold may be too long or too short in generic cases.

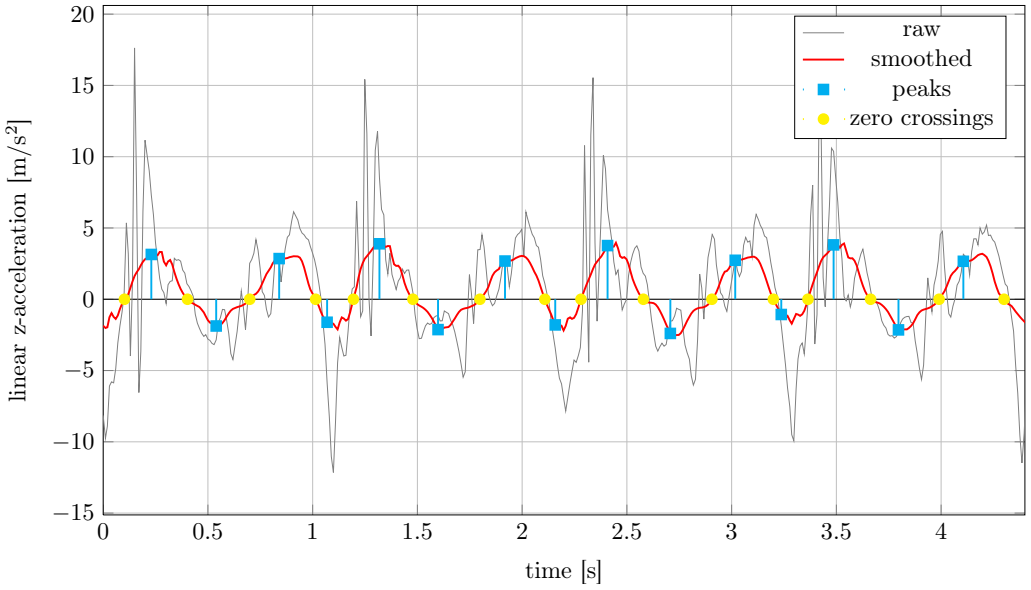

Figure 3.1 Peak- and zero crossing (ZC) Step Detection in Comparison. Typically, after the z-acceleration gets smoothed (here by a box-filter at a size of $250ms$), either peaks or zero crossings are calculated using a filtering technique. Due to the big smoothing kernel, the ZCs detection is unambiguous, whereas we have to use a timing-threshold for peak detection (here $200ms$).

A simpler method which does not exactly count ZCs is to just searches for a thresholded decrease in acceleration, e.g. used by Bitsch et al. [9].

3.2.3.2 Peak

Peak step detection is another prominent method to count steps (e.g. used in [31, 44, 49]). It observes the smoothed acceleration data and deducts local maxima and minima sequences inside a sliding window. A maximum is typically considered as valid, if it exceeds a certain upper threshold. The same holds for the minima with a predefined lower threshold. Furthermore, some implementations also introduce a cool-down threshold before detecting the next step.

An improvement to the fixed thresholds are dynamically adapted thresholds used by Fink et al. [12] and Pratama et al. [42]. They aggregate data values into a sum while storing maximum and minimum sums over a defined window. The dynamic threshold may then be some combination of both values (e.g. exactly in between).

Kang et al. [27] compare results using ZC and peak step detection in their algorithm. Their results show that both approaches give similar results. Shin et al. [31] show some experimental results of their peak step detection to distinguish between three different states of movement: stop, walking and running.

3.2.3.3 Gyroscope

Hong et al. [21] present another method to detect steps. They have observed the *turn insufficient effect* which states, that under some circumstances (e.g. the devices

location is straight in the trouser front pocket), the angular velocity of the y-axis while walking will be lower than while turning. They exploit this observation to detect steps and turning events. Their algorithm results for two different test paths (≈200m and 110m) in an average return position error of 1.33%.

In contrast to this, they present results using pure gyroscope and magnetic field sensor data which yield errors between 10% and 20% (which are most likely a result of accumulated errors).

3.2.3.4 Autocorrelation

Due to the repetitive nature of walking procedure, it is possible to exploit this for single step or movement detection. Rai et al. [43] present a new idea for step detection: Autocorrelation. As the acceleration pattern at walking is repetitive, it shows a strong correlation. To exploit this for step detection, they suggest to use a normalized autocorrelation technique over a certain window. The overall performance shows only a small amount of false positives or negatives for all different tested device locations (hand, trouser front and backpocket, shirt pocket and handbag).

3.2.3.5 Vision-aided

Visual information of the camera, i.e. a sequence of images, can provide information of the current devices movement which enables restricting the drift error of the IMU. The main idea is to extract this information from some type of optical flow, where we shortly present three different techniques.

Reference Database. Assuming we have a database with reference pictures of all possible navigation locations including its exact coordinates, a possible position estimation algorithm matches actual camera-images against the database. There are countless approaches for this purpose where fundamentals to this are presented by Forsyth and Ponce in [13]. The main disadvantage of this technique is the need for construction such a database which will be in most practical cases far too much overhead.

Feature Tacking. Hide and Botterill [19] describe classical approach for determining the displacement of the device. In general, they follow the DR approach, but furthermore, they calculate movement from images, where a Kalman Filter combines both inputs. Feature tracking allows optical flow determination. At first, a fast corner detection method will identify possible features. These features of subsequent images are then compared in terms of similarity as a sum-of-squares difference. A Direct Linear Transform (DLT) determines the homography-matrix H with the found correspondences. The homography matrix describes the change from one picture to the next. For good results, a random sample concensus (RANSAC) approach ensures validity of selected feature points. Finally, they get rotation R and translation $d^{-1}t$: $H = R + d^{-1}tn^T$ by decomposition.

Vanishing Point. The vanishing point describes the point at infinite distance, where all real-world parallel lines cross each other in images. Although it is not present at certain motion behavior (e.g. one-axis rotation only), it can be determined in most practical cases like corridors. Ruotsalainen et al. [45] present a simple technique to do so. At first, they detect edges of objects and identify straight lines. Afterwards, due to the characteristic of crossing lines, a voting approach estimates the vanishing point. As the pitch and roll (rotational change in x- and y-direction in terms of the extrinsic coordinate system seen in Figure 2.3) can be restricted, they use a simple equation to get a heading-change estimation. They introduce an approach to combine these results with IMU sensor data via a Kalman Filter [46].

Exploiting Video Compression. Bitsch et al. [8] present another approach exploiting the nature of video compression. They measure the current optical flow by extracting the movement vectors of the cameras compressed video stream. We already discussed this at Section 2.5.1.

3.2.4 Step Length Estimation

The PDR algorithms usually work with the two mentioned components - heading estimation and step detection. Another crucial part consists of the correct step length estimation. Wrong estimated step length values accumulate over time and unboundedly increase the error.

Besides other methods, a simple version to get a usable value just predefines a certain fixed value e.g. by a formula using some linear estimation function with a person's body height as a single parameter [9].

Foot mounted devices have the big advantage of the zero velocity updates which enable a new calibrated measurement for each step. Furthermore these devices typically have other hardware than Microelectromechanical systems (MEMS) sensors available. Simple integration of velocities will give useful results.

Randell et al. [44] introduce several ideas for step length estimation. Their first approach uses a preset of minimum and maximum step length. A second method scales a predefined step length depending on the acceleration magnitude of each step. The third method includes using GPS data.

Weinberg [53] introduced a commonly accepted and more sophisticated model (e.g. used in [24, 27]). The underlying geometry model uses the amount of bouncing (z-axis displacement) of the hip while walking, see Figure 3.2. His model states by exploiting the similar angles $\alpha = \theta$, that Stride $= 2 \cdot \text{Bounce}/\alpha$, where θ denotes the angle between left and right leg and α the angle between right leg and the next steps hip position. Weinberg roughly estimates the distance for one stride with the following formula, where $A_{\max}$ and $A_{\min}$ denote the min and max z-component value of acceleration inside a single stride (note that k will be some correction value defined by the developer):

$$\text{Distance}_W = k \cdot \sqrt[4]{A_{\max} - A_{\min}}. \tag{3.1}$$

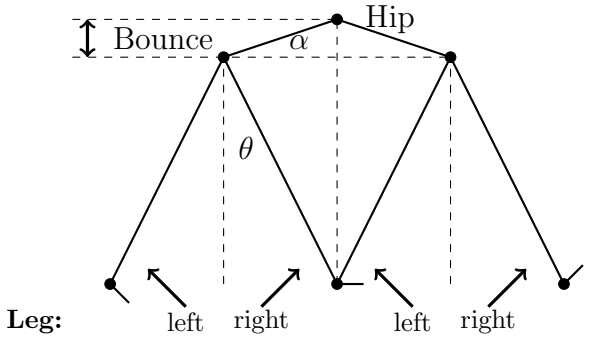

Figure 3.2 Weinberg Stride Length Estimation Geometry. The Weinberg model estimated the step length on the basis of Bounce magnitude of the walking procedure by using an approximation of the similar angles α and θ.

Kim et al. introduce in [29] another experimental derived formula, where N denotes the number of samples for a single stride:

$$\text{Distance}_K = k \cdot \sqrt[3]{\frac{\sum_{k=1}^{N} |A_k|}{N}}. \tag{3.2}$$

Scarlett [47] presents a third equation which correlates the minmum and maximum acceleration of a single stride:

$$\text{Distance}_S = k \cdot \frac{\frac{\sum_{k=1}^{N} |A_k|}{N} - A_{\min}}{A_{\max} - A_{\min}}. \tag{3.3}$$

All models can be easily adapted e.g. with the factor k. However, Pratama et al. [42] present an error estimation for each model, which shows that the Scarlett model outperforms all others. The Weinberg model is about 3% and the Kim model about 22% worse. Moreover, there are other models in use which are very similar to the introduced ones, but some regard more information e.g. like step time. Jahn et al. [23] show a theoretical error analysis and practical comparison as well.

3.2.5 Including Map Data

Although the presented techniques are already suited for navigation, including map data to the algorithms further improves performance. Rai et al. [43] as well as Kothari et al. [31] introduce a particle filter which keeps track of possible positioning probabilities. Shin et al. [49] propose a simple map matching algorithm which correlates the current user's position to nearest links and corners.

Using a Hidden Markov Model (HMM) for further improvements is similar to this approach, e.g. used by Liu et al. [35].

Bitsch et al. [9] introduce another map-matching approach which uses sequence aligning techniques to correlate measured headings to expected bearing on a given path.

3.2.6 Further Improvements

Neural networks. Shin et al. as well as Beauregard and Park et al. use Neural Networks (NNs) for further improving the step detection [7, 40, 49]. Such NNs get predefined features as input (e.g. step frequency, accelerometer peak values and heading) and calculate predefined results such as step length estimation and heading information. In fact, they act like a black box.

Kalman Filter. Another method to gain better performance is using a Kalman-filter to smooth results. It is widely used, e.g. in [15, 25, 29, 32, 50]. Simply said, a Kalman filter may be useful whenever a measured system shows noise as well as the measurement contains noise. Both error sources are finally put together via a specific model for each. The filter roughly contains two different steps - the measurement and time update. However, system modeling and setting up a Kalman filter is a complex process as the Kalman filter is often misunderstood and even wrongly used (resulting in a simple LP). Hence, we will not focus on this or the underlying statistical mathematics in this work any further.

Corridor environment restriction. Park et al. [40] introduce an approach which uses typical proposed methods: magnetic compass for heading estimation and acceleration data for step detection, but restrict used map data to perpendicular corridor design. This enables to stabilize the heading estimation values by detection corners along a navigation path - the rough idea is to only assume a corner when reaching a certain threshold of heading change. Moreover they distinguish different device positions by tracking the orientation sensor values to keep track of relative heading changes with a mean detection error of 5.75%.

Barometer. Todays smartphones have integrated air pressure sensors. Air pressure may be used to improve locating a user in terms of particle filters and map matching integration as they are typically sufficient precise. Keller et al. [28] have evaluated on the sensor output and conclude, that typical errors lead to wrong altitude estimations of about $\pm 0.85m$. Feliz et al. [11] also introduce the possibility to improve PDR via barometer sensor data.

3.2.7 Adding WiFi

Combining PDR with WiFi fingerprinting gives more useful information. E.g. Kothari et al. as well as Rai et al. present an architecture which uses PDR to track the user's position combined with a particle filter [31, 43]. Furthermore, their frameworks also includes WiFi scanning (measuring the RSSI values towards proximate APs) while navigating which helps to ensure certain locations. While Kothari et al. assume having a precollected WiFi database with information about APs, their signal strength and geographic location, Rai et al. proposed a crowdsourcing approach. However, both solutions have the clear advantage, that (subsequent) user's will not have to set their current start location for PDR anymore as this information may be available from the WiFi information.

3.2.8 Results

We have seen a lot of different methods - in placement of the device - and to get a heading as well as movement estimation. All implementations have certain different design decisions and parameter which makes the evaluations hard to compare. Nevertheless, Table 3.1 shows some comparable results.

In general, we have to interpret such results with care. The resulting error usually highly depends on the scenario. The most obvious observation is, that all foot mounted approaches provide good results with a low error rate mostly at about 1%.

Moreover, the other methods show, that most results are suited for navigation purposes. Step detection seems to work reliably regardless of the detection method. Although the exact methods are often not mentioned, comparing results which specify this explicitly, we can conclude that we get similar results in most cases.
However, in the context of this thesis, the device location is the most important and interesting question. The cases depicting the position, are mostly different from hand held. Investigation on these cases at the results of Hong et al. and Randell et al. shows, that the typical error becomes huge compared to others. Pratama et al. [42] mention that their technique is suitable for any location while not giving exact details on this. However, they introduce the possibility for the intrinsic to extrinsic transformation without any further analysis.

From these results, we conclude, that (at least to our knowledge) only few researchers have analyzed locations other than hand held (for non-foot mounted devices) in detail. All known cases which use a different location lead to the conclusion, that the dead reckoning error is much higher in general. Reasons for this may be an unreliability of the magnetic compass direction if the device is not in *null-orientation* together with higher dynamics in the whole devices movement and may lead to less stable sensor output. The data is typically harder to analyze and exploit for reliable heading estimation as well as step detection.

Approach	Heading	Step Detection	Step Length	Dist	rel. Err	Indoor	Device Location
Kothari [31]	mag	peak	*n/a*	100m	6%	√	*any*
Hong [21]	gyro	gyro	*n/a*	210m	9.8%	√	trouser
	mag	gyro		210m	9.2%	√	-pocket
	adv.	gyro		210m	0.9%	√	
Randell [44]	gyro, mag	peak	*n/a*	279m	9.7%	-	handheld
		peak		131m	29.8%	-	backpack
		peak		126m	3.2%	-	shoulder
Pratama [42]	gyro, mag	peak	fixed	30m	3.5%	√	(*any*)
		peak	scarlett	30m	2.0%	√	
		peak	weinberg	30m	2.2%	√	
		peak	kim	30m	2.6%	√	
Kang [27]	gyro	peak	weinberg	75m	3.2%	√	*n/a*
	mag	peak		75m	26.1%	√	
	adv.	peak		75m	1.3%	√	
	gyro	zc		75m	3.0%	√	
	mag	zc		75m	19.3%	√	
	adv.	zc		75m	1.7%	√	
Kim [29]	gyro, mag	-	kim	74m	3.0%	√	foot
		-		146m	4.2%	√	
Li [33]	gyro, mag	-	accels	289m	1.2%	√	foot
		-		192m	0.9%	√	
Godha [15]	gyro, GPS	-	accels	78m	1.2%	(√)	foot
		-		234m	0.9%	(√)	
Ojeda [39]	gyro	-	accels	524m	(0.5%)	(√)	foot

Table 3.1 Overview: Dead Reckoning Results. This tables combines experimental results of different approaches whenever suitable for this comparison. Note that some values where not mentioned in the work of the authors and are thus marked as not available (*n/a*). The heading column describes the used sensors for heading determination, where gyro and mag correspond to the gyroscope and magnetic field sensor. As there are sometimes some more sophisticated fusion techniques uses, we marked these as advanced (*adv.*). The step detections used are the peak, gyro and ZC methods. We have introduced the main methods for step length estimation, where accelerations at the foot mounted devices correspond to the direct integration of velocity.

3.3 Navigation Interfaces

Most navigation systems use a visual display and audio feedback (e.g. commercial car navigation systems). We will introduce some possibilities how to give reasonable feedback for the different media - these are visual, audio and vibration. All of these media are natively available on all current smartphones.

3.3.1 Visual

Typical visual interfaces work in three fundamentally different ways or combine them. The first shows a map with the calculated route, the user's position and the covered path. For example, this map may be (automatically) rotated to match the current user's heading and allows zooming and panning. The second method to give routing information in a turn-by-turn manner, are arrows. Showing an arrow for the next turn typically includes information about the distance towards the next event. Lastly, giving a pure textual representation may also provide enough navigation information.

However, the map-approach together with turn-by-turn commands via arrows has proven suitable and is well-established in commercial products.

Focusing on pedestrian navigation, another possibility may support the presented DR approaches. Photos of landmarks seem to be the most appropriate method for pedestrian (outdoor) navigation. Results of Goodman et al. [16] show, that a picture-based navigation can lead to less time spent on self-orientation of the user, especially for older people. Stark et al. [52] state, that such images decrease insecurity about the shortest way as well. Unfortunately, this method has the big disadvantage, that it relies on the occurrence of unique landmarks and needs highly detailed models, i.e. suitable images.

Holland et al. [20] give some reasons, why a visual interface may not be suitable in all kinds of pedestrian navigation situations. The strongest argument is, that the user may not be able to give enough attention to a visual interfaces while walking or doing something different.

3.3.2 Audio

In general, there are different varieties for feedback via audio. The verbal form gives navigation instructions via speech (typically either with recorded audio or synthesized speech using a Text-to-Speech (TTS) engine). Guidice et al. [14] investigated such verbal navigation methods and conclude, that minimal geometric information is sufficient for way finding. Finally, Stark et al. [52] show, that if a route is straight without many open spaces, audio navigation seems to be the most appropriate concept.

Besides verbal feedback, different sounds may give suitable information for navigation in a turn-by-turn manner. A different pitch, rhythm and amplitude yield several varieties for sound. We can separate the main navigation information into direction and distance, where some 3D-audio rendering or at least stereo sound with some

drawbacks (need of headphones or two distinct speakers) can encode the direction. Moreover, another encoding would consist of a set of special patterns for commands like turning left or right. The distance towards the next event is often encoded by a "Geiger counter" model, introducing a higher pattern frequency when approaching further towards a destination, e.g. presented by Holand et al. [20]. E.g. Wilson et al. [54] switch both pattern properties.

Soundmarks or ad hoc audio annotations may enhance the audio navigation like proposed by Kainulaien et al. and Wilson et al. [26, 54].

3.3.3 Vibration

We can adopt the concepts of non-verbal audio navigation for tactile feedback, i.e. vibration. Lin et al. [34] introduce a method to encode the direction into different vibration rhythms. Their method encodes the distance towards the next turning event into the tempo of these rhythms which increases gradually while approaching this destination. Evaluation shows the correct perception of these vibration patterns at 99%. For a more distinct direction information, Pielot et al. [41] present an alternative encoding of the direction, whereas the frequency also gives the distance information. Their approach (depicted in Figure 3.3) encodes a heading with two subsequent pulses - either a short pulse followed by a varying longer pulse for directions towards left and the same pattern reversed for headings to the right, whereas the length of the second pulse encodes the difference between current heading and the heading towards the next destination.

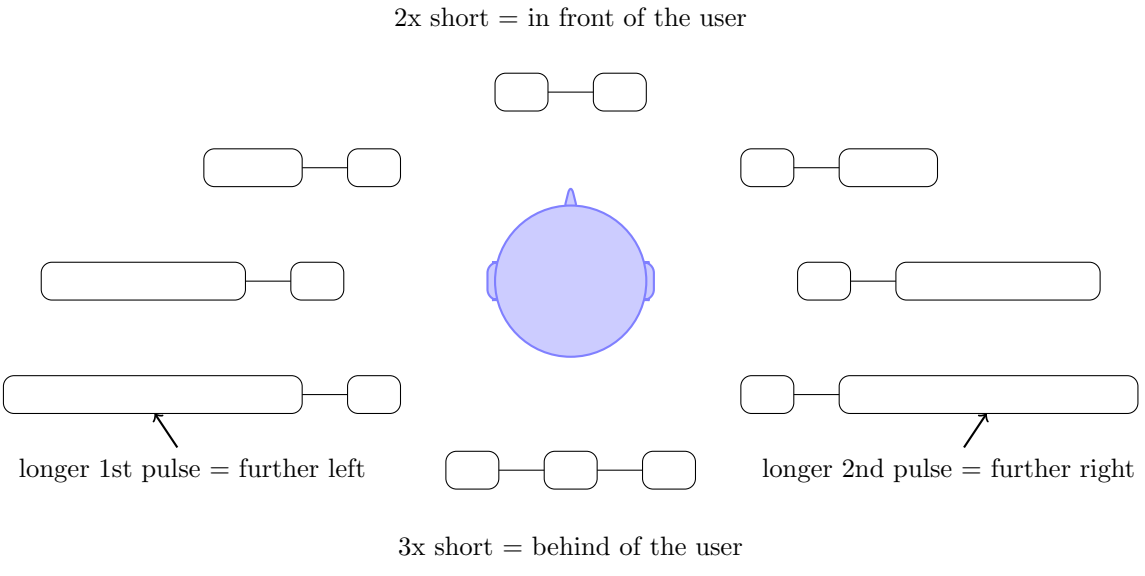

Figure 3.3 Vibration Direction Encoding. In general, two pulses determine the desired navigation direction.
Two short pulses denote the correct heading along the route, three short pulses state, that the user has to turn around. Different length on the first or second pulse denote that the desired navigation direction is either to the left (1st pulse varies) or to the right (2nd pulse varies) of the user [41].

3.4 Summary

We have introduced and discussed the different methods for indoor localization. This includes WiFi aided approaches (3.1) as well as PDR (3.2). The PDR is typically aided by foot-mounted or handheld devices. These methods typically use a heading estimation technique (3.2.2) in conjunction with a step detection (3.2.3) and a step length estimation (3.2.4). The heading estimation usually uses the magnetic field sensor or the gyroscope, whereas the step detection uses the acceleration sensor. Another approach to this has its origin in computer vision. We have seen some fundamental methods to extract movement information out of subsequent images by tracking, estimation the vanishing point or by exploiting the compressed camera video stream. The step length may either be fixed or estimated by a specific model which also uses acceleration data.

PDR results can be improved by adding specific filtering techniques, e.g. Kalman filters (3.2.6). Adding more data like a building map or WiFi information improves the overall performance of the localization (3.2.5, 3.2.7).

Current navigation interfaces usually make use of the devices display. Showing a map and giving turn-by-turn instructions in a visually and verbal via audio are a commonly established way to give feedback. We have shown how to use other media like vibration and how we can encode important navigation information (3.3).

4

Design

We have already given a rough introduction to FootPath and how it works in Section 2.5. This chapter will explain and discuss important design decisions which mean, that we will go into detail on new functional building blocks and how they are integrated with FootPath.

4.1 Overall Design Considerations - Layered Design

The work done in this thesis integrates with the existing FootPath project. We design new components as single modules without interacting too heavily with others to keep things simple. Thus, we introduce a layered design depicted in Figure 4.1. The movement determination only needs sensor data (or camera input). The matching algorithm then uses recognized steps and the corresponding estimated heading to correlate these to a given navigation path. The navigation module will receive new location information and decides whether or not to give user feedback or to recalculate a new route.

4.2 New Model for Step and Direction Detection

We have seen the current method to detect steps and determine the corresponding heading information in Section 2.5.1. However, the implementation relies on the assumption carrying the device in a *null orientation* - or at least a predefined attitude. We want to avoid this constraint to make FootPath able to detect steps and the heading independent from the attitude of the device. This means making it possible to do navigation while carrying the device e.g. in the jacket pocket. Thus, we have developed a new method to detect steps and the heading.

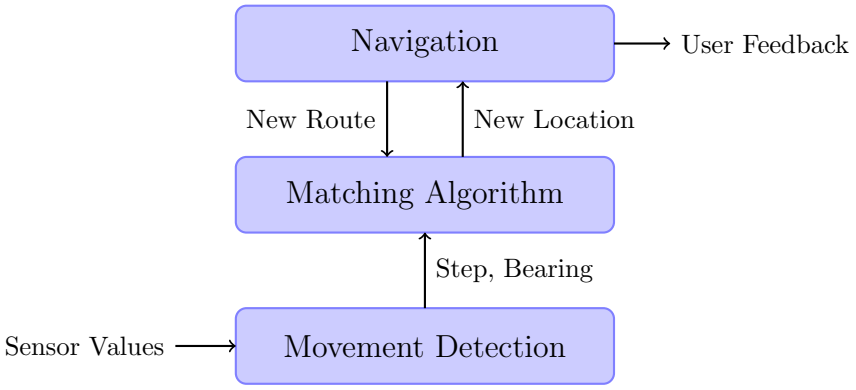

Figure 4.1 Layer Design of (new) Components. We have a separate movement detection, matching algorithm and feedback module which communicate via well defined interfaces for simplicity.

4.2.1 Functional Requirements Analysis

The main requirement is to detect the movement of a user, i.e. the device, independent of the attitude. We do this by following the current approach detecting the user's steps while estimating the current heading. As the movement determination should be self-contained, we do not want to give any external user determined information into the system. Thus, available data sources are only the devices sensors being primarily the gyroscope, accelerometer and magnetic field sensor. With sensor fusion techniques, we can extract more sophisticated data out of this information like linear acceleration, the devices orientation and magnetic north.

In order to make the movement detection realtime capable, it has to work with low latency and as an online algorithm. Furthermore, the memory consumption should be low or at least limited. The online algorithm should be able to react to changes and be adaptive as well. Furthermore, the method should be computationally cheap and configurable.

4.2.2 Magnetic Compass

The current step detection uses the magnetic field sensor to get the user's heading towards north. We can use the magnetic field sensor data directly for calculation of the heading towards north, if the user holds the device in *null-orientation*.

The magnetic field sensor provide a vector with a direction towards north in intrinsic coordinates, this means that the output is not restricted by attitude, but the given values are relative to the device. In order to get the heading towards north, the typical approach incorporates the accelerations in order to get a second fixed direction in world coordinates to calculate a rotation matrix vom instrinsic to extrinsic coordinates. From this, we can extract the current bearing.

However, the reulsting heading usually has two possible solutions. The first one is explicitly calculated and a second one, that will be in the opposite direction (flipped). We provide further discussion on this ambiguity in Chapter B.

In order to get the correct heading, we need to include other data. As we do not want any user interaction for this purpose, we can either try to fit the steps on the map, which will lead at some point to an impossible choice for one of both solutions, or by calculating the actual movement out of acceleration data.

As we want to follow the layer design, including map data here will not be a possible solution. This leads to the fact that we have to calculate on the acceleration data anyways. Moreover, evaluation shows, that a simple approach to integrate the acceleration data to get a rough movement direction for selecting the correct heading solution from the magnetic field sensor is not feasible.

4.2.3 Smartphone Carrying Locations - Survey

In order to concentrate on most typical device carrying locations, we decided to collect information about the usage of smartphones. We asked people about their typical carrying locations (depicted in Figure 4.2) for their smartphones. Furthermore, we asked about possible acceptable alternatives. The complete number of participants is $n = 103$ with $n_f = 28$ females. The participants' ages span a range of 18 to 45 years. This might be due to the fact, that our invitation to this survey has only reached an audience at university. Nevertheless, we assume, that this corresponds to the nowadays smartphone user group and thus, we consider the sample to be representative. Table A.1 and Table A.2 show the complete results for used carrying and acceptable locations.

Rai et al. interviewed in [43] 30 employees in an office about typical placement scenarios. They conclude, that most people carry their phone in their (front or rear) trouser pocket or pouches at the belt. Women mostly carry their phone in their bag and sometimes in a trouser pocket.

Our survey shows almost the same results. Most important is the observation that only 17% of the asked participants never carry their phone in the trousers pocket (front). Another popular location is the jacket pocket. About 45% of the participants are at least sometimes using this location. Moreover, there is no difference in the results for inner and outer jacket pocket. Although the bag or backpack are overall more often used than the jacket pocket, the jacket pocket is more popular regarding only the *always*-usage. Other alternatives are not considerably used.

The trends figured out from the used locations sustains for the question about acceptable locations. The most acceptable location is the trouser pocket with only about 10% giving this location no chance. Close to this is the jacket pocket with a definitely acceptance of about 41% whereas only 22% of the asked people will not ever use this location. Both, rejection of the bag and backpack are at about 38% while the rest splits up equally at a vote for definitely and maybe.

However, differences regarding the gender show in general, that women more often use their bag and men prefer the trouser pocket. This observation also holds for the acceptance of these locations.

1: trouser pocket (front)
2: trouser pocket (back)
3: jacket pocket (outer)
4: jacket pocket (inner)
5: belt
6: arm
7: shirt pocket
8: backpack
9: bag

Figure 4.2 Survey Device Locations. We have put these possible locations for carrying a smartphone to the vote in this survey.

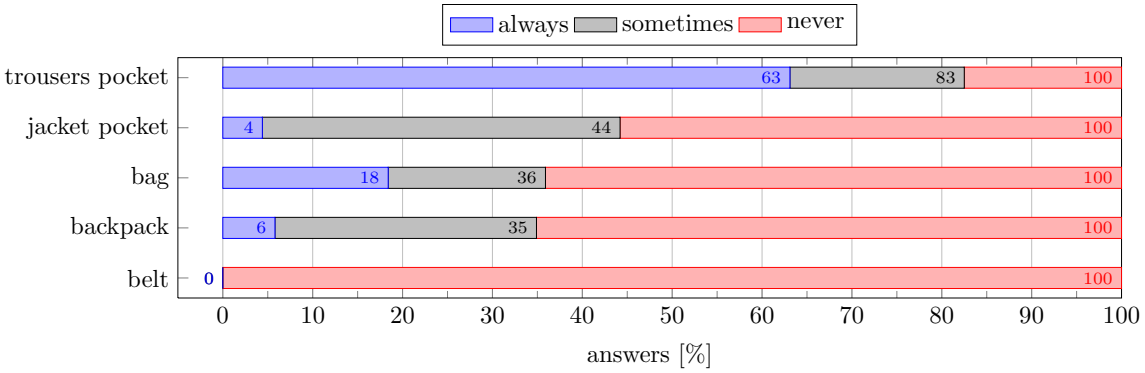

Figure 4.3 Survey Results - "Where do you carry your smartphone?". The blue marked area describes *always*, gray corresponds to *sometimes* and red to *never*. The survey states, that most people carry their phone in their trousers pocket. The bag, backpack or jacket pocket are the most used alternatives.

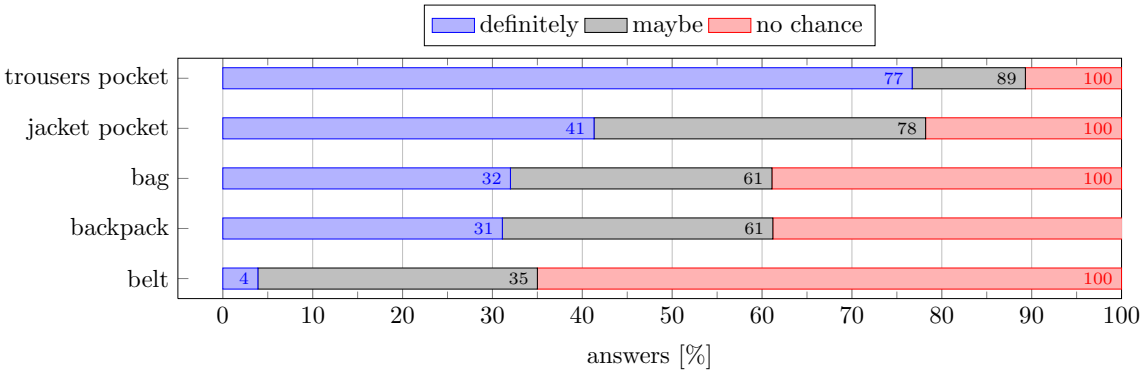

Figure 4.4 Survey Results - "What location would be acceptable to carry your smartphone?". The blue marked area describes *definitely*, gray corresponds to *maybe* and red to *no chance*. The acceptance for the trouser and jacket pockt is high - most people already use these locations. Moreover, bag and backpack are possible alternatives.

4.2.4 First Approaches

As we want to determine steps and the corresponding heading independently from the location of the device, we have to evaluate on an approach to do this task in a reliable way. In this section, we will discuss several procedures which may not give appropriate results.

4.2.4.1 Direct Rotated Linear Accelerations

A naive approach to extract the current devices movement would integrate over the accelerations and calculate the corresponding heading. Evaluation shows that the direct bearing extraction from the accelerations works well in some cases, but it is not feasible in general as the average error on our recorded test data is $\text{err}_{\text{avg}} = 81.38°$ whereas the median is close to this value with errors of $q_{.50} \approx 80°$ and $q_{.75} > 100°$. These results are not feasible for a reliable heading estimation and will most probable not be useful although a particle filter or some other map matching method may be used for correction.

4.2.4.2 Magnetic Field and Accelerations

Kothari et al. [31] use a method for determining a heading from acceleration and magnetic field sensor values which has the critical shortcoming, that their method will yield two possible solutions for a heading as it is not possible to distinguish between front and back - or in other words, the movement direction. This method is also used in official Android sources. We provide detailed information on this approach in Chapter B.

To determine the correct solutions from both, we have to consider more information. This can either be acceleration data or a trial and error approach via a particle filter which tries to match the movement onto given map data. As we want to rely on our layered design, the second method is not feasible. However, using direct acceleration data is not feasible as well. We have seen, that both, average and median, show an error near 90° which will not reliably enable the categorization of current movement into front and back. Thus, we cannot reliably use direct acceleration data to determine the correct heading estimation. We further discuss the topic flipping in Section 6.2.2.

4.2.4.3 Sensor Fused Rotation Vector

With Application Programming Interface (API) level 9, Android introduces a new virtual sensor which combines accelerometer, magnetic field and, depending on availability, gyroscope data. This virtual sensor already uses sophisticated filtering techniques in order to get smoother results. The rotation sensor data's representation are quaternions which introduce more complex mathematics. However, we can calculate a rotation matrix (see Section 2.2.4) and extract the rotation towards the positive y-axis which is equal to the world's heading towards north which intoduces also the shortcoming of two possible solutions (see Section B.2).

4.2.5 Final Model Decisions

As our first approaches do not succeed in reliably determining a heading estimation, we have to use another more sophisticated model. The survey clearly shows to concentrate on two different most popular locations - trouser and jacket pocket. By analyzing the typical linear acceleration patterns (already rotated into the null-orientation) at both locations, we can see a z-acceleration behavior which roughly shows a combination of two different sine-functions, because of the user walking step by step, leading to these acceleration changes in the z-direction. We can correlate each local minima to a finished step placing the foot again and again onto the ground. Depending on the location of the device, we can see a primary and an secondary step (left and right leg) which will differ in amplitude.

By looking into recorded data from a test run, e.g., with the device in the user's hand, we can also find a similar pattern. Inspired by this, we conclude that it is possible to create a generalized model out of this pattern for optimal bearing determination. Figure 4.5 shows a complete overview of our step detection and bearing estimation pipeline.

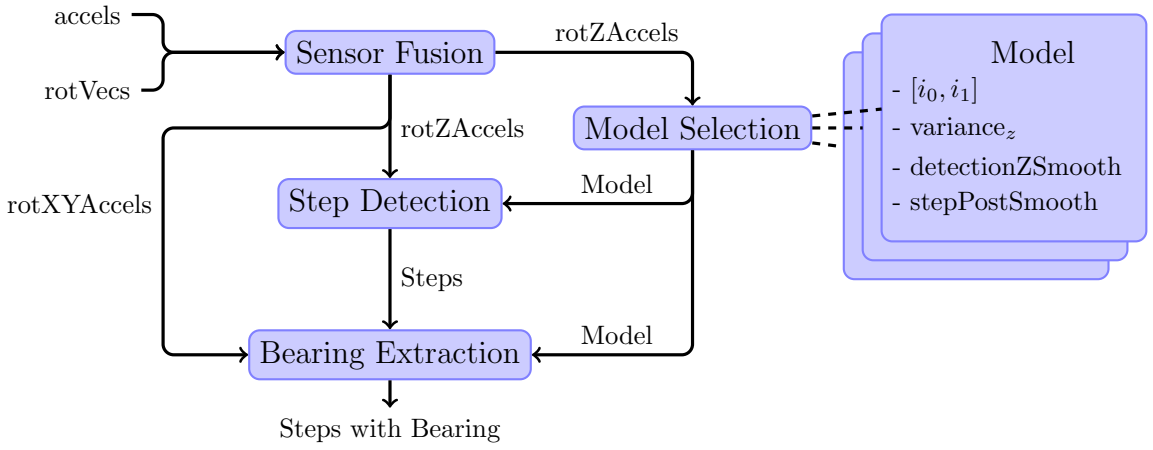

Figure 4.5 Step Detection and Heading Estimation Pipeline. The Sensor Fusion transforms linear accelerations via rotation vectors from intrinsic to extrinsic coordinates. The model selection decides on the rotated z-accelerations on the model to use. Besides valid variances for this model, it contains further information about smoothing for step detection, the interval for bearing extraction and a postsmoothing parameter for the actual heading estimations. Having information about which model to use, the step detection smoothes the rotated z-accelerations and detects local minima which represent a finished step. Afterwards, the bearing extraction conducts from the model parameters, step events and the rotated linear x- and y-accelerations the bearing which may finally get smoothed as well.

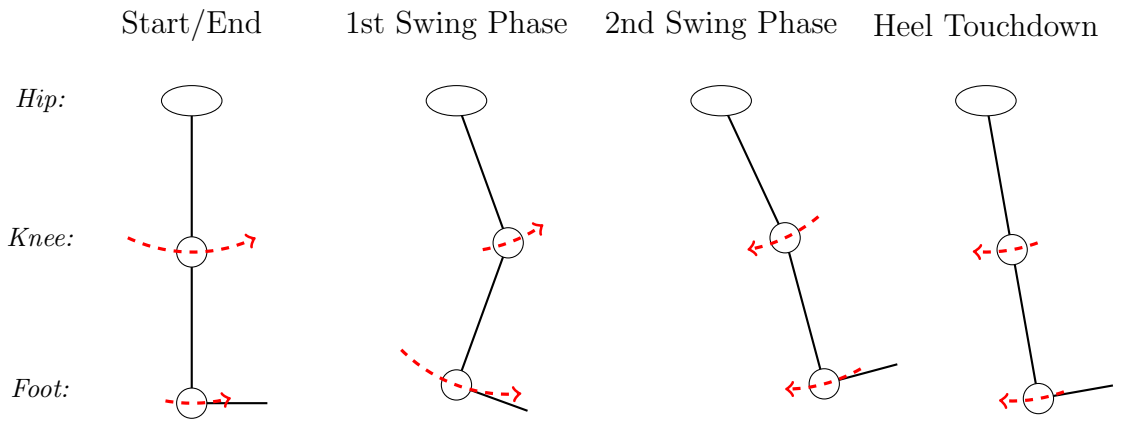

Figure 4.6 Walking Motion - Different Phases. Arrows describe the movement of the thigh and shank relative to the hip. The first phase describes the lifting of the foot. Afterwards the second phase describes the acceleration of the leg into the walking direction where this will reach a maximum until the foot is going down towards the ground again. The complete step will finish by the third phase in which the user places his foot down on the ground again.

4.2.5.1 Generalized Model

Introducing a generalized model with currently two different instances, one for trouser pocket and one for jacket pocket, gives us the possibility to adapt the model more precisely and to get more accurate results. Primarily, we have to find a simple approach to distinguish when to apply which model. The variance in the linear z-acceleration has proven to be useful (see Section 6.1) - other simple features like pure attitude from the orientation values will typically not be sufficient.
Simply said, the movement in the trouser pocket will be greater than in the jacket pocket and the variance will be greater as well. This also enables to switch the used model ad hoc by just recalculating the variance and switching to the corresponding model.

4.2.5.2 Model Design

By analyzing human walking motion, we can split each step into different phases as shown in Figure 4.6. Let us assume that the start of the first phase will be the foot on the ground. Now, the foot moves up to make the next step. After having lifted the foot, the leg will accelerate into the walking direction which depicts the second phase. The third phase describes the deacceleration of the leg and the new placing of the foot again onto the ground. The idea is now to track the accelerations of the second phase as it executes the largest displacement and use these values for bearing calculation.

Step Detection. Minima in the z-acceleration depict a finished single step sequence. In order to segment the sensor data into such step segments, we first smooth the data to filter for noise and higher frequencies. Then we calculate the first derivative of the z-accelerations and search for zero crossings (ZCs) to determine local extrema, i.e., $d/dz = 0$. To ensure only considering local minima, a simple check of

a sign change around such an extrema is sufficient. This approach corresponds to a peak-step (3.2.3.2) detection.

After each detected step, we apply a step timeout depending on the determined model. These values are currently fixed at $300ms$ for the jacket location and $600ms$ for the trousers location. This will lead for the trousers location in detecting each primary step, which describes the step of the leg where the phone is placed. However, these values are chosen by subjective observation while elaborating on the step detection.

Bearing Extraction. We will extracted the bearing from the summed x- and y-acceleration values of a certain interval in each step sequence. Thus, we define relative positions inside the complete step length.
Formally, we have a series of tuples $(t, x, y, z) \in \mathbb{R}^+ \times \mathbb{R}^3$ defining linear acceleration in the extrinsic representation (see Section 2.3). Our model defines the relative boundaries for this interval: $(i_0, i_1) \in [0, 1] \times [0, 1]$ with $i_0 < i_1$, where we will use this interval for actual bearing calculation.
Given step time-boundaries $t_0 < t_1$, the time-step length will be $l_t = t_1 - t_0$. The bearing calculation will use the following functions:

$$\operatorname{atan2}(y, x) = \begin{cases} \arctan(y/x) & x > 0 \\ \arctan(y/x) + 180° & y \geq 0, x < 0 \\ \arctan(y/x) - 180° & y < 0, x < 0 \\ 90° & y > 0, x = 0 \\ -90° & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases} \tag{4.1}$$

$$(x_{\text{sum}}, y_{\text{sum}}) = \sum_{\substack{(t,x,y), \\ t_0 \leq t \leq t_1}} (x, y) \tag{4.2}$$

Due to the fact, that the typical compass bearing defines clockwise north as 0° (depending on used coordinate system), we have to mirror it at the y-axis and rotate it by 90°:

$$\text{bearing} = 90° - \operatorname{atan2}(y_{\text{sum}}, x_{\text{sum}}) \tag{4.3}$$

4.2.6 Method to get Parameters

We have now defined a generalized model for step detection and bearing estimation. To use this, we have to get parameter values for the model to work properly. In general, the dimension for parameters is $[0,1] \times [0,1]$. Usually, we have a bounded samplerate for sensor data. The Samsung Galaxy Nexus for example gives roughly 100 samples per second. Furthermore, we can assume that one step will at most take about 1,5 seconds. This leads here to 1500 samples for a complete step process, where in theory we can select 1500^2 different combinations for the interval $i = [i_0, i_1]$.

These combinations can be further restricted to those, where $i_0 < i_1$ holds, but the resulting combinations remain by far unfeasible. Furthermore, all combinations spanning only a small amount of data may be fitting better on given training data, but will most likely be unstable for generalized usage. Thus, we restrict the possible combinations to a discretized grid with step length 0.05.

Determining Model Parameters. In order to determine the model parameters, we first search for suitable intervals in the z-acceleration variance for a correct model decision. The testing data shows variances for jacket pocket inside a range of $variance^z_{jacket} \in [2, 24.3[$ and for the trouser pocket $variance^z_{trousers} \in [24.3, 100[$.

Secondly, we have to determine the model parameters, i.e. the interval boundaries for bearing extraction. For this purpose, we use least-squares optimization. The target function sums up all squared differences of calculated bearings in contrast to training-data's groundtruth bearing. We will choose the interval with the least error value for our algorithm as it shows best results.

Evaluation shows, that extending the interval to the possibility to contain a step boundary, i.e., $i_{\text{extended}} \in [0,2] \times [0,2]$, does not lead to better model parameters on our training data.

4.2.6.1 Smoothing

It is also possible to smooth the acceleration values before the bearing calculation. Evaluation shows, that pre-smoothing does not lead to better results on our training data. But smoothing the resulting estimated bearings improves performance. Using e.g. an average over more than the last two estimated headings may lead to a noticeable latency. Moreover smoothing is not as easy in the domain of angles $\mathbb{R}_{360}$ which does not simply enable to e.g. take the average in a common way: $\text{avg}(10°, 350°) \in \mathbb{R} = 180°$, but $\text{avg}(10°, 350°) \in \mathbb{R}_{360} = 0°$.

4.2.7 Non-Step updates

The current heading estimation obviously depends on step events. We already discussed the possibility to use the magnetic compass and decided on not using the compass heading by itself. Although we extract the heading estimation from actual accelerations, we can use the magnetic compass for a heading estimation whenever there is no step event, i.e., there is no matching model meaning that there has not been a step event.

However, we can also track the rotation vector and whether we have to apply flipping by 180° to the calculated heading. Thus, we can also give heading information without movement in the correct way independent of movement.

4.3 Navigation Interface

In order to get valuable feedback from the navigation, there has to be a suitable interface. We will discuss the current interface (display). Due to the need of a new interface as the display will not be viewable with the device in e.g. the jacket pocket, we introduce new methods to give feedback (audio and vibration) which require a method to determine what information to give.

4.3.1 Current Interface: Display

As FootPath assumes a usage while holding the device in your hand, the current navigation interface is the device's display. It displays the current map with the user position, calculated path towards the destination and the currently assumed covered path while navigating. This makes it easy for the user to get information about upcoming route changes and recognizable objects on the way. Furthermore, the map also provides information, e.g. about walls, doors and stairs, and shows background-tile images from OpenStreetMap. For convenience, the interface supports zooming and panning via well known gestures.

4.3.2 Overlay Graph and Interestpoints

The current map model is FootPath-internally represented as a graph. In more detail, we split the data of the source into two groups. One group contains all wall-related information as well as special areas such as stairs and elevators. The other group contains all pedestrian-passable ways in a graph structure.

We perform path-determination for navigation from one room with an implementation of Dijkstra's algorithm which calculates the shortest path between start and destination. The algorithm then returns a list of all nodes on this path.

To follow the layered-design, the navigation-interface should be self-contained and not depend on other modules of the FootPath project. We instantiate the navigation module with information about the desired feedback-type, the map data as well as the destination for navigation. This means, that currently the only interface between

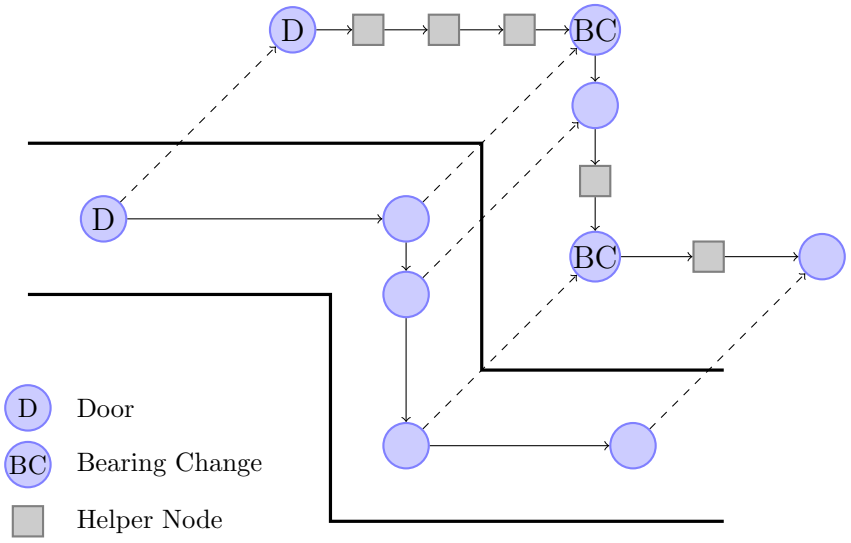

Figure 4.7 Example - Actual Path and corresponding Overlay Graph. Note that the overlay directly carries over the door-property of the actual node. An additional calculation step adds bearing changes.

the path-matching-module and the navigation will be a single listener which gets informed about a new estimated user location.

Consequently, the navigation is then responsible to check if the current path remains correct or if we have to recalculate it. The module is capable of doing this, but the current matching algorithms do not support an ad hoc path update for the time being.

As the navigation will only get an estimation of the user's position, there must be an efficient method to correlate this position with the calculated path. To do so, we decided to only model a path as a list of `ActionNode`s as an overlay graph. Figure 4.7 illustrates an example. Any efficient datastructure such as a quadtree then enables a fast lookup for the nearest path-node towards a given location. To prevent the algorithm to consider a wrong path node, we introduce a threshold for a maximum distance. Exceeding this threshold results in regarding this path as useless and will trigger a recalculation.

However, typical nodes may have a large distance. Thus, we introduce helper nodes in between at path creation to keep this threshold small.

4.3.2.1 Interest Points

In order to enables feedback, like "pass 3 doors on your left", we include a variety of information into the overlay graph where the actual navigation feedback depends on. More precisely, we add each "event" to the overlay graph nodes. Thus, after having calculated a new path towards a destination, we precompute such Interest Points. This information enables creation of actual navigation feedback.

We can collect some information directly from the original map nodes, e.g. doors as they have a certain key-value pair in the source XML data. We calculate other information like a bearing change or stairs each time we have calculated a new path as this will not remain equal.

We introduce precomputation filters to accomplish this task. These filters traverse the path and insert `Interest Points`. The current set of filters allows easy extensions by adding new ones which may also involve the creating of new types of Interest Points.

Current filters are the following:

- **BearingChange**. We track the current bearing from node to node over the whole path. We consider a bearing change to happen whenever the bearing differs by an amount greater than a certain threshold (45°). Depending on the previous bearing, the change will be categorized to either *left* or *right*. Moreover, we provide the exact bearing-change in degrees.
- **Doors**. We model doors explicitly in the source map data which allows for an easy addition of this information as an InterestPoint into the overlay graph. This may also include a name if the door belongs to a certain room.
- **OtherPossibleDirection**. Some nodes of a passable way have a degree greater than two. This means, that there is at least one more possible direction - not belonging to the actual calculated path. We calculate the bearing into such a direction and categorize it into *left*, *right* and *none* indicating the bearing change for this direction.
- **NearDoor**. At some passages on the way through a building, there will not be any special Interest Point like a bearing change or door for a longer distance. To get valuable information anyway, the number of doors to pass (on the left or right) may be of interest. If there are other possible directions at a certain node, we will track the path into the other direction inside a given bearing threshold as long as a maximum distance has not exceeded to check whether there is a door in this direction. This ensures a direct line of sight without the need to compute costly line intersections with walls etc.
- **Stairs**. Stairs are only modeled at the source graph edges. Thus, if an adjacent edge of a node is indicating stair, the node will get a corresponding Interest Point.
- **LevelChange**. The source graph has level information for all nodes. We will store level changes as an Interest Point. Important information here are the floor levels and the direction *up*/*down*.
- **Crossing**. Each time when there are possible direction *left* and *right*, the current node may be a crossing. To make sure this is a real crossing and there are not only two rooms in both directions, we check if the path will be longer than a certain length without a door in between.
- **Entrance**. The source map data introduces a specialized tag for building entries. As these denote a good recognition value for humans, we search for entries in the same way as in the NearDoor filter.
- **OtherTag**. To keep things open and to enable specialized information, we introduce the possibility to define own tags at the nodes in the map data. We add them in the Interest Point filtering step into the overlay. We allow checking for these tags for presence or certain content later on.

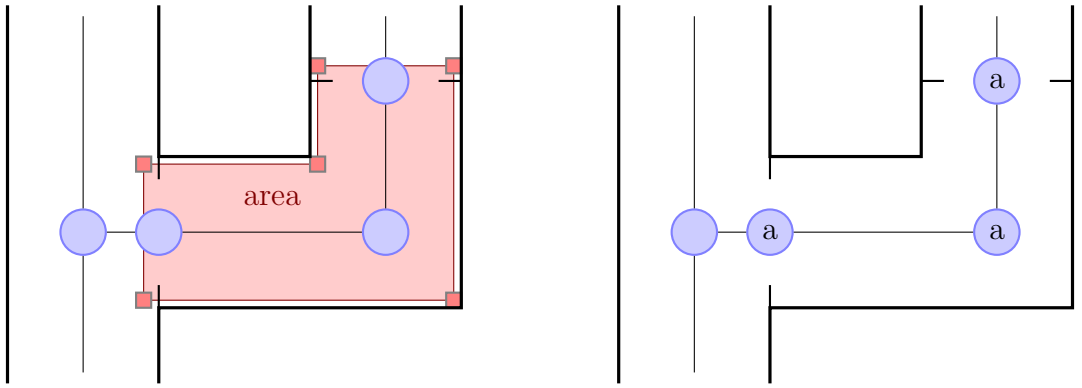

Figure 4.8 Modeling Building Areas.
Left: A polygonal area defined as a graphway.
Right: The area modeled indirect inside the passable way nodes where each Node inside the area will get a reference for belonging to this area.

The presented filters are straight forward. Specialized building parts or objects are more challenging, but usually have a remarkable recognition value like a glass door, a well or some kind of exposed arts - or even a glass corridor. In general, we can subdivide these things into building parts to pass through and recognizable objects.

Objects. One possibility is to model such objects via some other tag. We do so by involving OtherTag Interest Points. They allow natural usage like other Interest Points as they just might occur on a route. This solution fits well in the current configuration, but needs some extra work at map creation time (obviously, such nodes have to be connected to the passable way graph). In the future, it should be considered to just add such objects as single non-connected nodes or areas and to do some more sophisticated polygon-membership tests regarding walls etc. to get automated connections to the passable way graph we are working on.

Building parts. We have to handle building parts, e.g. a classcorridor, differently. Building parts span over a complete area, thus we cannot directly model them via one single node. This means we have to represent them by an area. One way would be to model them directly inside the OSM data like stairs as a graphway, or we may include them inside the passable ways for walking. Setting a certain key-value pair on each node of the walking way trough the area might be a valid solution. Figure 4.8 shows both possibilities. Modeling areas will be more convenient at map creation time, but also introduces the need of precomputation. Modeling directly inside the nodes will not have this disadvantage, but obviously needs more work at map creation. Moreover the direct modeling also makes distinction of several areas at the same time more complicated.

Besides these building part areas, they may be objects to pass in a “via” relation. The prominent example of doors has already been considered directly. However, there may be countless other scenarios which may be modeled directly with a certain tag inside a node like for objects.

4.3.3 Alternative Interfaces

The current display (see Section 2.5.4) will not be useful as a navigation interface in our scenario where we do not want to carry the device in our hands. Thus, we have to use a medium other than a display (optical). Moreover, other media will make the navigation useful to handicapped people like blind or deaf persons.

Todays smartphones offer other media to provide user feedback. For example, we can use audio, vibration or a combination of both as a feedback channel.

4.3.3.1 Audio

There are different methods to use audio for giving feedback (see Section 3.3.2). However, speech synthesis is available in a variety of languages. We can just use synthesis functionalities as a black box for audio-output of arbitrary text. There remain two questions - what information do we want to give to the user and when.

In general, the amount of information to give, should be very limited. Because of limited short term memory of the user, he will not keep all information - or will at least not be capable to extract the relevant information if the amount of data is too high [48]. This leads to a fundamental guideline to only give feedback in cases where there is some valuable event (Interest Point) on the current upcoming path in a range of less than about 15m. Moreover, a single short sentence with an additional rough distance estimation encoded e.g. into words (e.g. a mapping function $distToText : \mathbb{R}^+ \rightarrow \{\text{now}, \text{soon}, \text{later}, \emptyset\}$) should encapsulate this information. There may be cases where multiple events occur right after each other like a bearing change after passing a door. In such cases, we can aggregate this information for feedback in order to not surprise the user with a new command right after the previous one just happened.

However, giving the user all available information is not feasible, as the amount is often too high and most of the information is currently unimportant. The solution to face this, is to give static priorities to events in conjunction with their current distance. This means, that we can ignore all events with less priority, if there is an event with a higher priority. E.g., if the current path contains stairs and a level change in the upcoming next 15m, the information of bearing changes on the stairs is not an important information. Instead a simple command to just go up-/downstairs to a certain other level will be appropriate.

4.3.3.2 Vibration

The vibration feedback is very limited. Primarily, the amount of possible encoded information into vibration is small. A morse-code encoding would be a valid method, but on the one hand, the user must know such a language by heart and on the other hand, vibration has a high impact on the step detection (i.e. acceleration behavior of the device). Although this vibration pattern may be filtered out by the step detection, we conclude that this will usually not be feasible.

Thus, we have to concentrate on a small amount of short and different well distinguishable vibration patterns to indicate most important information like an error or

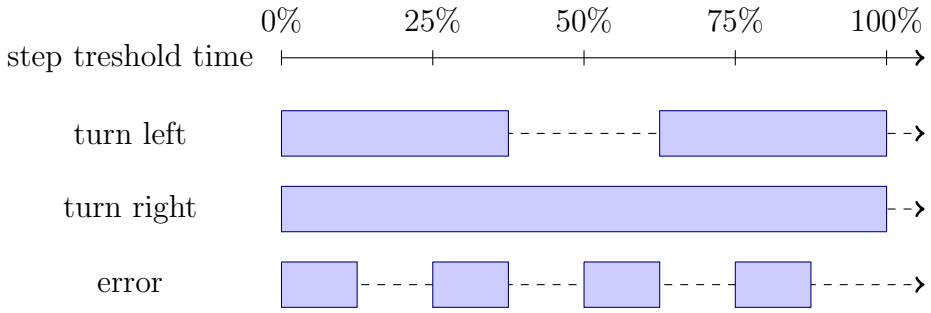

Figure 4.9 Example Set of Vibration-Feedback Patterns. The colored boxes depict the periods, where the vibration is turned on. In order to avoid influences on the step detection, we limit these sequences to the step-detection threshold time.

bearing change towards left or right. Usually, whenever a new position estimation is available, an algorithm will check for the need of giving feedback, i.e. in Pedestrian Dead Reckoning (PDR) soon after a new detected step event. To avoid influences on the step detection, we have to restrict the total vibration pattern length to the step-detection threshold. Figure 4.9 shows a possible set of patterns.

4.3.4 Metric Linear Temporal Logic

Our navigation module provides an overlay graph containing Interest Point information and returns the user's current position on this graph at each location update from the matching algorithm. For feedback creation, we have to evaluate the current scenario out of this graph starting at the given user position. However, this may be a complex task. In order to allow for easy scenario determination, we introduce a special logic, Metric Linear Temporal Logic (MLTL).

Definition 1. Paths. A path π describes a finite sequence of elements $\pi = v_0 \dots v_{n-1}$ and a set of atomic propositions AP which allow evaluation for each element. We define the language of such an element $L(v)$ as the set of all propositions $p \in AP$, where $p(v)$ holds. Furthermore, we denote the length of the path as $|\pi| = n$ [18].

Definition 2. LTL Syntax. Given a set AP of atomic propositions, Linear Temporal Logic (LTL) has the following abstract syntax, where $a \in AP$ [55]:

$$\varphi ::= a \mid (\varphi \wedge \varphi) \mid (\neg\varphi) \mid (\mathcal{X}\,\varphi) \mid (\varphi\ \mathcal{U}\ \varphi).$$

Definition 3. LTL Semantics. We define the semantics of LTL by the satisfactional relation $\models_{\text{LTL}}$ (short $\models$) which evaluates LTL formulas in the context of paths as follows [55]:

$$\begin{array}{lll}
\pi \models a & \textit{iff} & a \in L(\pi_0) \\
\pi \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \pi \models \varphi_1 \text{ and } \pi \models \varphi_2 \\
\pi \models \neg\varphi & \textit{iff} & \pi \not\models \varphi \\
\pi \models \mathcal{X}\,\varphi & \textit{iff} & \pi_1 \models \varphi \\
\pi \models \varphi_1\ \mathcal{U}\ \varphi_2 & \textit{iff} & \exists j \geq 0 : \pi_j \models \varphi_2 \wedge \forall 0 \leq i \leq j : \pi_i \models \varphi_1.
\end{array}$$

Lemma 4.3.1. *Additional Logic Operators. With the given syntax of LTL, we define new operators with the following semantics:*

$$\begin{array}{lll} \pi \models & \varphi_1 \vee \varphi_2 & \textit{iff} \quad \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\ \pi \models & \varphi_1 \rightarrow \varphi_2 & \textit{iff} \quad \neg\varphi_1 \vee \varphi_2 \\ \pi \models & \textit{true} & \textit{iff} \quad (\neg\varphi \vee \varphi) \wedge |\pi| > 0, \textit{ where } \varphi \in AP \\ \pi \models & \textit{false} & \textit{iff} \quad \neg \textit{true} \\ \pi \models & \mathcal{F}\,\varphi & \textit{iff} \quad \textit{true}\,\mathcal{U}\ \varphi \\ \pi \models & \mathcal{G}\,\varphi & \textit{iff} \quad \neg\,\mathcal{F}\,\neg\varphi. \end{array}$$

Definition 4. Metric LTL. MLTL is a superset of LTL. We will extend a path π by a metric-function d : element $\rightarrow [0, \infty[$ where its values define an ordered sequence, i.e. $v_i, v_j \in \pi, i < j$ with $d(v_i) \leq d(v_j)$. Furthermore, we will extend the logical operator $\mathcal{U}$ inductively by a metric-interval $[k_1, k_2]$ with $k_1, k_2 \in [0, \infty[\wedge k_1 \leq k_2$ by the following definition [55]:

$$\begin{array}{rcl} \pi & \models & \varphi_1\ \mathcal{U}^{\,[k_1,k_2]}\,\varphi_2 \\ \textit{iff}\quad \pi & \models & \begin{cases} \varphi_1\ \mathcal{U}\ \varphi_2, & \text{for } [k_1,k_2] = [0,\infty) \\ \varphi_2, & \text{for } [k_1,k_2] = [0,0] \\ \varphi_1 \wedge \mathcal{X}(\varphi_1\ \mathcal{U}^{\,[k_1-d,k_2-d]}\,\varphi_2) & \text{for } k_1 > 0, \text{ where } d = \mathrm{dist}(\mathcal{X}\,\pi) \\ \varphi_2 \vee (\varphi_1 \wedge \mathcal{X}(\varphi_1\ \mathcal{U}^{\,[0,k_2-d]}\,\varphi_2)) & \text{for } k_1 = 0, k_2 > 0, \text{ where } d = \mathrm{dist}(\mathcal{X}\,\pi) \end{cases} \end{array}.$$

Definition 5. Metric Finally and Globally. We will extend the operators $\mathcal{G}$ and $\mathcal{F}$ by a metric interval as well. Their definition is the following:

$$\begin{array}{lll} \pi \models & \mathcal{F}^{[k_1,k_2]}\,\varphi & \textit{iff} \quad \textit{true}\,\mathcal{U}^{\,[k_1,k_2]}\,\varphi \\ \pi \models & \mathcal{G}^{[k_1,k_2]}\,\varphi & \textit{iff} \quad \neg\,\mathcal{F}^{[k_1,k_2]}\,\neg\varphi. \end{array}$$

Figure 4.10 depicts some simple examples on evaluation of MLTL formulas. We provide more examples in Section C.

4.3.4.1 From Overlay-Graph-Paths to MLTL-ready Paths

It is important to determine in exactly what special scenario a user currently is and how to guide him best through the building with natural and well-understood descriptions and commands. In comparison to the vibration feedback, the audio interface is well suited to give extended information. The amount of information which can be encoded into vibration-patterns is restricted and not suitable to give detailed feedback, whereas a single sentence via audio may contain sophisticated navigation hints.

The user wants to get aggregated, accurate, useful information. Moreover, it is desirable giving only the most important navigation hints over a short period of time. As this is a whole new research topic for e.g. psychologists, we will not focus on "good" rules here, but we introduce the MLTL to enable people without profound java-programming skills to build rules for user feedback. The logic may introduce initial hurdles, but as it is very domain-specific, it aims directly towards the target of determining the user's current scenario.

The introduced InterestPoints are straight-forward and easy to understand. To let users evaluate the upcoming remaining path for navigation in a simple way, we show how the MLTL concept matches our requirements.

MLTL to Paths. To make use of MLTL formulas to evaluate real-world paths, we will reduce the real-world paths to those being usable with our logic. As this reduction is straight forward, we will only explain the mappings.

A real-world path consists of a finite sequence of nodes $\Pi = V_0 \dots V_{n-1}$ where V is of type `ActionNode`. Each of these nodes has a set of type `InterestPoint` I which may be empty. Furthermore, each node has real world coordinates (Latitude, Longitude, Building level) which enable a distance measure. For using them in our logic, we use the same nodes as a path $\pi = v_0 \dots v_{n-1}$ and a set of atomic propositions for all possible Interest Points $AP = \{$`NearDoor`, `BearingChange`, `LevelChange`, `Door`, `Stairs`, $\dots\}$ which evaluate for each node V_m if a corresponding InterestPoint i is present in its set. We define the metric for a certain node v_m as follows: $d(\pi_m = v_0, \dots, v_m) = \sum_{i=0}^{m} \text{geoDistance}(v_i, v_{i+1})$ which represents the geographic distance for the complete path to the node v_m if the nodes are on the same level.

Whenever a path contains level changes, this metric will only be an approximate measure as the nodes are not on a plane. However, typical paths through buildings solely contain level changes and the occurrence of a level change (via stairs, elevators or ladders) will be more remarkable than an exact distance measure for navigation purposes. Moreover, the modeled map only gives level information which makes it impossible to give an exact measure. Although it would be possible to give a better estimation, we simply ignore this case. Subsequent nodes with the same latitude/longitude coordinate on different levels have a distance of zero in terms of our metric.

Usually, evaluation of such rules will follow some priority and the whole scenario-determination chain will abort at some point. It is possible to exploit this fact to exclude multiple scenarios for the rules with a lower-priority or to introduce hierarchical rules. Furthermore, knowing the exact evaluation implementation enables to create equivalent formulas which will be processed faster.

Abstract information. In most cases, it is not enough to determine the current abstract scenario. The InterestPoints as such give abstract information e.g. of a bearing change, but do not specify directly into which direction. The Interest Point entities contain detailed specifications. Therefore, we decided to give optional reference labels for atomic propositions enabling efficient access afterwards. This also includes the distance of a matching Interest Point.

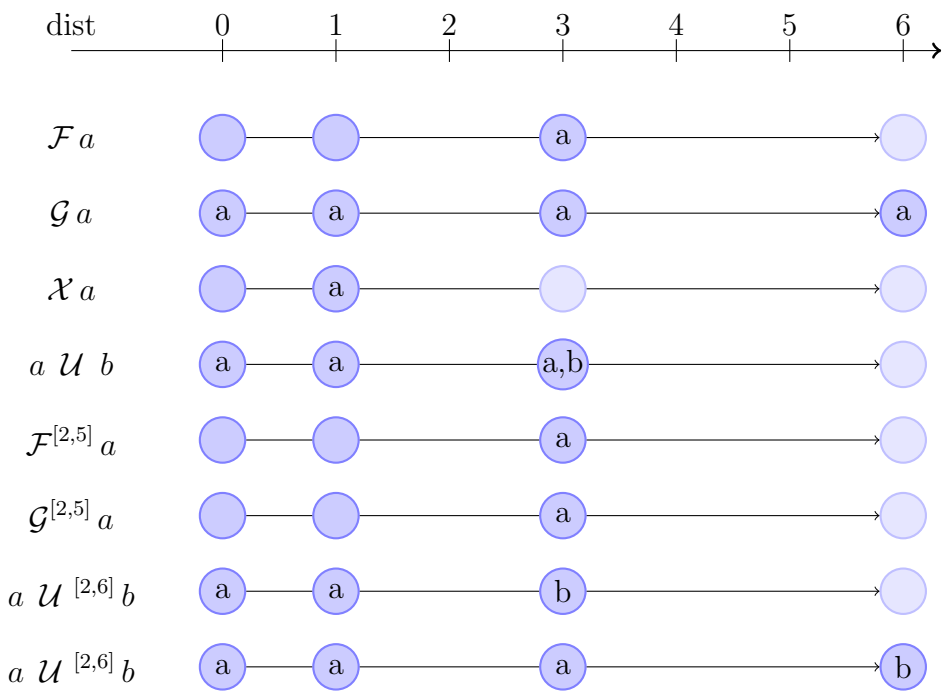

Figure 4.10 Example (M)LTL Path Evaluations. All formulas get evaluated to *true* here.

4.4 Summary

In this chapter, we have presented our main design decisions. The main part consists of a new heading estimation technique which enables the walking-direction determination at new carrying positions, i.e. non hand-held, of the device (4.2). We round this up by a new navigation interface as the current one using the display is not usable for these new locations (4.3).

We discuss different possibilities for heading estimation in general which have not always succeeded to be reliable in our evaluation (4.2.4). For this reason, we deduct from our survey about smartphone carrying locations, that the trousers and jacket pocket are most popular locations (4.2.3). We introduce a new general heading estimation model which bases on typical walking behavior patterns (4.2.5.1). Furthermore, we show how to distinguish between both locations by variance in z-Acceleration (4.2.5.2) and how to get concrete model parameters via optimization on collected training data (4.2.6). As this approach relies on movement of the user, we further introduce a method for determining the heading while not moving in conjunction with the new heading-estimation (4.2.7).

Due to the need of a new navigation interface, we present a framework for path-modeling and an efficient datastructure for this purpose (4.3.2). For giving valuable feedback, e.g. via vibration or audio, we enable non-skilled programmers creating feedback rules with a logic, MLTL, making the current user's scenario determination easier (4.3.4).

5

Implementation

We have implemented the new step detection and bearing extraction in two different phases. At first, we collected sensor data of different persons and analyzed them in MATLAB afterwards. Furthermore, we tested a prototype step detection and heading estimation. We ported the best results regarding evaluation in terms of total error to Java for integration into FootPath.

As step detection and heading estimation is not the only module of this work, we set up a navigation framework for different interfaces (textual, audio and vibration).

5.1 MATLAB

MATLAB in general gives good opportunities for rapid prototyping for any data-related application in a numerical sense as it provides an extensive API for several functionalities. MATLAB is highly specialized for computations with matrices. Moreover, it provides easy to use methods for visualizing results [4].

Each run of sensor collection contains information of several internal sensors. These are the accelerations, gyroscope, linear accelerations, magnetic field, GPS (latitude, longitude, bearing and accuracy) and air pressure. We combine these different categories in a single wrapping object. Having the sensor information, we deduct acceleration and magnetic field values in extrinsic coordinates using the rotation vectors.

All data series allow direct access on values of a certain time interval and sampling rate for convenience. Furthermore, we have employed a module for simple filtering operations with different kernels like gauss, binomial, average, integration and derivation, whereas these kernels introduce a parameter for width or order.

Moreover, all different kinds of sensor data allow access to specific properties like magnitude for acceleration, distance between two series of bearings, rotation matrices for rotation vectors, azimuth for geomagnetic data (heading towards north) and characteristic statistical numbers for calculated errors data-series.

Besides the implementations of zero-crossing (and autocorrelated) step detection and heading estimation by our model, we also introduce a simple method for optimizing model parameters which takes possible model parameters, calculates results and uses squared errors towards predefined ground truth as its metric.

5.2 From Offline to Online

The algorithms in MATLAB follow an offline approach. This means, that complete sensor data information is available even before runtime. In contrast to this, an online approach will get subsequent new sensor data over time. There are fine, but differences at both approaches. As the offline method has knowledge about the whole data, e.g. we can compute the variance of the complete data series. The online method will only have local knowledge from the past until the current timestamp of the data.

The conversion of the MATLAB offline algorithms into online Java algorithms is not a trivial task. We have to consider the mentioned drawbacks of online algorithms precisely.

Instead of calculating filters etc. directly on the whole data, the online sensor fusion algorithm runs periodically (currently each 1.5 seconds) on collected sensor data, whereas we store new sensor values whenever they are available.

We have seen, that the algorithm works with several fused data series. Thus, we separate the fusion steps into different modules. E.g. at first we calculate rotated sensor values to get their extrinsic representation. Afterwards, we extract the z-accelerations as we use them for step detection. At last, we do the step detection and heading estimation.

5.2.1 Time Bounds

It is not feasible to store all sensor values since the start of the algorithm. This means, that we have to introduce time bounds. Due to the nature of our model, we only need a certain interval of past sensor values. A time of roughly three times the periodicity has proven to be enough.

To enforce the time bounds, each time the different algorithm-modules run, we do time bounding which deletes all values being old enough. This may also introduce problems with the Java runtime. Due to automatic garbage collection, each time we delete all references to an object, the garbage collector deletes it from memory at some time introducing overhead. Currently we do not see the need for reusing certain sensor-value wrapping object to prevent garbage collection, but it would most probable result in less overhead.

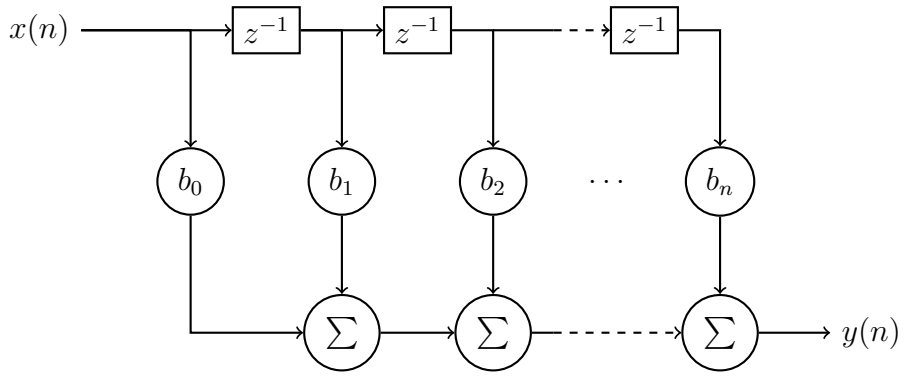

Figure 5.1 Finite Impulse Response (FIR) Filter. Usually, we can express such filters by a single kernel vector defining the coefficients b_i.

5.2.2 Limited Knowledge

We have already introduced the restriction of limited knowledge. This is a disadvantage in general, but in this case, in terms of step detection, it is not. Typically, the training data only shows values while the user is walking. Due to the global distinction of models by variance, we assume that the user is walking all time. Of course, our training data does not have other behavior, but it may be the case in practical examples. Due to this limited knowledge, calculating the current variance in a resulting sliding windows approach gives two advantages - at first, if the user stops walking, the variance will decrease rapidly such that this model distinction value will give the information of the user's state "stopped". Secondly, we can adapt naturally and automatically to model changes - and thus, device placement changes.

5.3 FIR vs. Convolution

In MATLAB we use convolution by a kernel H. The discretized convolution of such a kernel with a certain function $s(t)$ implement the following equations:

$$(s \circ h)(n) = \sum_{m=-\infty}^{\infty} s(n-m) \cdot h(m). \tag{5.1}$$

We can simplify this due to the fact of having a finite kernel:

$$(s \circ h)(n) = \sum_{m=-|H|/2}^{|H|/2} s(n-m) \cdot h(m). \tag{5.2}$$

This means, that for all possible values of $s(t)$, we calculate a weighted sum in an interval of $|H|$ values around each t with the kernel's factors.

As we have an online algorithm, we cannot do this for the newest values as data is missing (we have a causal signal). Thus, we can change the convolution into the following method which describes a Finite Impule Response (FIR) filter shown in Figure 5.1:

$$\widetilde{(s \circ h)}(n) = \sum_{m=0}^{|H|-1} s(n-m) \cdot h(m). \tag{5.3}$$

To transform this FIR result into the convolution, we have to delay the timestamps by exactly $|H|/2$:

$$(s \circ h)(n) = \widetilde{(s \circ h)}(n + |H|/2). \tag{5.4}$$

Note, that both, the FIR and convolution approaches here are discrete variants. This generally means, that the filtering works on all data series, but explicitly disregards any time information. This is no problem as we can correlate the data entries to a time - but problems arise, if the sampling intervals of these values change or are not equidistant. This leads to the main restriction of equally sampled values in this context.

5.4 Actiongraph

The actiongraph holds current routing information and supports giving navigation instructions. Thus, each time we set up a new route, we create a separate action graph as well. To do so, we take the route's nodes from the map and then traverse them with filters which calculate Interest Point information.

Due to our layered design, the navigation module must work independently. As seen, this module will only receive new current estimated user positions. In order to get the corresponding position on the route easily, we determine the nearest node from our actiongraph. To do so in an efficient manner, we do not search along nearest distance towards edges, but only nodes of the actiongraph which discretize edges. For this reason, we introduce extra nodes along each edge which in turn will represent this edge. The datastructure backing up the graph is a quadtree. This quadtree consists of recursively defined quads which in sum are responsible for a complete area. Each quad contains a list of nodes with certain coordinates.
We will use a latitude/longitude to 2D-pixel mapping function (which are usually used to transform a geo-location into pixel values for the map display) as a metric for the quadtree x- and y-coordinate.

By now, we can only distinguish between different geographic locations, but have not considered multiple possible building levels. For this reason, we introduce such a quadtree for each building level.

5.5 MLTL

We represent the Metric Linear Temporal Logic (MLTL) logic as proposition objects each containing a method to evaluate a certain path. In order to keep track of the current metric (distance), we give the current distance value right into this evaluation method.

In general, the propositions separate into primitive propositions for Interest Point checks and language propositions defining the operators like logical and, until and finally. According to the theory in Section 4.3.4, we define the operators using the implemented until.

Moreover, we store important backreferences to certain Interest Points into a hashmap of matching nodes. This means, that we store a reference in a hashmap by user specified keys to an Interest Point whenever we find a holding primitive proposition for his point. Furthermore, if the user does not specify a key, we automatically use a random UUID.

For evaluation, we traverse each node on the given path in a recursive descend approach.

MLTL Overhead. The abstracted logic introduces computational overhead. Naive approaches will define a recursive-descend evaluation introducing exponential complexity. But we can assume that the targeted instances in this setup will remain small, as the density of nodes on a path is on average restricted and evaluation abort at typical restricted distances of only few meters, e.g. 15m, because anything beyond does not contribute to reasonable feedback.

5.6 Class Overview

To give an overview on the complete implementation, we will present some simplified class diagrams and give information about interaction and functionalities. The major components are orientation determination and navigation, where the orientation gets its own external library.

5.6.1 Orientation

Figure 5.2 shows a simplified class diagram. The main single external interface is the `AdvStepEventListener`. It provides a method to notify new detected step events as well as listener functionalities for sensor data given by the Android system.

5.6.1.1 Data Collections

A data collection stores either raw sensor data or fused data. As most sensors typically return multiple values at a time, these values get wrapped into an extra object. For efficient access, we store references to these wrapping objects twice. On the one hand, we provide a list of these values sorted by timestamp. On the other hand, we give sophisticated access on the sensor-data by timestamp via a TreeMap. Although typical different sensor data types seem (for some internal Android or hardware reason) to have the same timestamp for each time they arrive in the Android system, we cannot ensure this fact which is the main reason for the mentioned mapping approach.

5.6.1.2 Datamanager

We store all data collections at the datamanager. To distinguish between data series, they simply get a unique name. The manager provides methods to add new sensor data and to get certain data. Moreover, the manager object also gives the opportunity to trigger sensor fusion techniques and do time bounding on the contained data collections.

5.6.1.3 Sensor Fusion

Each sensor fusion technique will have its own identifier for the datamanager as well. In order to do sensor fusion in a hierarchical manner, each sensor fusion instance provides a priority which enables to define the sequence in which the fusion takes place. E.g. at very first we want to get an extrinsic representation of our data series which means, that this transformation will get the highest priority, whereas the step detection depends on this fusioned data and thus, gets a lower priority.

In general, the datamanager has the responsibility to trigger the different fusion instances and results in the execution of a calculation method. Here, we do the main computation which may be simple rotation of acceleration data by the rotation vectors - or even more complicated step detection. The most critical job here is to keep track of already evaluated information. For this reason, we store the last processed timestamp. However, we have to do further processing and filtering of data with care. The FIR-simulated convolution needs to get some data before the actual timestamp we want to compute as the timestamps get delayed.

5.6.1.4 Step Detection

To get a better impression of how such a fusion process works, we will explain the step detection in more detail.

At first, we demand all used data collections from the datamanager - these are the steps and extrinsic linear z-accelerations (and later on the extrinsic linear accelerations). To determine which model we will have to use, the variance of the z-acceleration series gets calculated and compared against available model parameters. If there is no matching model, we are finished - most likely, the variance was too small which leads to the assumption that there has not been any step event. Otherwise, we smooth the accelerations by the defined width of the matching model in order to detect zero crossings. Remember: We have to define the bounds for the smoothed accelerations carefully. Our implementation of the zero crossing utility already allows us to set a threshold after each crossing. The actual calculation of the zero crossings (ZCs) is simply done by applying a first derivative kernel on the data and afterwards analyzing all zero entries and sign changes to just get local minima.

Now that we have all assumed step events, we will continue with the heading estimation. The given model already defines parameters about the relative interval of important acceleration for bearing extraction. Iterating through all step events allows summing up linear x- and y-accelerations within each of these relative intervals and then calculating the actual estimated bearing towards north.

Note, that we decided to not keep track of different smoothed linear z-accelerations series as the sampling rate may change. Moreover, just smoothing needed acceleration values on demand may save some computation time, especially as there might be a lot different models in the future.

5.6.1.5 Non-Step Updates

In order to get a heading estimation while not moving, i.e. without step events, we cannot use acceleration data, as there is no acceleration while standing still or just rotating. For this reason, we have to use the magnetic field sensor or rotation vector for a heading estimation. Due to simplicity and less overhead, we decided to use the rotation vector directly as the magnetic compass combined with the acceleration data would introduce the need for tracking more sensor data. Moreover, the heading from the rotation vector provides similar results.

In order to not complicate our design and eliminate the need to introduce some other "non-step" event, we decided to give the possibility to poll the current heading estimation from the rotation vector. This estimation will be smoothed by a parameter which specifies a timespan for calculating a current average. Therefore, it is up to the navigation module to get this additional information and using some kind of timeout mechanism in order to detect periods where the user is not moving. Whenever the user stops walking, it is up to the navigation module to give feedback on whether the user has to turn into a certain direction or not.

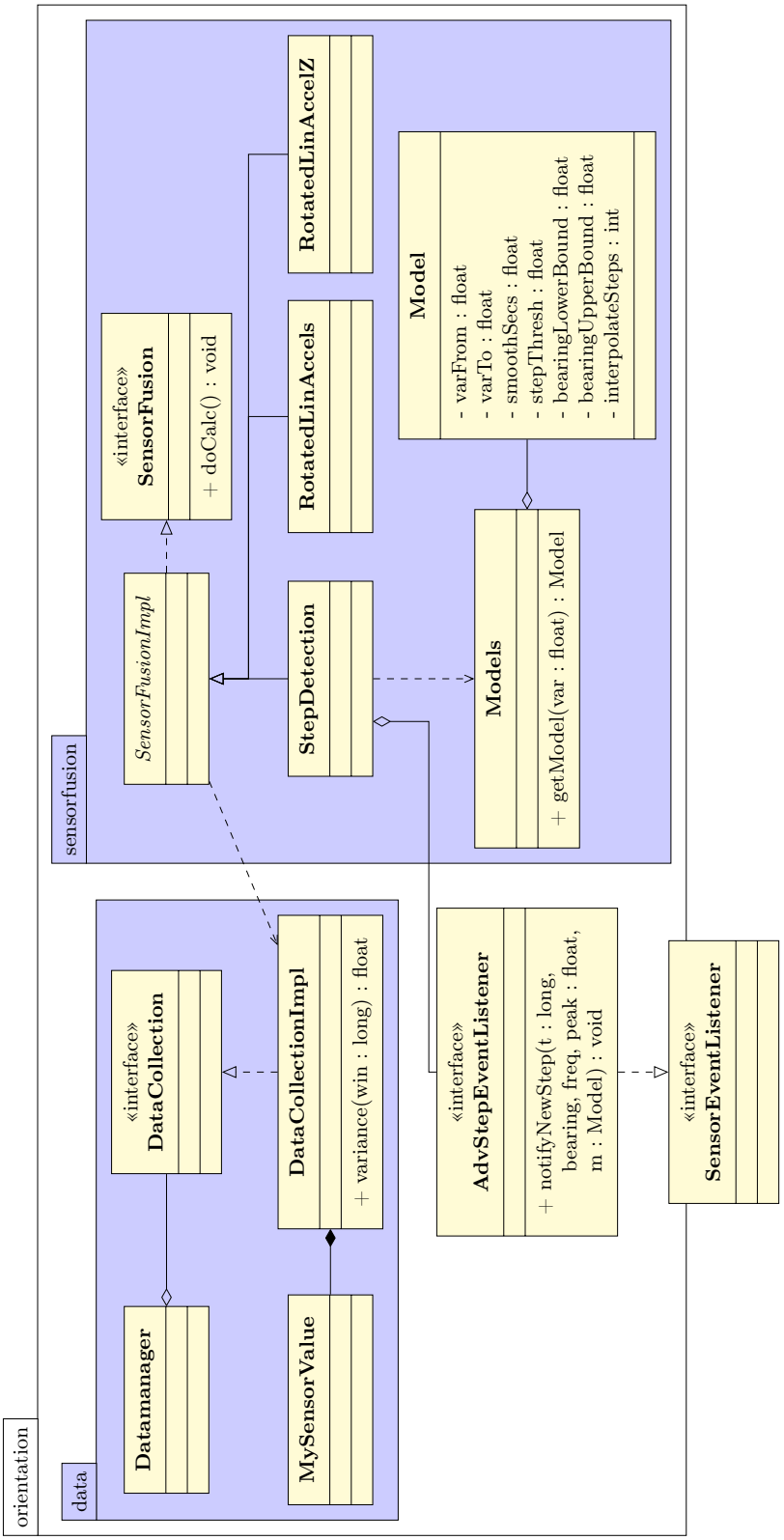

Figure 5.2 Simplified Class Diagram - Orientation Determination. It consists of two main packages - data and sensor fusion. The data package defines used datastructures for efficient access to stored sensor data. The sensor fusion package has a modular design to create new calculated data series out of preliminary calculated or basic ones. It is easily possible to integrate this library into other projects by implementing the step listener and registering at the step detection. To gather sensor information, we also use the step listener.

5.6.2 Navigation

The user navigation consists three parts. In general, we display the current user's position and map whenever we are navigating. Moreover, we have implemented the possibility to select another additional type of active feedback - which is textual, audio or vibration. Figure 5.3 shows a schematic overview.

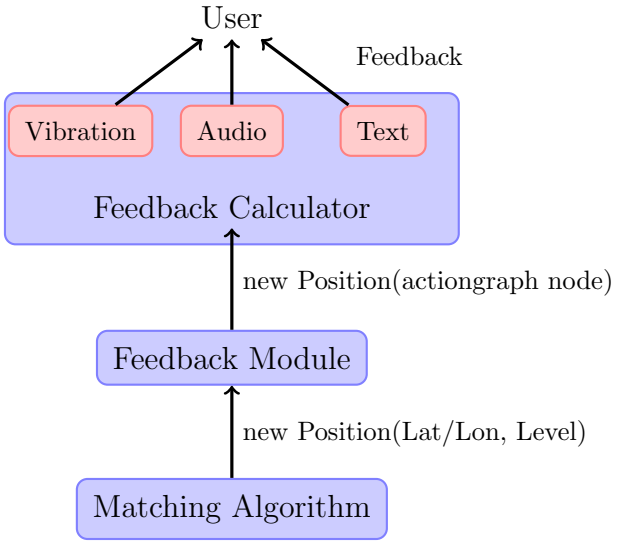

Figure 5.3 Navigation Feedback Overview. Each time the matching algorithm returns a new estimated position, the feedback module maps this position (given as geographic coordinates latitude/longitude and building-level) onto the corresponding actiongraph node. Afterwards we process this position in the selected feedback calculator which determines the current user's scenario and gives desired output.

In order to give additional feedback, a feedback module keeps track of the current route and is responsible for creating the actiongraph with containing Interest Points. Depending on the selected feedback type, we instantiate a specialized module which calculates and gives output.

Each time the matching algorithm returns a new estimated user position, we trigger the navigation module. We match the updated position onto the current route via searching inside the quadtree. If the nearest actiongraph node exceeds a predefined maximum distance, i.e. the location cannot be found, we calculate a new route from the nearest actual map walking-way node. Otherwise, we hand over the new position to the actual feedback-output module which calculates the current apparent scenario.

Here, we can specify when to give feedback (including timing thresholds to not give a command at every step). We use our MLTL approach on the resulting path from the current estimated user location towards the destination for scenario determination.

For easy integration of new feedback types, we have defined an interface `Feedback-Strategy` and provide an abstract implementation for convenience. Depending on the user's selection, we use a concrete implementation of the feedback strategy to give actual feedback - which currently may be either vibration or an aggregated text/audio variant.

5.7 Insidious Bugs and Unittests

During implementation, it is important to get good quality source code. We measure quality in terms of correct functionality, readability and performance. In order to fulfill these general requirements, the we have to implement our application with care.

Readability is the easiest measure to fulfill - common Integrated Development Environment (IDE)'s provide methods for automatic code cleanup by configurable coding conventions. Moreover, useful comments and hints are important to enable others to understand how a problem is solved. Refactoring tools enable fast renaming of e.g. methods and classes. In addition, object oriented design patterns should be used whenever appropriate as typical software engineers know them by heart. Furthermore, they often provide elegant out of the box solutions for typically encountered problems.

The most crucial quality measurement is the correct functionality and performance. As long as a program does not fulfill its functional requirements and shows logical bugs, there is no need to speed things up. However, we can avoid performance traps right at the beginning. An example to this is our `DataCollection` implementation. It is supposed to hold sensor data and enable random access by timestamps. Furthermore, we need sequential access on subsequent data items. We have implemented this requirement by storing data items in a List while also storing each data item with its corresponding timestamp in a TreeMap which allows efficient random access.

We primarily focused on functionality first. Performance issues became interesting not before we ran into trouble with simple implementation. Many bugs arise because of spending far too much time on speeding things up, although a simpler solution would be adequate.

5.7.1 Unittests

However, we ensure functional correctness by using unittests [3] excessively. Unit tests in general have the advantage, that they typically test only small parts of a program and if such tests are managed correctly, they provide direct feedback on the correctness of the application. In fact, it is recommended to create tests defining and ensuring application behavior before writing any line of code of the actual programm.

Besides unittest for each single method we use at our Java implementation which can usually be tested without any further contex, we also use a more complicated unittest environment to load stored sensor data (cache it) and to process it inside the FootPath environment. This allows us to rerun a test under the same conditions and with the same sensor data again and again. Furhtermore, debugging via step-by-step code execution becomes feasible.

5.7.2 Challenging Bugs

We have spent most time at verifying results. On the on hand, we have the MATLAB implementation where we can fastly verify the results e.g. by plotting data series. On the other hand, we have the Android implementation which has to be executed directly on an Android device. Although our JUnit framework allow us to reproduce different test scenarios, it is hard to check the processed sensor data after each important calculation step. Furthermore, due to different implementations, results from our Android implementation do not exactly match those from our MATLAB implementation. Thus, we exported calculated results from the Java implementation in order to verify the results via MATLAB visualization techniques.

The most prominent bugs have been found in our actual step detection and heading estimation module. Creating an online algorithm is challenging as we have to keep track of the exact timestamps which data has already been processed and which not. Moreover, smoothing with our FIR-implementation works, but in order to get results as expected when having the complete data available. We cannot just start filtering the first unprocessed data item with a new filtering instance, but have to advance preliminarily processed data items for creating the correct context.

6

Evaluation

In this chapter, we will discuss the evaluation results of our implemented methods. For evaluation purposes and model fitting, we collected training data of 15 different participants. The training data consists of 24 runs with the device in the trouser pocket and 25 runs with the jacket pocket location. Moreover, the test runs split into roughly two groups with each the right and left position. We performed all device-related tests on a Samsung Galaxy Nexus (GT-I9250) smartphone. It provides all mentioned typical MEMS sensors like accelerometer, gyroscope, magnetometer and barometer. We use the maximum sampling rate, which is roughly 100Hz.

6.1 Device Carrying Location Estimation

In order to use the correct model, we presented a method for distinction via the variance in linear rotated z-acceleration. Best results for separating between both locations yields a threshold of $variance^{z}_{linAccel} = 24.3$. Table D.1 shows a comparison of the detected device locations against actual locations. However, we obtained these results in an offline approach, i.e. using the complete dataset.

The results show, that our method determines the trousers pocket location well with an error rate of 4.2%, whereas there are false positives at a rate of 12.0%. Taking a look into these falsely categorized jacket pocket datasets shows, that the used feature completely does not fit at all with corresponding variances at values of 31.22, 37.10 and 30.03. Moreover, the wrongly estimated location for the trousers pocket dataset shows a variance of 18.57.

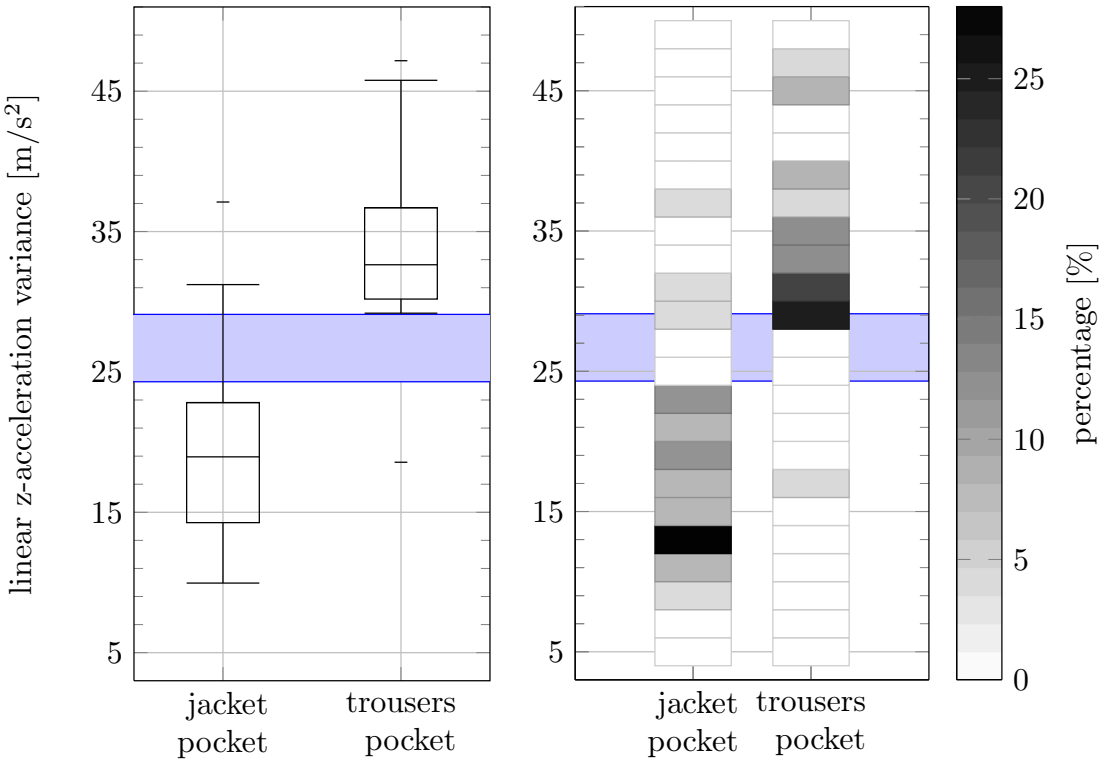

Figure 6.1 Model Distinction by z-Acceleration Variance.
Left: Box plot of the linear z-acceleration variance of the modeled locations. We can separate most of the datasets of the different locations by a single threshold. The blue marked area depicts the valid cutting range of the optimal threshold. Note, that the Whiskers denote minimum and maximum values here.
Right: A more detailed overview of the variance distribution shows the strong distinction for the trousers pocket location to variances below 29.

6.2 Evaluation on Collected Data

Sensor data output may be disturbed by the environment, i.e. the magnetic field sensor reacts on iron bodies, electric currents or permanent magnets. Furthermore, fused sensor data, that uses the magnetic field sensor, is affected as well. To avoid this problem, we decided to collect training data not in a building, but outside.

We have saved all sensor data of interest, i.e. accelerometer, magnetic field sensor and gyroscope, at each data-collection run. Furthermore we have also stored fused data like linear acceleration and the rotation vector. Besides these inertial sensors, we have also tracked Global Positioning System (GPS) coordinates and bearing information. We have restricted the participants to follow a predefined path for these runs in order to deduct a reliable ground truth out of static map data.

The evaluation will primarily rely on offline data calculated with our MATLAB implementation. Furthermore, we give an overview of similar results with our Java implementation of an online algorithm which is integrated into FootPath.

6.2.1 Heading Estimation Approchaes

In order to evaluate our new model, we use three other heading estimation approaches. In general, we use our presented peak step detection method (see Section 4.2.5.2).The heading estimation techniques try to evaluate an estimation for each detected step out of the present sensor data which are acceleration, linear accelerations and the magnetic field, whereas we rotate the accelerations into extrinsic coordinates by using the rotation vectors (see Section 2.3). We will present these methods briefly:

Direct Accelerations. The Direct Accelerations approach integrates over the complete current step time period the extrinsic x- and y-linear accelerations. We then transform the resulting vector into a heading towards north via the atan2 function, whereas we further have to mirror and rotate this angle (see Section 4.2.5.2).

Rotation Vector. We have seen, that the android system provides a fused virtual sensor called Rotation Vector (see Section 2.2.4). We use these values to transform all intrinsic coordinates to extrinsic coordinates via calculating a rotation matrix out of these values. However, we can directly extract a heading estimation towards north out of this matrix.

The Rotation Vector approach calculates an average over a complete step time period. For this reason, we calculate a heading estimation out of the rotation matrix for each data entry inside the current step time period. To avoid error prone average calculations directly on angles, we integrate over the resulting x- and y-components and recalculate a final heading estimation. Note, that there is no need for a further transformation as we already calculate on headings towards north.

New Jacket and New Trousers. The New Jacket approach defines our method, but restricts it to only use the jacket model parameters regardless of the classification of the dataset. The same hold for the New Trousers method, which restricts the usage to the trousers model.

New Estimation. The New Estimation method will use our complete new heading estimation approach. It will classify the current dataset by the feature already introduced in Section 4.2.5.1 and evaluated in Section 6.1. Depending on this classification, it will use the specific model for heading estimation, i.e. either the trousers or the jacket model.

6.2.2 Flipping

Due to the fact, that the usage of the Magnetic and Rotation Vector methods have the problem of possibly flipped reults (see Section 4.2.4, Section 4.2.2 and Chapter B), we have evaluated on different possibilities for flipping. This means, we have calculated heading estimations on the complete data set with both approaches. Under the assumption, that we know the actual heading, we flip the result if it has an absolute difference greater than 90°.

We give a comparison between different flipping references at both methods in Figure D.1. The used references are Direct Accelerations, our New Estimation and the known groundtruth of the data. For completeness, we also show results withhout flipping.

However, by careful investigation of each dataset, we observe that the flipping will only occur in rare cases in our collected sensor data. Generally, Direct Accelerations do not yield better results than using no flipping. The usage of our New Estimation method yields slightly better results than no flipping. We observe similar results when using the groundtruth as a reference (which is only available in our test setup) with the exception, that the resulting error is strictly bounded to a maximum of 90°. In fact, flipping by New Estimation or ground truth will in most cases probably result in an error-correction behavior.

The upper quantile of the non-flipped Rotation Vector is roughly 90° which in turn leads to 25% flipped estimations. Although this huge amount of values gets flipped, the overall error distribution changes its median only from $q_{.50} \approx 65°$ to $q_{.50} \approx 60°$. The same pattern holds for the Magnetic approach.

Nevertheless, we decided to use flipping by the groundtruth which makes both methods more competitive and thus, we denote instances using the ground truth flipping as Rotation Vector* and Magnetic*.

Further statistical data and a distinction between trousers and jacket data sets is provides in Section D.2.

6.2.3 Smoothing

In order to further improve results, we have tested a different data smoothing approaches. On the one hand, we tested data pre-smoothing which uses the step detection on the unsmoothed data, but the actual heading estimation then uses smoothed data. On the other hand, we tested post-smoothing of the bearing estimation results.

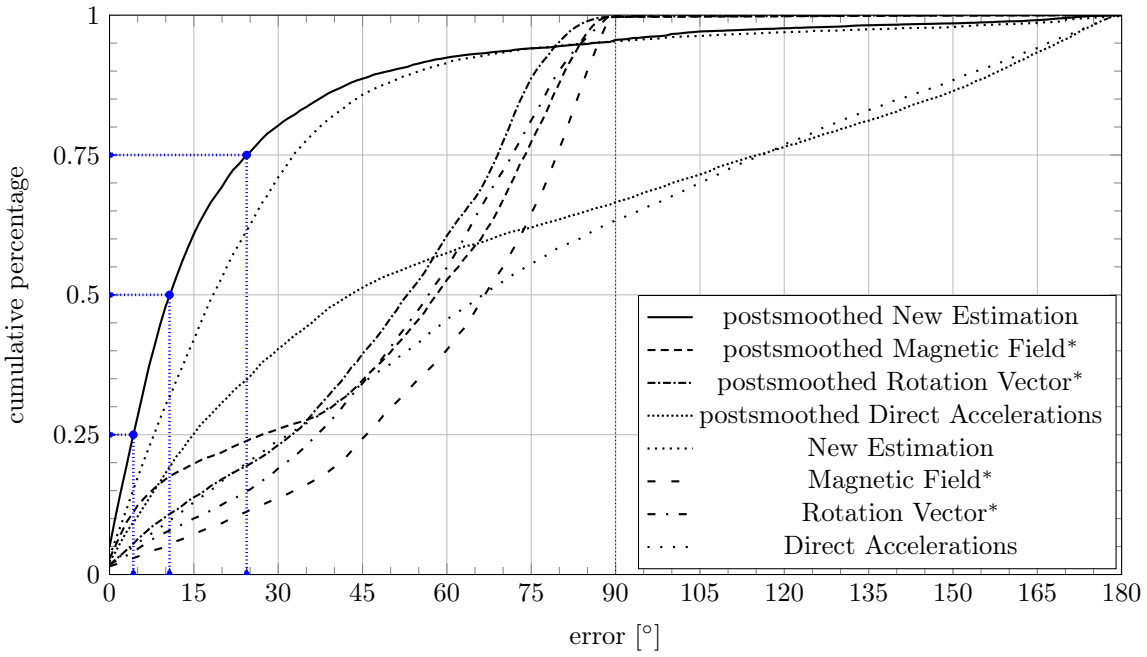

Figure 6.2 Model Evaluation: Post-Smoothing. This figure depicts the error distributions of the regarded heading estimation approaches with and without post-smoothing.
In general, the simple post-smoothing improves results regardless of the used heading estimation technique.

6.2.3.1 Pre-smoothing

In general, pre-smoothing will not yield better results. Pre-smoothing is discussed in Section D.3.1.

6.2.3.2 Post-smoothing

The tested postsmoothing simply takes the average of each new estimated value with the last estimation. We have tested this smoothing technique on our datasets and present the results in Figure 6.2.

The post-smoothing results in better performance for all heading estimation techniques. Our New Estimation approch shifts its median and upper quantile of its absolute error distribution down by over 10°. The Rotation Vector method also improves its absolute error distribution by a rough average of 5°. Moreover, we observe even better improvements at the Magnetic method, where the absolute error distribution shifts down by up to 30°. The most remarkable difference shows the Direction Accelerations which changes from a uniform-like error-distribution to a median of $\approx 45°$.

As the smoothing improves all results, we decided to use the post-smoothing for all subsequent tests and evaluations.

6.2.4 Model Parameter Determination

We calculate the model parameters (see Section 4.2.5.2), i.e. the interval $I = [i_0, i_1]$ to use for heading estimation, using optimization. For this reason, we calculate the total error for possible intervals at a certain discretization. We have chosen the discretization to a steplength of 0.025 which introduces $\sum_{i=0}^{40} i = \binom{41}{2}/2$ possible intervals, whereas we restrict these to $i_0 < i_1$ leading to 820 combinations. Assuming a normal step taking about 1s, we will have 100 samples with our used smartphone (100Hz). The chosen optimization resolution then enables a distinction of 2.5 samples which we regard as fine enough.

The actual optimization algorithm calculates the heading estimation on a given traning data set with our new generalized model using each time the specialized interval combination. While optimization, besides the total heading estimation error, we also track the error variance among all tested datasets for this interval which is a measure for the stability of this interval on the complete training data.

A final model decision may be a mixture of both measurements - the total error and the variance. However, we have decided on the parameter values $I_{\text{trousers}} = [0.825, 0.975]$ and $I_{\text{jacket}} = [0.525, 0.95]$, whereas they each did not provide the lowest total error, but introduced an slightly higher error at a much lower variance measure. In fact, the chosen trousers model instance has an average heading estimation error about 5° over the instances with the least error value. Moreover, the jacket model instance shows an average heading estimation error less than 2° over the least error value.

We have visualized our optimization results for the trousers datasets in Figure 6.4 and for the jacket datasets in Figure 6.3. The trousers model shows different areas of possible "good" model paramter combinations. The upper left region at $I = [0.25, 0.95]$ actually performs good in terms of average error and variance, but is slightly worse at both measurements. The second big region of possible intervals may be a result of secondary steps which also have influence of the primary leg (i.e. the one, where the device is located).

However, we also present optimization results over the complete dataset leading to a generic model for both locations in Figure D.12. Furthermore, we have also investigated on the heading estimation with best combined model parameters and our normal step detection method. The results are presented in Figure D.15.

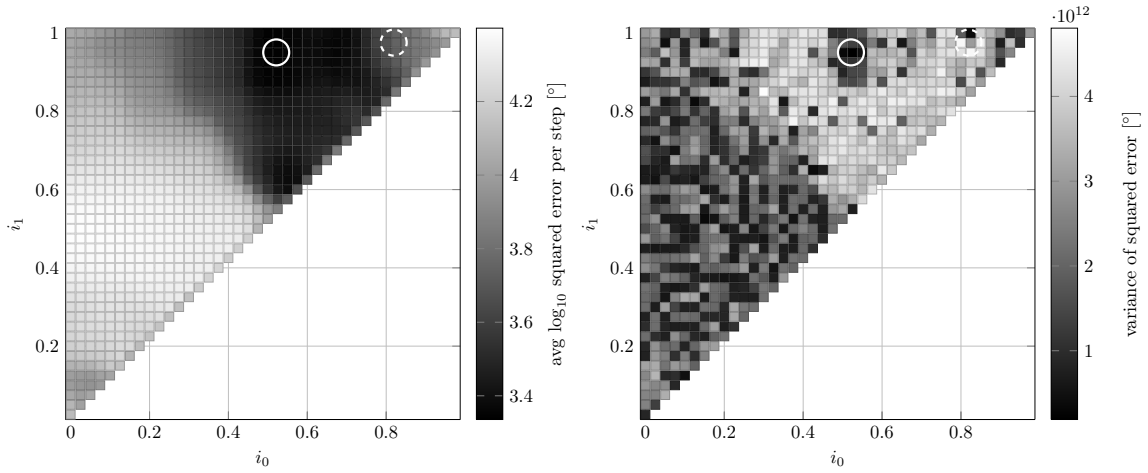

Figure 6.3 Model Parameter Determination: Jacket. This figure shows the results of model optimization for the jacket testruns. Both graphics show all valid model-intervals $[i_0, i_1[$ which determine the range of values for the heading estimation per step. Generally, darker quads relate to better results, whereas lighter quads relate to worse results.
Left: Each quad encodes the average $\log_{10}$ squared error per step in degree into a gray-value. We can see an area of "good" model parameters around $i_0 = [0.5, 0.7]$ and $i_1 = [0.6, 1]$.
Right: Each quad depicts the average variance of the sum of squared errors over all testruns. The results vary highly between subsequent possible discretized interval values. However, we have chosen our final model parameter by taking both values into account. We have marked the parameter combination with the white circle at $i_0 = 0.525$ and $i_1 = 0.95$.
Note, that we have marked the optimal parameters with the white solid circle. The dashed circle marks our parameters for the trousers model.

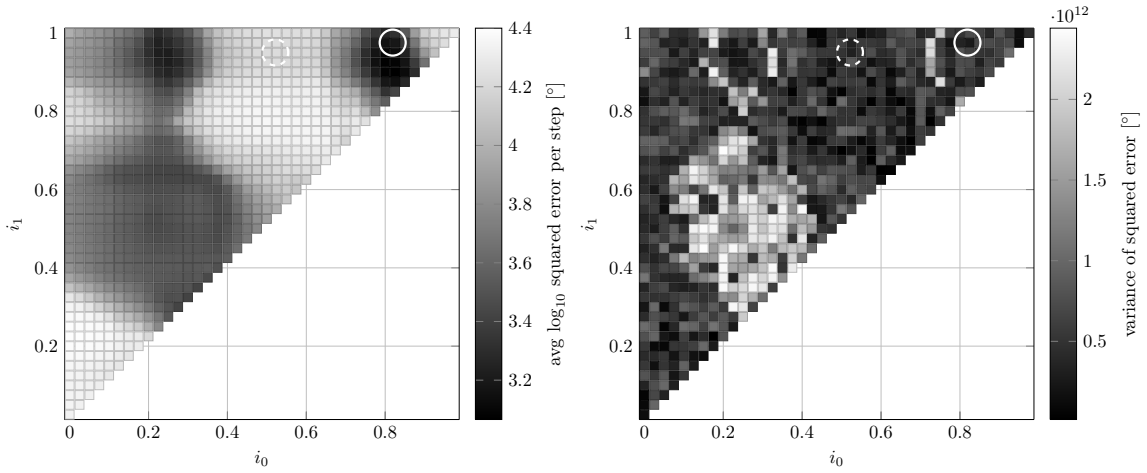

Figure 6.4 Model Parameter Determination: Trousers. This figure shows the results of model optimization for the trousers testruns. Both graphics show all valid model-intervals $[i_0, i_1[$ which determine the range of values for the heading estimation per step. Generally, darker quads relate to better results, whereas lighter quads relate to worse results.
Left: Each quad encodes the average $\log_{10}$ squared error per step in degree into a gray-value. We can see two areas of "good" model parameters around $i_0^1 = 0.25$ and $i_1^1 = 0.95$ and $i_0^2 = 0.8$ and $i_1^2 = 0.9$.
Right: Each quad depicts the average variance of the sum of squared errors over all testruns. The results show higher variance at an area of $i_0 = [0.15, 0.4[$ and $i_1 = [0.35, 0.65[$. However, we have chosen our final model parameter by taking both values into account. We have marked the parameter combination with the white circle at $i_0 = 0.825$ and $i_1 = 0.975$.
Note, that we have marked the optimal parameters with the white solid circle. The dashed circle marks our parameters for the jacket model.

6.2.5 Offline Approach

For the main evaluation we used our offline MATLAB implementation. We discuss a comparison between the online and offline in Section 6.2.6. For evaluation we calculated for a certain datasat (only trousers, only jacket or both) the steps and subsequently the heading estimations for all different approaches. As we collected the outside, we also have GPS data available and provide this as a reference.

6.2.5.1 Jacket Dataset

We show a general overview on the results for the jacket datasets as a boxplot in Figure 6.5. Our evaluation on the jacket dataset shows, that both, the New Estimation as well as the restricted New Jacket method perform well and provide comparable results to the GPS reference. Note, that the outliers at the Global Positioning System (GPS) dataset are a result of lag after a turn while collection the data.

The restricted New Trousers model and Direct Accelerations show a similar error distribution which is confirmed by the error distribution shown in Figure D.13. The error-range spans at complete 180°, whereas the lower quantile and median are close at $q_{.25} \approx 15°$ and $q_{.50} \approx 40°$. While estimating half of the data at an error below these angles, the next quarter settles up to $q_{.75} \lesssim 100°$. We can observe the high specialization of the trousers model here as it does not fit on the jacket datasets well.

The Magnetic and Rotation Vector methods also yield similar results. Their complete inter-quartile distance ranges from $\approx 35°$ up to 70° where it focuses at $q_{.50} \approx 65°$. Although both datasets are generally restricted by our optimal flipping towards the groundtruth, we can observe errors greater than 90°. The reason for outliers beyond this angle is our post-smoothing approach. The raw headings are restricted, but due to special circumstances, solely averages will get a higher value. We show a more detailed overview on this fact in Figure D.13 which depicts the error distribution.

6.2.5.2 Trousers Dataset

We also give an overview on evaluation results for the trousers datasets as a boxplot shown in Figure 6.6. Again, we can see some outliers at the GPS data. Moreover, our New Estimation method as well as the New Trousers model fits well on the trousers datasets are provide comparable results to GPS. Their upper quantile shows errors below $q_{.75} \approx 30°$.

The Direct Acclerations are uniformly distributed over the complate range of 180°. Moreover, the Rotation Vector as well shows a uniform distribution over its possible error range up to 90° due to flipping. This is confirmed by the error distribution shown in Figure D.14.

Even worse are the results of our New Jacket model on the trousers datasets. The lower quantile is at $q_{.25} \approx 95°$, whereas the upper quantile shows errors below $q_{.75} \approx$

165°. As well as seen before at the jacket datasets, we can observe the specialization of the models towards their target datasets.

The Magnetic approach does not provide a reliable heading estimation either. Although the lower quantile is at $q_{.25} \approx 20°$, the focus of the complete error distribution is at $\approx 80°$.

In fact, both results, of New Jacket and Magnetic method, would both be more reasonable if flipped by 180°. However, we currently have no explanation for this behavior. Note, that again, the Rotation Vector as well as the Magnetic approach are not completely bound to 90° here due to post-smoothing.

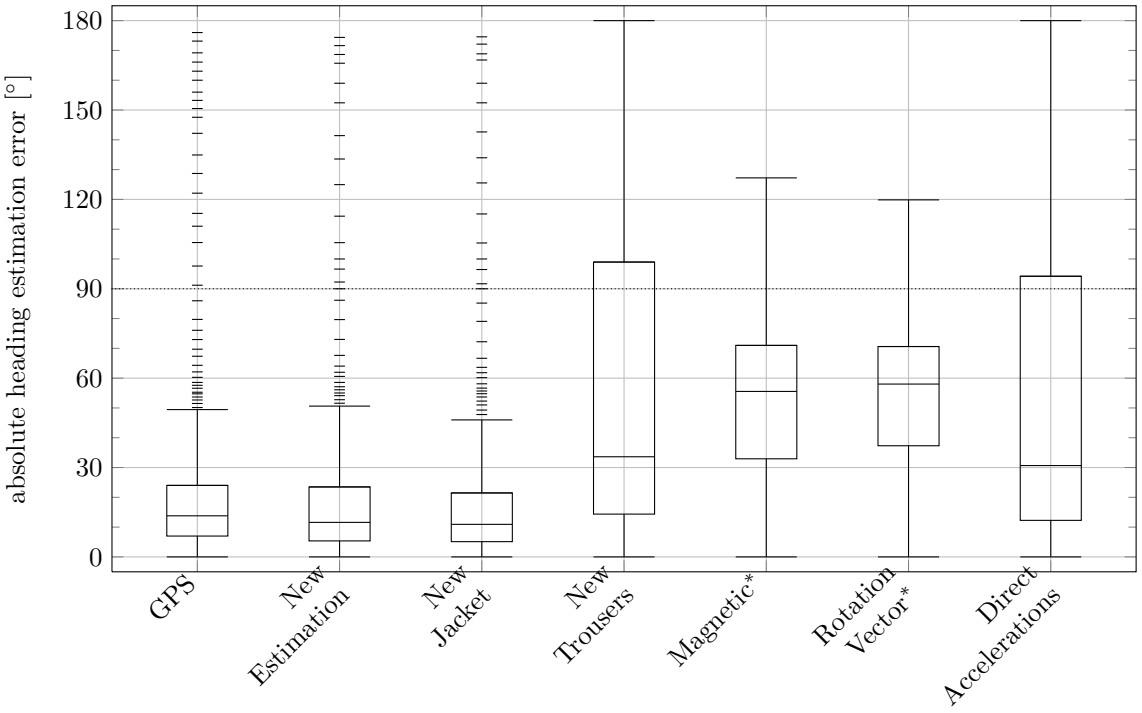

Figure 6.5 Model Evaluation: Jacket. This figure shows a boxplot of all different regarded techniques for heading estimation over all jacket testruns. We also include Global Positioning System (GPS) as a reference.
As the New Estimation and New Jacket method provide good results. We see, that our jacket model is competitive to GPS. The Magnetic and Rotation Vector approach are similar. New Trousers model and the similar Direct Accelerations technique are both outperformed by all other methods.

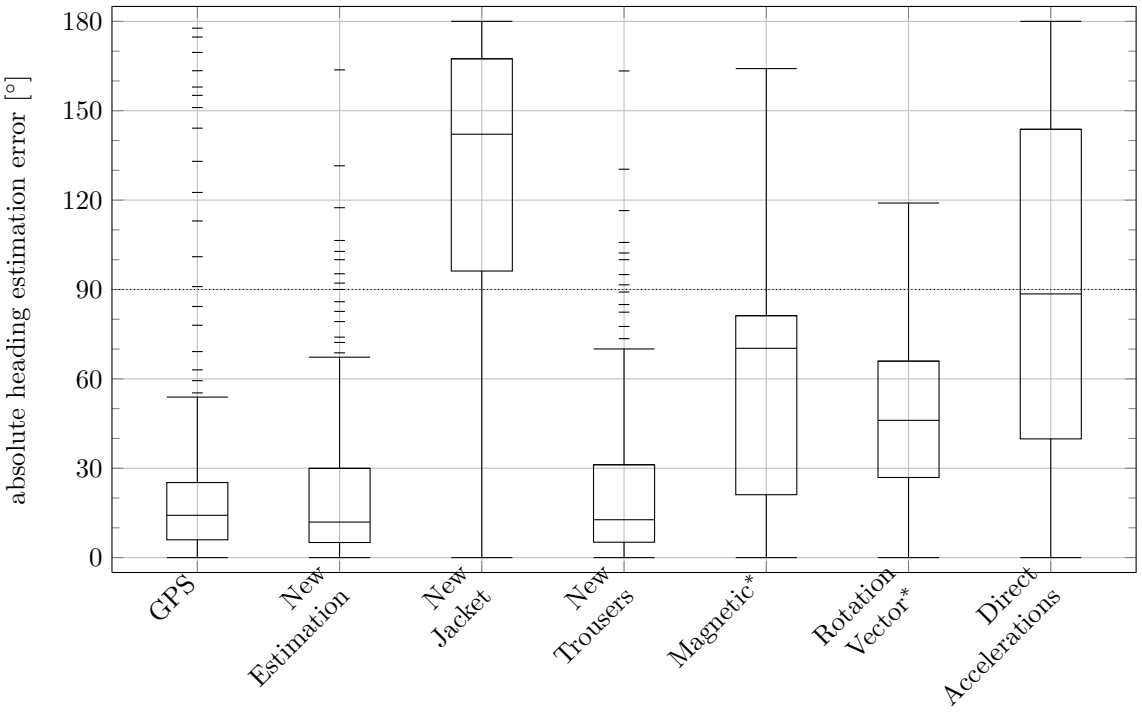

Figure 6.6 Model Evaluation: Trousers. This figure shows a boxplot of all different regarded techniques for heading estimation over all trousers testruns. We also include Global Positioning System (GPS) as a reference.
The New Estimation and New Trousers method provide good results. We see, that our trousers model is competitive to GPS. The Rotation Vector approach is slightly better than the Magnetic technique. The Direct Accelerations show a uniform error distribution, whereas the jacket model shows a bias into the opposite direction and thus, does not fit at all.

6.2.5.3 Complete Dataset

As the results on the complete dataset are the most important ones, we provide two different representations of the absolute error distribution. These are a boxplot in Figure 6.7 and the detailed error distribution in Figure 6.8 which also include GPS as a reference.

The GPS and our New Estimation approach prove to give accurate and similar results by looking into the distinct evaluations for our trousers and jacket model.

On average, the restricted New Jacket and New Trousers model give accurate results as well with a median at $q_{.50} \lesssim 25°$, but this representation is deceptive - at least for the New Jacket method. We have already seen in the sections before (see Section 6.2.5.2 and Section 6.2.5.1), that the models do not correspond to datasets from the type.

The Direct Acceleration technique does not provide an accurate heading estimation in general. Although the median is at $q_{.50} \approx 45°$, the error distribution curve flattens until reaching the upper quantile at $q_{.75} \lesssim 120°$, i.e. about one third of the estimations show an error at over 90°.

Overall, the Rotation Vector and Magnetic heading estimation perform similar. Nevertheless, the Magnetic approach is much better in the lower quantile, whereas the Rotation Vector is slightly better in the other regions. By taking a look into the error distribution, we observe that both approaches show a linear behavior in their error distribution after the lower quantile at $q_{.25} \approx 30°$ until reaching the 90° boundary caused by flipping.

Note again, that the Rotation Vector as well as the Magnetic approach are not completely bound to 90° here due to post-smoothing. We can confirm the seldom occurrences of higher errors by the presented error distribution.

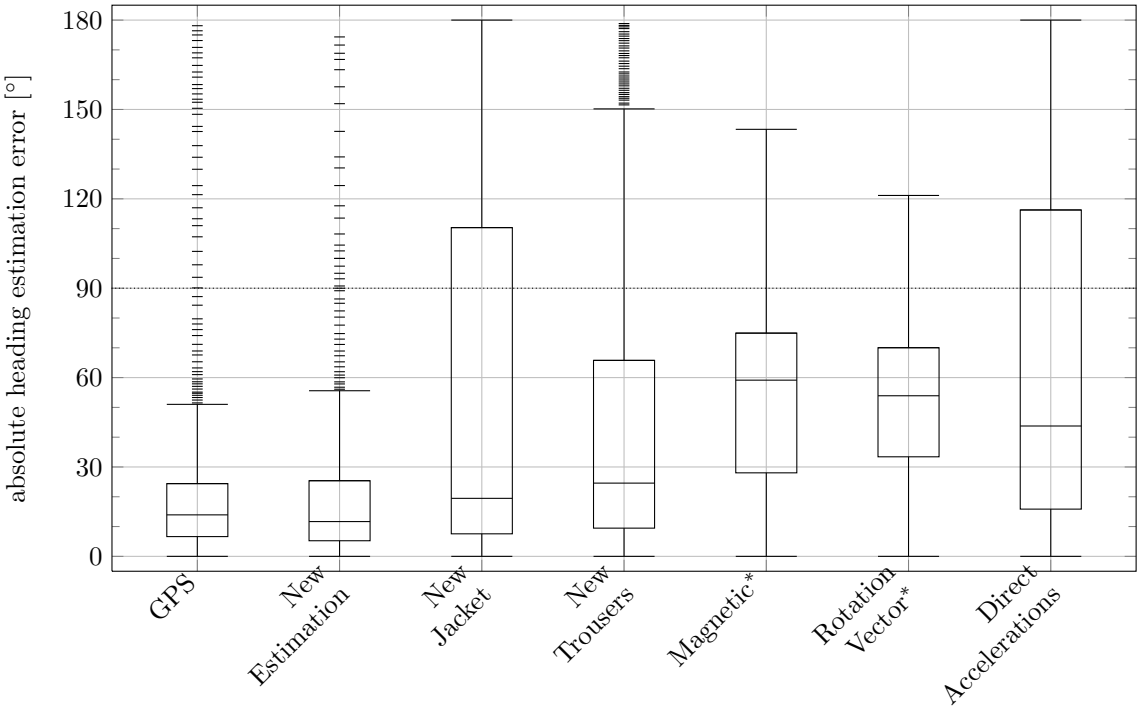

Figure 6.7 Model Evaluation: Overall. This figure shows a boxplot of all different regarded techniques for heading estimation over all testruns. We also include Global Positioning System (GPS) as a reference.
Our New Estimation approach gives overall similar results to GPS. Both, the New Trousers and New Jacket model show a low median error due to the good results on their specific target testruns, but cannot considered to give accurate results in general. Again, the Magnetic and Rotation Vector approaches are similar, but not as good as our new methods. Direct Accelerations have a median error below 50°, but its error distribution spans at a wide range.

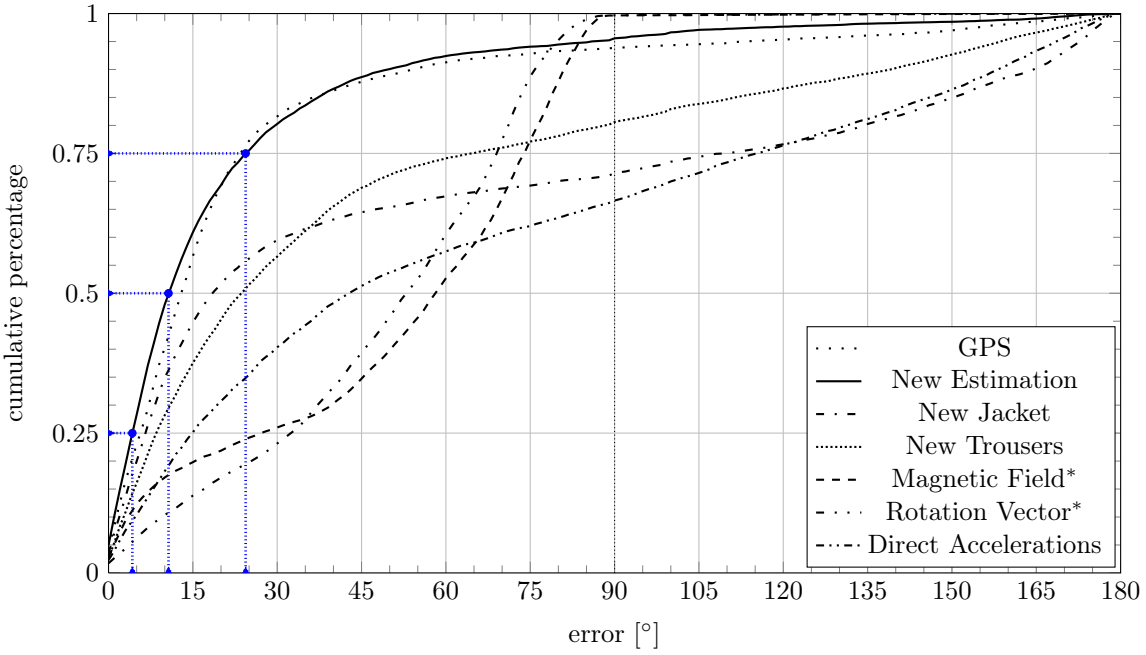

Figure 6.8 Model Evaluation: Cumulative Error Distribution - Overall. This figure shows the complete error distribution of the different regarded heading estimation techniques on the whole testing data including Global Positioning System (GPS) as a reference. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error. Our New Estimation technique is almost similar to GPS. The trousers and jacket model provide not as good overall results as they are designed for their specific device locations. Rotation Vector and Magnetic are bound to and error below 90° degree due to optimal flipping. They both outperform the Direct Accelerations technique in terms of the $q_{.75}$ quantile and their restriction to 90°.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) for our New Estimation approach.

6.2.6 Online Approach

On order to integrate our new heading estimation technique into FootPath (see Section 2.5 and Section 4.1), we have implemented an online variant in java for android. The evaluation uses the created JUnit framework (see Section 5.7.1) and testcases to process the collected datasets. We have stored the processed results for subsequent analysis and statistical evaluation.

The design of the testing framework allows us to give the complete data into the system and let it act like an offline variant as well. Thus, we have run two different instances of the framework on the whole data collection and separated the results accordingly into online and offline. Furthermore, we track the actual type of the data and keep this information to distinguish between the trousers and jacket datasets.

We present the results in a boxplot in Figure 6.9. In general, there are only small differences between the online and offline variant for both datasets, i.e. trousers and jacket. However, the online variant typically performs slightly worse. We also provide a more detailed overview on the error distribution in Figure D.22.

Due to the limited local knowledge at the online approach, there arise differences to the offline variant. We have seen, that our model distinction bases on the linear z-acceleration variance which results in some cases to locally different values than for the complete dataset. Thus, the online algorithm switches the used model parameters ad hoc and calculates sometimes values for the other model.

Moreover, the step detection depends on the model determination as well. We have also investigated on the step detection in the online approach in comparison to the offline approach. Figure 6.10 shows quantitative differences between both variants. Due to different local variances in some cases, the step detection will detect a smaller or greater amount of steps due to ad hoc model switching.

Trousers. If a trousers dataset shows locally smaller variances than typically usual, it may be determined as a jacket instance while heading estimation and thus, the step detection threshold will be lower. This leads to a higher amount of detected steps. We can observe this behavior in the given figure as the online approach detects 8% more steps per dataset on average.

Jacket. The other way round, if a jacket dataset has sometimes a greater local variance, it these segments may be determined as a trousers instance. This defines a higher step detection threshold and thus, leads to a smaller amount of detected steps. This behavior is also seen at the figure, where the online variant detects 4.1% less steps than the offline variant.

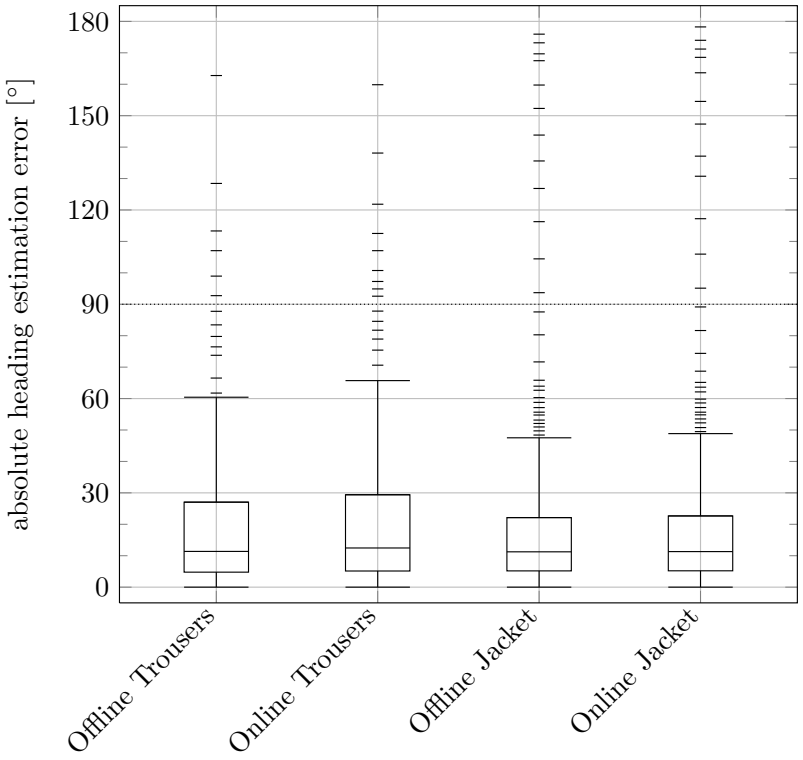

Figure 6.9 Model Evaluation: Online vs. Offline. This boxplot shows a comparison between our offline and online version of the presented new approach. The online versions perform slightly worse than their corresponding offline algorithm. This is caused mainly due to the local knowledge restriction for model selection, i.e. distinction between trousers and jacket model.

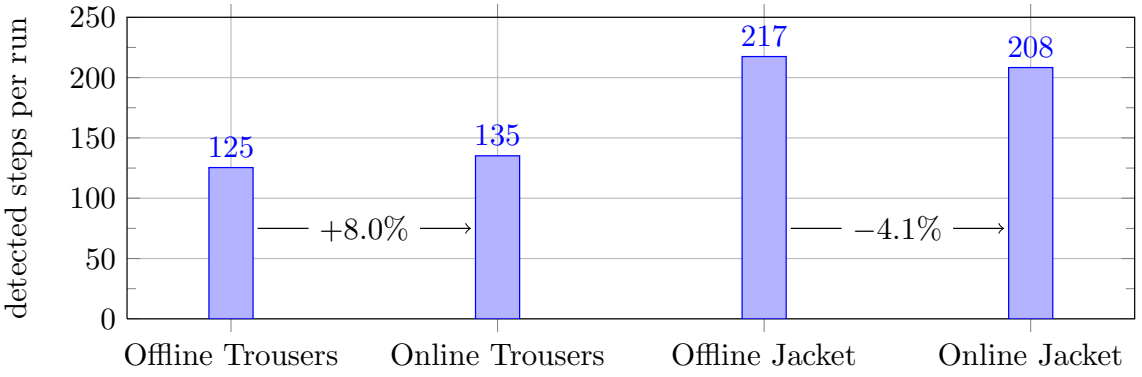

Figure 6.10 Step Detection: Online vs. Offline. This chart shows a comparison of detected steps per testrun (separated into trousers and jacket) between our online and offline version of the step detection.
As our step detection also depends on correct model distinction, we can observe small differences. The main reason is the local knowledge restriction for model selection, i.e. distinction between trousers and jacket model, which leads to false detections. However, trousers testruns tend to be classified as jacket instances at some timepoints. This leads to a smaller step-detection timeout and results in a higher amount of detected steps (+8.0%). The same holds for the jacket testruns into the other direction (−4.1%).

6.2.7 Outliers

We have evaluated all datasets individually to investigate the different heading estimation techniques on each of them. We will focus on different types of outliers in this section. These are false classified datasets and general outliers, where our heading estimation method does not fit well.

6.2.7.1 False Device Location Estimation

As seen in Section 6.1, that we have four false detections for the device carrying location. Three of these false detections are actually jacket-datasets, but are classified as the trousers location. Moreover, our model distinction algorithm classifies one actual trousers dataset as a jacket instance. We present evaluation results on these specific datasets as a boxplot in Figure D.16.

In general, we observe, that all results are good. Although we use the other model compared to the actual type, our algorithm provides a reliable heading estimation with some outliers for these specific datasets.

However, in general, the step detection misses every second peak due to the step timeout at wrongly classified jacket datasets. The same holds for the trousers datasets, but in the other direction. At these wrongly classified trousers datasets, the step detection recognizes every primary and secondary step which would usually not be the case. We show a comparison of used model bearing estimation intervals mapped to actual step intervals in Figure D.17.

Applying the jacket model on a trousers instance. Using the jacket model on a trousers dataset results in more steps than we would usually detect as the step timeout is smaller. Under the assumption, that we detect booth, each primary and secondary step, the step amount will roughly be at factor 2 of the amount which would have been detected using the trousers model. This means, that we calculate two heading estimations for a single actual trousers-model-step period with the jacket model interval $I_t = [0.525, 0.95]$. Mapping this interval on the actual step period results in $I_{\text{secondary}} = [0.5i_0^t, 0.5i_1^t]$ and $I_{\text{primary}} = [0.5 + 0.5i_0^t, 0.5 + 0.5i_1^t]$. These result at our configuration in $I_{\text{secondary}} = [0.2625, 0.475]$ and $I_{\text{primary}} = [0.7625, 0.975]$.

Taking a look into the trousers model parameter optimization (see Figure 6.3) shows, that the used intervals I_{primary} and $I_{\text{secondary}}$ are by far not optimal, but also give accurate results, whereas the primary interval shows a smaller error variance than the secondary interval.

Applying the trousers model on a jacket instance. Due to the missing of each second step, we only calculate a heading estimation over a period of two steps using the trousers model interval $I_t = [0.825, 0.975]$. We can map this applied interval on the period of two steps which results in $I = [2i_0^t, 2i_1^t]$. Furthermore, the actual relative interval applied to the second step at our configuration is $I' = [0.65, 0, 95]$

Taking a look into the jacket model parameter optimization (see Figure 6.3) shows, that the used interval I' is not as robust as the chosen parameter, but the average error per step is also low. This is an explanation why the false detected datasets perform good anyway.

Note, that this also relates to our combined model presented in Section D.4.

6.2.7.2 Outliers in Heading Estimation

Evaluation on individual datasets with our New Estimation approach shows, that 7 datasets result in error rate with an upper quantile of $q_{.75} > 45°$. We show the resulting error distributions of these datasets in Figure 6.11. Furthermore, we provide a boxplot in Figure D.18.

We can see in the error distribution, that most of these outliers perform at an upper quantile below $\approx 70°$. We consider them as sufficiently reliable as their median shows an error roughly below 50° which will be most likely sufficient for determining a correct movement direction in combination with map data.

Two datasets (Jacket 3 and Trousers 3) give worse results. Their error distribution shows a wide-ranged saddle point at their median at about 30° to 75°.

Taking a closer look into the heading estimation results of the Jacket 3 dataset, we can observe good performance (absolute errors below 30° which corresponds to the median) for the first half of the testrun. But in the second half, after having turned around, the heading estimation completely fails, whereas the step detection works as expected (detecting every second step). We have no explanation for this behavior.

A detailed view on the heading estimation results of the Trousers 3 dataset reveals a good performance for the first half of the testrun as well (absolute errors below 30° which corresponds to the median). However, the estimation results on the second half are shifted by $\approx +100°$ which explains the increasing percentages of the error distribution between 100° and 120°. The step detection works as expected (detecting the primary and secondary step). We have no explanation for this behavior.

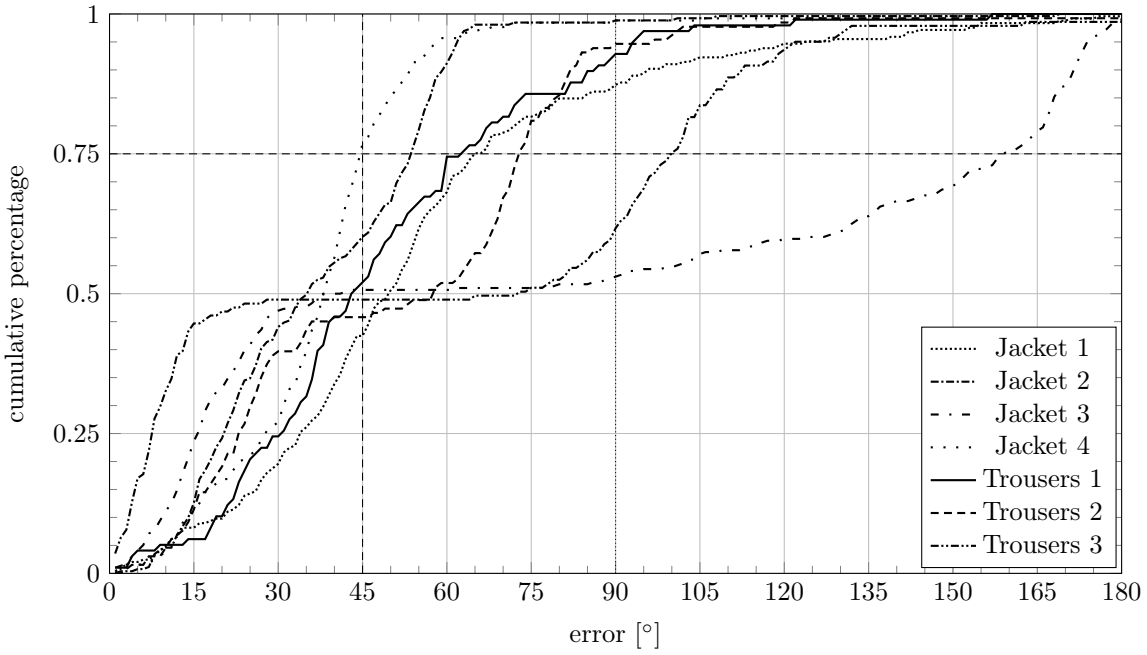

Figure 6.11 False Model Selection: Comparison of Outliers. This figure depicts the error distributions of out New Estimation approach at the estimation outliers having $q_{.75} > 45°$.
Most of the outliers show an upper quantile below 65° and we consider them as sufficiently reliable. However, the Jacket 3 and Trousers 3 datasets show a wide-ranged saddle point near their median at 30° to 75°. Moreover, the estimation on Trousers 3 dataset works a lot better than on the Jacket 3 dataset.

6.2.8 Example Path

In order to give a better understanding how the different methods relate and behave on a real-world example, we present two (three) different test runs - with the device inside the trousers pocket and inside the jacket pocket (we present a handheld example as well in Figure D.24). The accomplished path (shown in Figure 6.12) in has an overall distance of 200m and includes 4 doors as well as two level changes via a staircase.

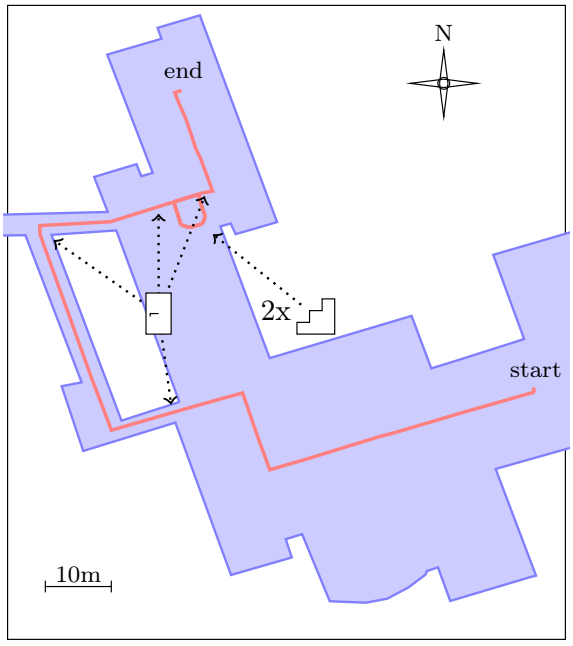

Figure 6.12 Example Run indoors - Map. This figure shows a map of the building where we accomplished our the two exemplary test runs.
The path includes 4 doors and two level changes via a staircase and has an overall distance of about 200m.

We can generally observe at these test runs, that our New Estimation method performs good which holds especially in comparison to the other methods. However, there are errors each time a significant bearing change takes place. These errors have probably their origin in the lag of our method. Assume we have detected a step and continue into the same direction. Whenever we change the direction right before doing the next step, the heading estimation will fail compared to the groundtruth as it calculates on the past sensor data since the last step. We illustrate this behavior in Figure 6.13.

However, in the jacket example, both, the Rotation Vector and Magnetic method show high errors rates with a median at $q_{.50} \approx 65°$, whereas the Magnetic approach is generally better. Taking a closer look into the estimated bearings in Figure D.23 reveals, that both approaches completely fail to estimate the correct heading from the beginning until 95s. Afterwards, we cannot find any clear error pattern as this period introduces a high amount if bearing changes due to the staircase over two levels.

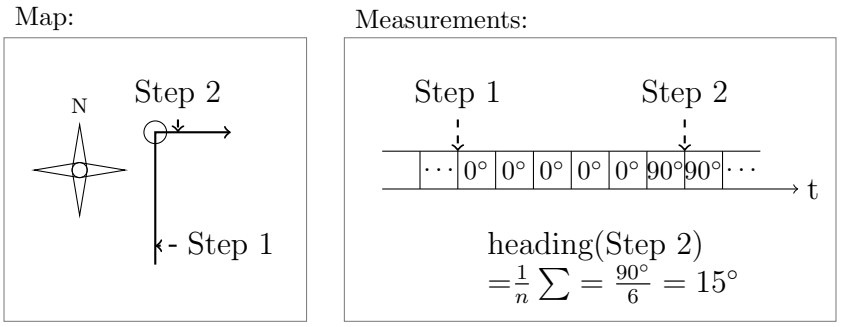

Figure 6.13 Heading Estimation Lag. This figure shows why heading changes introduce errors compared to the groundtruth.
Every heading estimation depends on the collected sensor data since the last step. This small example shows a constellation, where the user walks towards north and turns towards east right before doing the next step. For simplicity of the example, we will assume having direct heading data values available. Furthermore, we assume using the average over all heading values since the last step for a new heading estimation. Due to the fact, that the groundtruth at Step 2 is 90°, but new estimation calculation involves all data since the last step, this example will result in an absolute estimation error of 75°.
Note, that this error only happens on significant bearing changes will most probable not have any critical effect in practice, but will at least result in latency.

The trousers example shows some more differences between the Rotation Vector and Magnetic approach. Although the lower quantile of the Rotation Vector technique is lower than the corresponding quantile of the Magnetic method, the Magnetic method outperforms the Rotation Vector between their lower quantile and median at an error of 40°.

The Direct Accelerations perform good at both test runs which contradicts to the generic information retrieved from the overall performance evaluation in Section 6.2.5, which states, that this method does not work very good at the trousers location. The median is at $q_{.50}^{\text{trousers}} \lesssim 30°$ and $q_{.50}^{\text{jacket}} \approx 30°$, whereas the upper quantile is at $q_{.75}^{\text{trousers}} \approx 45°$ and $q_{.75}^{\text{jacket}} \lesssim 100°$.

We provide an even more detailed view on the statistics via boxplots in Figure D.25 and Figure D.29, as well as in the error distributions in Figure D.27 and Figure D.30.

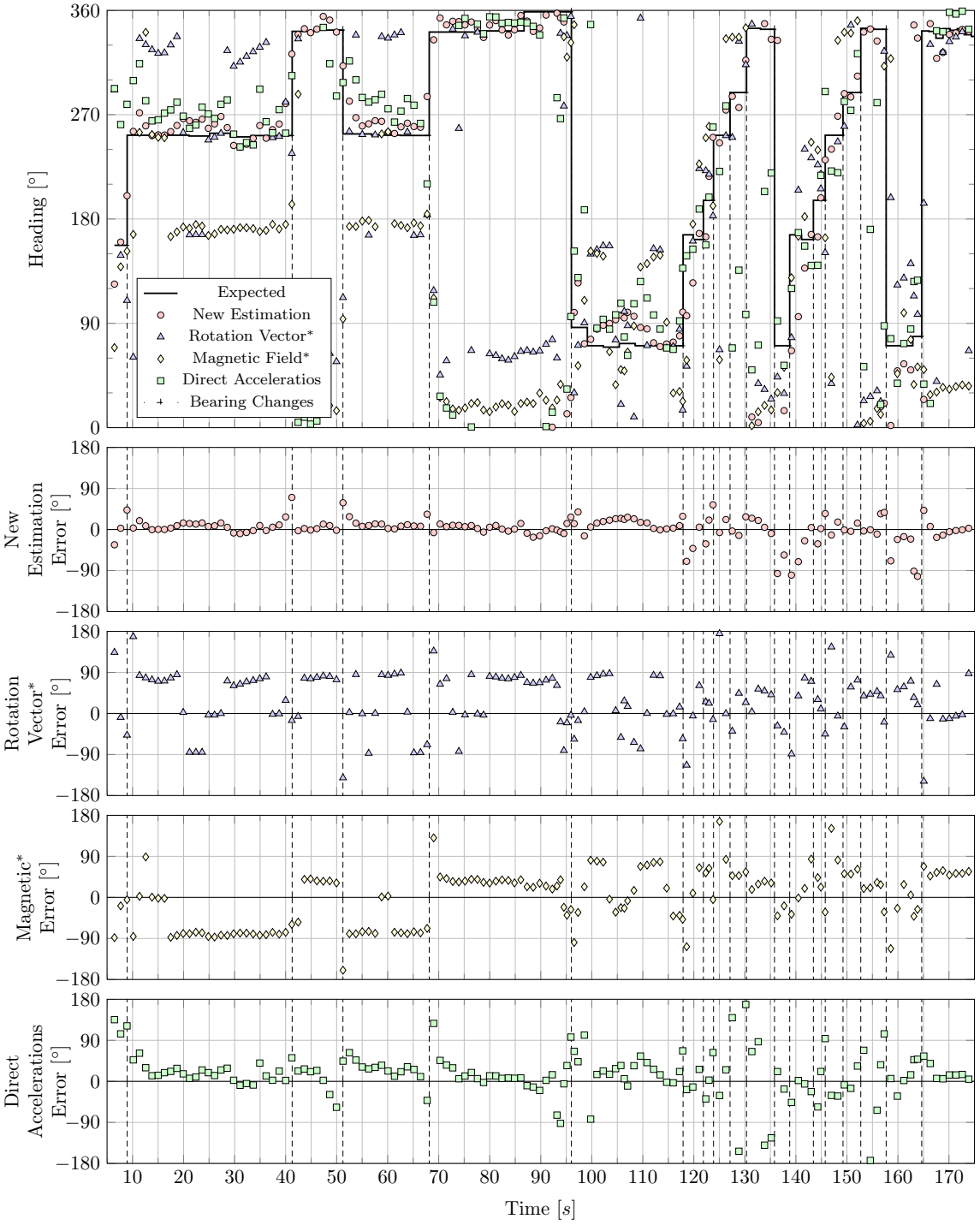

Figure 6.14 Example Run indoors - Results of Trousers Pocket. This figure shows the results of our example run of the path depicted in Figure 6.12 with the smartphone in the trousers pocket. The x-axis describes the passed time of the run. *Top:* The upper graph's y-axis describes the current heading in degree. We illustrate the expected heading values extracted from map data and estimated heading information of the regarded techniques. Furthermore, we marked significant heading changes by vertical dashed lines.
Center to Bottom: The other four subgraphs show an exact evaluation of the estimated headings, where the y-axis represents the error.
The New Estimated performs well at low error rates except for higher errors at each bearing change. Rotation Vector* and Magnetic* heading estimation results show significantly higher error rates, although we apply flipping. Direct Accelerations provide a much better heading estimation in this example, but is not as good as our New Estimation method.

6.2.9 Decreased Sampling Rate

We have seen, that our testing device returns sensor values at 100Hz which is the maximum the device can deliver. However, this is not the case with all devices and all android versions. In fact, there are no precisely defined sampling rate operating modes, but programming hints and the possibility to request a specific delay between sensor events. The Android Application Programming Interface (API) introduces 4 different sampling rates which are *fastest*, *game*, *ui* and *normal*. According to Milette et al. [37] these hints are hardcoded for Android 4.0.3 at sampling rates of 0Hz, 50Hz, 15Hz and 5Hz.

The general question is how our technique performs at lower sampling rates. Thus, we have evaluated our model against the testing data while gradually decreasing the resolution. In order to get a rough overview of our New Estimation at different sampling rates, we decided to use the fractions $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ..., \frac{1}{16}$ of the full sampling. This results in sensor data frequencies of $50.0\text{Hz}, 33.3\text{Hz}, \ldots, 6.3\text{Hz}$.

We accomplish sensor data downsampling by simply taking each n^{th} sample in order to get the fractional frequency of $\frac{1}{n}$ from the full resolution version.

6.2.9.1 Model Distinction at Lower Resolution

The first module in our heading estimation pipeline is the model determination. This means we have to deduct where the device is placed and which model to used from the available sensor data. These model parameters are the basis for step detection and heading estimation.

We have tested our model classification via linear z-acceleration on different sampling rates. We depict the results in Figure 6.15. The number of false detections remains stable at an absolute number of four over all different sampling rates. Moreover, the interval for an optimal separation between the trousers and jacket model remains also very stable and is overall valid in between a variance interval from 25.5 to 28 at all tested sampling rates.

Thus, we conclude, that our model distinction feature is very robust against different sampling rates and usable even at very low frequencies.

6.2.9.2 Step Detection at Lower Resolution

To make sure, that the whole system works at lower resolutions, we have also evaluated our step detection at different sampling rates. This part is very crucial as all subsequent processes and modules depend on a reliable step detection.

We illustrate the quantitative step detection results for different frequencies in Figure 6.16. In general, there are only small variations in the number of detected steps relative to the 100Hz datasets. The amount of detected steps remains very stable until there arise higher relative errors at 11.1Hz which already denotes a resolution decrease by factor 9. However, the lower frequencies introduce a maximum relative error of about 4%.

Thus, we conclude, that the step detection is very robust against different sampling rates as well.

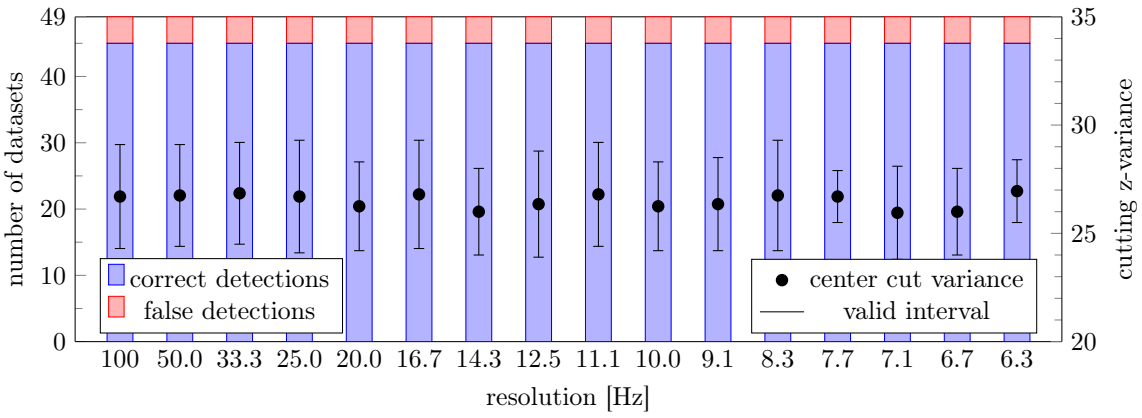

Figure 6.15 Model Distinction at different Sampling Rates. This figure shows model distinction at different sensor data sampling rates of our collected data. The colored bars denote the number of correctly and wrongly classified test runs, whereas we show the valid separating variance areas at the error bars.
Generally, the z-acceleration feature for model distinction is very robust against different sampling rates.

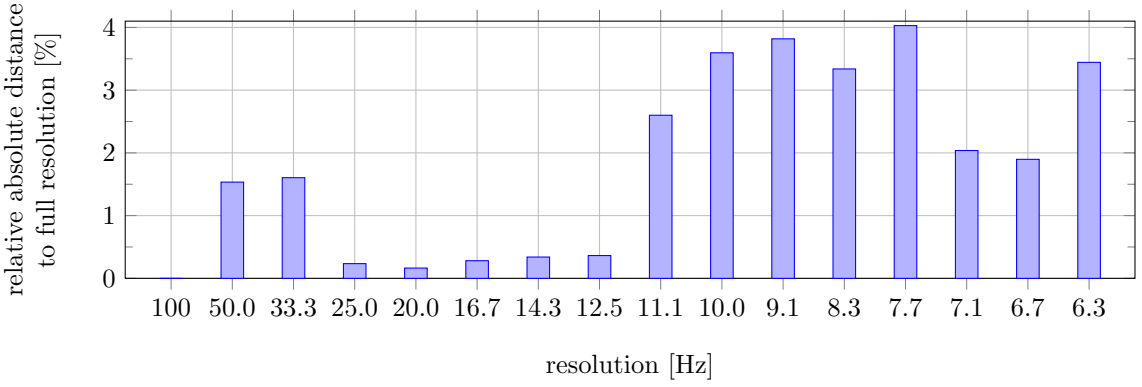

Figure 6.16 Step Detection at different Sampling Rates. This figure shows the relative average amount of detected steps over all testruns at different sensor data sampling rates. We normalized the values to the full sampling rate of 100Hz. The step detection is very robust against different sampling rates as the amount decreases at lower sampling rates, but remains below an absolute error of 4%.

6.2.9.3 Model Evaluation at Lower Resolution

We have now proven, that the model distinction and step detection work as desired at all different tested sampling rates on our test data. Furthermore, we have tested our previously determined model parameters on the datasets with lower sampling rate. We will continue discussing the results.

New Estimation. We depict our evaluation results for the New Estimation heading estimation technique at lower sampling rates in Figure 6.17 as an error distribution. It shows clearly, that the accuracy of heading estimation decreases with a decreasing sampling rate. Our New Estimation technique outperforms all other presented approaches in terms of the median at all sampling frequencies greater or equal to about 10Hz, even when the others use the full sampling rate of 100Hz.

Sampling rates below 12.5Hz seem to suffer from discretization issues as they show a significant rise of the error distribution at absolute errors of $90 \pm 45°$ and $90° \pm 55°$. However, we have not yet analyzed this behavior any further as these low sampling rates will usually be unlikely for a sensor-based application in the given context of using smartphones.

Magnetic. We illustrate the results for the Magnetic method in Figure D.19 as an error distribution as well. This method shows a contradicting behavior to the New Estimation technique. Lower sampling rates lead to better heading estimation results. When decreasing the resolution by factor 13, we improve the estimations by values of about 5° to 15°. However, we have no explanation for this behavior, maybe the resolution reduction acts like a big Low-Pass (LP)-filter here.

Direction Accelerations and **Rotation Vector.** We provide error distributions of the Rotation Vector in Figure D.20 and for the Direct Accelerations in Figure D.21. The main observation at the Rotation Vector result is, that does not change at all. The Direct Accelerations show a decreasing performance at decreasing sampling rates until reaching a uniform distribution below a sampling rate of 10Hz. The shape of the error distribution gradually approaches this uniform distribution.

6.2.9.4 Model Optimization at Lower Resolution

We have shown in Section 6.2.9.3, that the heading estimation performance of our New Estimated method decreases with decreasing sampling rates of the sensor data. However, the reason for this may be, that the used model parameters (obtained at 100Hz) do not fit on lower sampling rates.

In order to investigate this idea, we optimized the model parameters at the different sampling rates as well. Due to lower sampling rates, we also decrease the model parameter discretization resolution to 0.05 which is typically more than needed. We present the results for the trousers model in Figure 6.18 and for the jacket model in Figure 6.19.

Trousers Model. Looking at the full resolution of 100Hz, we can determine the regions of well fitting parameters. By decreasing the sampling rate, these regions get blurred, i.e. the overall error increases. by further decreasing the sampling rate, the regions around $i_1 \approx 0.9$ are eroded. The remaining best parameter combinations

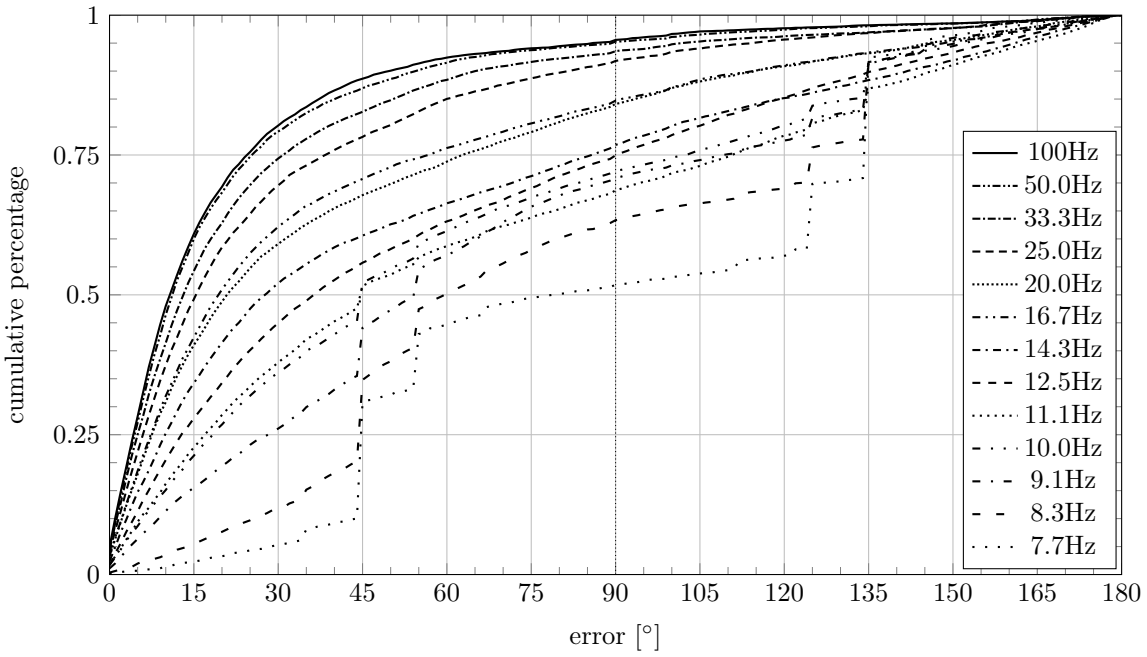

Figure 6.17 Model Evaluation at different Sampling Rates: Cumulative Error Distribution - New Estimation. This figure depicts the error distributions of our New Estimation approach at different sensor data sampling rates.
We can observe, that the heading estimation performs well with a $q_{.75}$ error below 90° at a decreased sampling rate by factor 8 (12.5Hz). Even lower sampling rates introduce higher errors. These instances also seem to suffer from discretization errors at $90° \pm 45°$ and $90° \pm 55°$.

shift into a region around $I = [0.1, 0.45]$ which shows, that our current model parameters, which have been fitting for the full sampling rate, do not fit for the lower sampling rates.

Jacket Model. Although we can observe a similar behavior of an increasing error while decreasing the sampling rate at the jacket optimization results and blurring, the optimal area does not shift as much as seen at the trousers model. Thus, the 100Hz jacket model remains more stable for lower frequencies than the 100Hz trousers model.

Discretization Issues. Moreover, we can see discretization artifacts at both optimization results. The main reason is, that the used optimization discretization resolution becomes higher than the actual sensor data resolution. To make this clear, consider a simple example with a jacket testrun. A typical step takes about $t = 600\text{ms}$. The optimization now tries to find two good interval parameters at a resolution of $r = 0.05$ chunking this period into $\frac{1}{r} = 20$ parts. These 20 timepoints can only be distinguished if there are at least 20 different sensor data values available. In other words, for proper model fitting on a given model discretization, the sampling rate of the sensor data is supposed to be at least $\frac{20}{600\text{ms}} = 33.3\text{Hz}$.

More general, let t denote the minimum time period for a step. Furthermore, let s be the desired used frequency. Then the discretization step size r is sufficiently small at $r = \frac{t}{s}$.

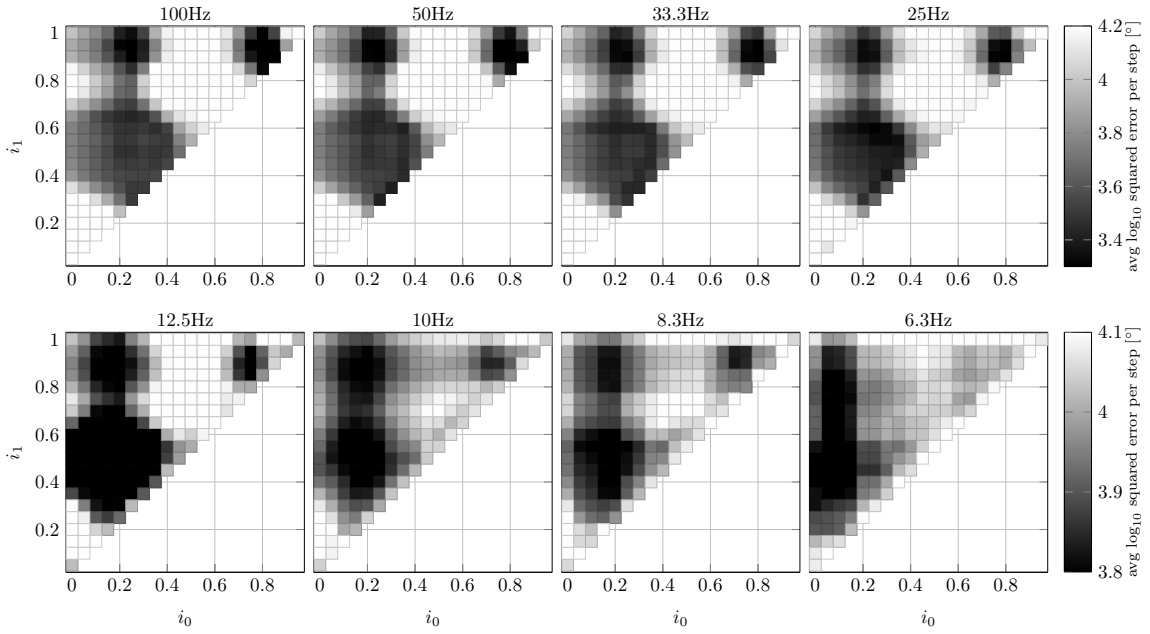

Figure 6.18 Model Parameter Determination at different Sampling Rates: Trousers. This figure shows the model parameter optimization results in terms of the average $\log_{10}$ squared error per step over all trousers testruns at different sensor data sampling rates. The sampling rates span a range between full 100Hz and 6.3Hz. We can see, that the optimal parameters change at lower sampling rates to a region at $i_0 = 0.1$ and $i_1 = 0.5$, whereas the average error increases.

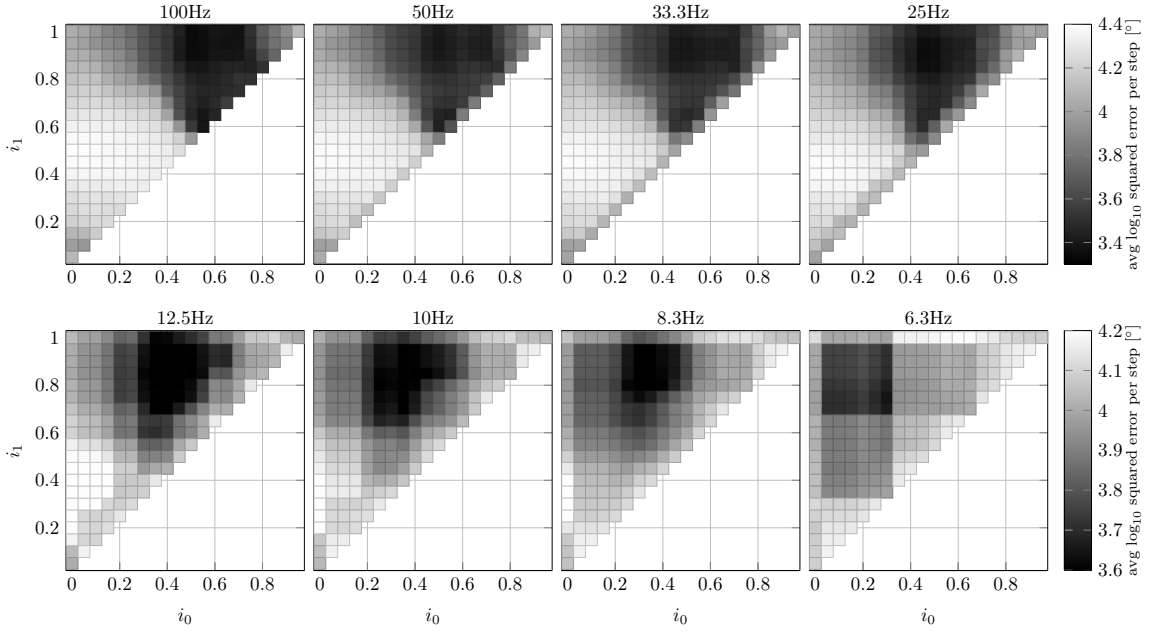

Figure 6.19 Model Parameter Determination at different Sampling Rates: Jacket. This figure shows the model parameter optimization results in terms of the average $\log_{10}$ squared error per step over all jacket testruns at different sensor data sampling rates. The sampling rates span a range between full 100Hz and 6.3Hz.
We can see, that our optimal chosen parameter ($i_0 = 0.525$, $i_1 = 0.95$) remains stable, but the area of "good" model parameters grows at lower sampling rates whereas the average error increases. Furthermore, we probably can observe discretization artifacts at frequencies below 8Hz.

6.3 Requirements fulfilled?

Movement determination. We have shown, that our method is capable of determining device movement by step detection. Furthermore, we succeed in providing a reliable heading estimation out of the internal sensor data in most cases.

Realtime capability. Our online implementation is capable of detecting steps and estimating the heading estimation at low latency. The latency depends on a configured sampling interval for our algorithm in general. Due to the restriction that our method depends on steps, intervals lower than typical step lengths simply make no sense. However, typical examples of the trousers location have a maximum step length of about 1 to 1.5 seconds or even less. The jacket location usually has a lower step length in terms of the time period. Thus, the latency is very limited.

Moreover, we give the possibility to poll the current heading while we are not moving (see Section 4.2.7 and Section 5.6.1.5).

Reaction on changes. Our online implementation is capable of changing the used model ad hoc as we determine the type to use out of the local variance in linear z-acceleration. This leads to solely false model usage in general, but the impact is limited as shown in Section 6.2.6. Thus, our approach will probably be useful even when the device position is changing while heading estimation.

Memory consumption. The online algorithm stores all gathered sensor data, but only keeps a configurable time-period. This means, that we delete outdated sensor data if the timestamp exceeds a configurable threshold. We have seen in Section 6.2.3.1, that pre-smoothing leads to overall worse results. Therefore we conclude, that any data aggregation will typically represent some pre-smoothing and hence, will lead to worse heading estimation results. This means, that we can only reduce the amount of stored data at a tradeoff to accuracy.

Computational costs. The computational time of the offline approach is irrelevant as we just use it for evaluation purposes. In practice, we can only use the online variant.

The implementation of our datacollection has a limited memory consumption. However, due to simplicity, we currently instantiate sensor-data wrapping objects in order to store them. Deletion of such an objects will make it unreferenced and hence, the java garbage collector will clear its memory at any time in the future which takes computation time. A solution to avoid this, is reusing outdated wrapper-instances.

A prominent and costly operation is the rotation-matrix determination at our sensor fusion module. Furthermore, we have to rotate each single acceleration data item in order to get its extrinsic representation which involves a costly 3 by 3 matrix multiplication.

Our complete heading estimation pipeline consists of three general phases which are device location determination for model distinction, step detection and the actual heading estimation.

The location determination calculates the variance over a sliding window of the linear z-acceleration with a configurable size. We do this at each execution of our algorithm. However, the variance calculation can be speed up by using an incremental approach.

In order to detect steps, we use an Finite Impule Response (FIR)-filter to smooth the z-accelerations, where the used filter mask depends on the currently used model. Because of a possible model change, we do the actual smoothing when not before needed, i.e. at each execution of our algorithm. The actual step detection then uses a second (small-filter mask) FIR filtering instance for calculating the derivative of z-accelerations. However, local extrema determination out of this data is a more exhausting task because of multiple data iterations.

For the actual heading estimation, we request the sensor values at an interval of interest (depending on the currently used model) and calculate an angle from the linear x- and y-acceleration integral. Depending on model parameters, the amount of used data varies. Currently, due to optimization results, we only use small intervals.

6.4 Heading Estimation - Comparison to Related Work

In general, the research community has only rarely focused on the topic of heading estimation at different device locations till now - at least to our knowledge. Although e.g. Park et al. [40] and Randell et al. [44] use locations different to the handheld position, they do not give any information about heading estimation errors, but give results on dead reckoning in general. This fact disallows us to compare our results on the heading estimation towards theirs.

In the context of FootPath, it is possible to compare this approach with other dead reckoning techniques, but we decided not to do any quantitative evaluation for several reasons. First of all, FootPath uses sequence aligning for position estimation (see Section 2.5.2) which in fact typically enhances the pure heading estimations in a complex relationship. The positioning results will highly depend on the used aligning implementation. Tests have shown, that the current BestFit algorithm does not give accurate results in connection to our new heading estimation. The origin of this problem is most probable the currently sharp distinguishing scoring function. We consider MultiFit as highly experimental. Furthremore, it takes a lot computation time. FirstFit provides worse results than all other methods in genal and thus, has not been tested.

Moreover, Footpath uses a fixed step length for the time being (see Section 2.5.1). This will cause the complete system to be error prone towards distance measurements in general - especially, if this step length is not suitably defined for the current user. This fact complicates a reasonable evaluation.

6.5 New Navigation Interfaces

We have discussed alternate navigation interfaces in Section 3.3.2. We decided to introduce two new feedback types which are presented in Section 4.3.3.

As an analysis of "good" feedback types via different media is a very nontrivial task, we only provide a framework which easily enables complex feedback rules and provides the possibility for verbal audio and vibrational feedback.

Such rules base on typical interest points which are extracted from map data like doors, level changes, stairs and significant bearing changes (see Section 4.3.2). Our introduced Metric Linear Temporal Logic (MLTL) logic (see Section 4.3.4) allows sophisticated scenario determination, e.g. checking if there is a door on the upcoming path at a range of 20m, but without any level changes until then.

However, we implemented some example rules which generally give simple turn by turn instructions. This is similar to commercial (car-)navigation systems, but with adapted feedback for pedestrian navigation purposes. We give these instructions either via the vibration interface or via audio. The vibration interface uses three patterns for left, right or an error (see Section 4.3.3.2), whereas the audio interface uses a Text-to-Speech (TTS) engine which enables to give more complex information (see Section 4.3.3.1).

Practical usage - Audio. In order to test our implementation, we did several test runs with at the typical handheld device location which is already implemented in FootPath. The navigation module works as expected and gives the feedback as desired.

However, for practical usage, we run into trouble because of the fixed step length. Feedback can get lost, is given too early or too late on the path due to the fundamental design of FootPath. If we set the step length too large, the navigation module assumes the user to be at a certain position where he should change the direction, e.g. turn left, although this does not correspond to the actual user position, but some future point on the navigation path. Moreover, if we set the step length too small, the user may miss actual bearing changes as the navigation module assumes the user to be at a path's previous position.

Even more critical is the aligning method of FootPath. Due to the sequence aligning and the resulting position estimation enhancing, FootPath may changed the estimated position drastically, i.e. jumps, and thus, the rules for navigation feedback may fail giving reliable feedback.

6.6 Summary

In this chapter, we have presented a very detailed evaluation on our new method for heading estimation. This includes the investigation on the device location estimation algorithm (6.1) as well as a sophisticated analysis of our new technique compared to other commonly used ones for handheld device locations (6.2.5)). In general, our approach shows comparable (or even better) results to GPS. Although we do the theoretical assumption of optimal flipping for the Magnetic and Rotation Vector method to make them even more competitive (6.2.2), our method outperforms them by far. The Direct Accelerations approach has proven to be worse than all others in most cases. Furthermore, a combined model for both locations, Jacket and Trousers, has been evaluated, but performs worse in comparison to our New Estimation technique.

Furthermore, we have discussed the different outliers in terms of false location detection and failed heading estimation in detail (6.2.7).

Due to the need of an online, adaptable realtime algorithm, we present results of our java online implementation. Generally, this variant suffers from local data knowledge, but results are comparable (6.2.6).

To generally improve results, we present a comparison of non-post-smoothing against simple post-smoothing method, which turned out to yield better results for all presented heading estimation approaches (6.2.3.2), whereas pre-smoothing does not result in better estimations.

Moreover, we provide an overview how our method behaves on reduced sensor data sampling rate (6.2.9.3). Generally, the accuracy of the heading estimation decreases when decreasing the sampling rate. However, to show that this is not a fact of a failing step detection (6.2.9.2) or wrong data classification (6.2.9.1), we show results of both modules at lower sampling rates which have proven to be robust even at very low frequencies. Moreover, we present model optimization results for lower sampling rates which lead to other model parameters at very low sampling rates (6.2.9.4).

A very important keynote is, that we use the same datasets for model fitting and evaluation. Although the collected data has been recorded from over 15 participants, it may not be representative. Nevertheless, due to the lack of more data, we decided to keep the approach in this constellation. Furthermore, exemplary testruns have been recorded and are presented (6.2.8) which reveal similar good results for our new estimation technique as seen in the overall evaluation.

We provide even more evaluation results in Appendix D.

7

Future Work

We have introduced a new generalized model for heading estimation which has proven to outperform other tested methods. However, there are several possibilities to enhance the system. In this chapter, we will give some short ideads on future work.

7.1 Device Location Determination

The device location estimation cannot always distinguish between the trousers and jacket location. This means, that the used feature (variance of linear z-acceleration) is not sufficient. Although the resulting outliers of our testing data show nevertheless mostly good results, it is possible to enhance the determination technique.

A good starting point for enhancing this estimation will probably be the observation of primary and secondary steps at the trousers pocket location which typically result in two different amplitudes of peaks. We have seen, that our step detection can reliably detect all peaks. Currently we filter for only the primary peaks by smoothing and a step detection timeout. However, a careful estimation on primary and secondary steps by tracking the peaks' amplitudes may lead to a more robust location estimation.

Another possibility will be using a Fourier Transform in order to detect major frequencies on the z-acceleration. Depending on the device location, e.g. trousers or jacket pocket, we can observe a major frequencies at roughly 1Hz for the trousers location and about 2Hz for the jacket location. As the frequency spectrum usually has multiple frequencies with similar power, we cannot simply distinguish between both locations by only regarding the most prominent frequency which means, that this approach needs further analysis. Besides using a Discrete Fourier Transform (DFT), a Discrete Cosine Transform (DCT) may also be suitable for this approach.

7.2 New Models

Our new approach considers two different new locations. Although the elaborated model instances may fit to other carrying locations, other new specialized model instances will most probable improve the heading estimation for new use cases. Model parameters can be determined by collecting training data and following our optimization approach.

We have shown how to distinguish between the new different locations - the trousers pocket and jacket pocket. However, the we use only the z-acceleration variance as a feature for this purpose, which is will not be sufficient to do a more sophisticated distinction between these two locations and others. Thus, introducing new model instances will also need to find more features for device location determination.

7.3 Individual Model Fitting and Learning

We have introduced a generalized model which may have different instances as well as specialized methods for device location distinction. Although our model has been trained with data of different persons, the typical walking pattern of a certain person may be totally different.

Manual Training. Nevertheless, it is possible to create a personal model for each individual. As a disadvantage, this will need training data directly from the user and will need some computation time on current smartphones for getting optimal parameters.

Learning. In order to create an adaptive model, there is need to create a database with training data. Data for this database will be provided each time using FoothPath for navigation. As the step detection works with a low error rate, whenever there is a segment of the navigation path being unique, i.e. there are no other walking possibilities, we can use these segments to improve the current model. Furthermore, it is possible to aid this kind of training with some more user interaction, where it can be selected after having reached a destination via navigation, whether there had been any remarkable wrong turns.

7.4 Better Rules for User-Feedback

We have presented a framework for determining the current users scenario on a given path and provide the possibility for vibrational and audio feedback. Furhtermore, we implemented some simple rules for user feedback. However, these rules are by far not trivial - not in a programmatric aspect, but in a psychological manner. The whole topic on human perception and cognition is more up to the field of psychology or communication science than computer science.

7.5 Adapt Step Length on the Fly

This work concentrates on heading estimation. Nevertheless, an accurate step length estimation is crucial for correct user position determination which is in turn important for non-visual navigation feedback. On the one hand, if the step length is estimated too large, the system will most probable provide navigation feedback too early. On the other hand, if the step length is estimated too small, the navigation feedback will be given too late.

For the time being, FootPath does not introduce any variable step length or an adaptive estimation technique, but a fixed size. However, we already introduced different approaches to this issue (see Section 3.2.4). Thus, using some kind of adaptive step length estimation will enhance the system.

7.6 Using Barometer

There has been research on also using barometer sensor data for better user localization. Related work shows, that the barometer sensor data is very realiable and thus, we can use it for detecting stairs and level changes exactly. Although the atmospheric pressure tends to change in general, this is a long-term process due to weather changes. Thus, this sensor gives useful information about relative altitude changes.

7.7 Combine Multiple Localization Technologies

Besides the Pedestrian Dead Reckoning (PDR) approach for indoor localization, there have been introduced several other methods which use e.g. Wireless Fidelity (WiFi), Radio-frequency identification (RFID) or Bluetooth. Some approaches also combine PDR with such other data sources. Incorporating more distinct data about locations will also enable the system to improve location estimation performance. Even only having single certain fixed points available will reset a possibly accumulated error in a PDR approach.

8

Conclusion

We have introduced a general method to enable an attitude independent movement detection using typical Microelectromechanical systems (MEMS) sensors available on todays smartphones. We have deducted a model for heading estimation from the typical human walking procedure and present a method for its parameter determination using optimization on given training data. Movement determination in general bases on three different tasks: step detection, heading estimation and step length estimation, whereas step length estimation is considered to be out of focus as there already have been introduced methods for this purpose by several researchers.

We introduced two exemplary model instances: for the trousers and the jacket pocket. For this reason, we elaborated on a simple feature for their distinction. Moreover, the model parameters are calculated by optimization on training data against two target functions - the least squares error and error variance between different testruns.

In order to get qualitative results, we evaluated our new methods in detail. We compare results towards three common other approaches which use either extrinsic direct linear acceleration values, the magnetic field in combination with accelerations (Magnetic) or the android internal fused Rotation Vector sensor. Although we assume theoretical optimal conditions for the Magnetic and Rotation Vector approach using flipping, our new approach outperforms all other presented techniques by far.

To generally improve the results, we investigated the impact of sensor data pre-smoothing at different magnitudes as well as simple post-smoothing on estimation results. The pre-smoothing does not yield better results at all, whereas post-smoothing improves the estimations for all regarded heading estimation approaches.

Our device location algorithm shows good overall results. However, rare testruns are classified wrongly, whereas using our jacket model on a trousers dataset (and the other way round) on the affected datasets show acceptable results anyway.

As most evaluation has been performed with an offline variant of our method, we also provide a comparison to our online algorithm. We can observe, that the location classification rarely fails due to limited data knowledge, but we consider this impact non-critical as results are similar to the offline variant.

The used datasets have been recorded at full possible sampling rate (100Hz) of the used device which have been used directly. In order to illustrate the behavior of lower sampling frequencies, we evaluated the regarded techniques at decreased sampling rates by up to factor 16 (6.3Hz). Results clearly show, that the performance of our heading estimation technique decreases with decreasing sampling rates in general. However, we consider our technique as still usable down to 25Hz. Due to similar optimization results on the lower resolution data, we conclude, that other model parameters will not improve these results significantly.

Besides the heading estimation, we also provide a framework for FootPath, that is capable of extracting navigation relevant information from given map data in order to create Interest Points for a predefined path automatically. We enable non-expert programmers to create sophisticated feedback rules by introducing Metric Linear Temporal Logic (MLTL) which allows for easy scenario determination.

Moreover, this framework tracks the current user position and enables the triggering of navigation feedback which is currently given either via vibration or verbal audio using a Text-to-Speech (TTS) engine. However, creating precise environment-dependent navigation hints in terms of human cognition are out of the scope of this thesis. Therefore, we solely provide a set of proof of concept feedback rules.

Bibliography

[1] Android API: SensorEvent. Online Resource. `http://developer.android.com/reference/android/hardware/SensorEvent.html`, accessed 2013/02/21.

[2] Indoor Mapping. Online Resource. `http://wiki.openstreetmap.org/wiki/Indoor_Mapping`, accessed 2012/02/16.

[3] Junit. Online Resource. `http://junit.org`, accessed 2013-03-21.

[4] MATLAB Product Website. Online Resource. `http://www.mathworks.de/products/matlab/`, accessed 2013-03-02.

[5] Marktforscher: Android und iPhone verdrängen die Konkurrenz. Online Resource, August 2012. `http://heise.de/-1663552`, accessed 2013/02/15.

[6] Bahl, P., and Padmanabhan, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, IEEE, pp. 775–784.

[7] Beauregard, S., and Haas, H. Pedestrian dead reckoning: A basis for personal positioning. In *Proceedings Of the 3rd workshop on Positioning, Navigation and Communication* (2006), pp. 27–36.

[8] Bitsch Link, J. A., Gerdsmeier, F., Smith, P., and Wehrle, K. Indoor navigation on wheels (and on foot) using smartphones. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Sydney, Australia* (11 2012), pp. 1–10.

[9] Bitsch Link, J. A., Smith, P., Viol, N., and Wehrle, K. Footpath: Accurate map-based indoor navigation using smartphones. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Guimaraes, Portugal* (9 2011), IEEE, pp. 1–8.

[10] Farrell, J., and Barth, M. *The global positioning system and inertial navigation*. McGraw-Hill, 1999.

[11] Feliz, R., Zalama, E., and Gómez, J. Pedestrian tracking using inertial sensors. *Journal of Physical Agents 3*, 1 (2009).

[12] Fink, A., Schröder, C., Schellin, A., and Beikirch, H. Embedded intertial measurement unit for real-time sensor integration and data processing. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* (November 2012), IEEE.

[13] Forsyth, D. A., and Ponce, J. *Computer Vision: A Modern Approach*, 1 ed. Prentice Hall, August 2002.

[14] Giudice, N. A., Bakdash, J. Z., and Legge, G. E. Wayfinding with words: spatial learning and navigation using dynamically updated verbal descriptions. *Psychological research 71*, 3 (2007), 347–358.

[15] Godha, S., Lachapelle, G., and Cannon, M. E. Integrated gps/ins system for pedestrian navigation in a signal degraded environment. In *In ION GNSS* (2006).

[16] Goodman, J., Gray, P., Khammampad, K., and Brewster, S. Using landmarks to support older people in navigation. *Mobile Human-Computer Interaction–MobileHCI 2004* (2004), 37–57.

[17] Groves, P. *Principles of GNSS, inertial, and multi-sensor integrated navigation systems*. GNSS technology and applications series. Artech House, 2008.

[18] Grädel, E. Course Notes: Mathematische Logik, 2009.

[19] Hide, C., Botterill, T., and Andreotti, M. Low cost vision-aided imu for pedestrian navigation. In *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)* (2010), IEEE, pp. 1–7.

[20] Holland, S., Morse, D. R., and Gedenryd, H. Audiogps: Spatial audio navigation with a minimal attention interface. *Personal and Ubiquitous Computing 6*, 4 (2002), 253–259.

[21] Hong, F., Chu, H., Wang, L., Feng, Y., and Guo, Z. Pocket mattering: Indoor pedestrian tracking with commercial smartphone. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on)* (2012), vol. 13.

[22] ITU-T Study Group 16. Video coding for low bit rate communication. Recommendation H.263, International Telecommunication Union, January 2005.

[23] Jahn, J., Batzer, U., Seitz, J., Patino-Studencka, L., and Gutiérrez Boronat, J. Comparison and evaluation of acceleration based step length estimators for handheld devices. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on* (September 2010), pp. 1–6.

[24] Jimenez, A., Seco, F., Prieto, C., and Guevara, J. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on* (August 2009), pp. 37 –42.

[25] Jirawimut, R., Ptasinski, P., Garaj, V., Cecelja, F., and Balachandran, W. A method for dead reckoning parameter correction in pedestrian navigation system. *Instrumentation and Measurement, IEEE Transactions on 52*, 1 (February 2003), 209–215.

[26] Kainulainen, A., Turunen, M., Hakulinen, J., and Melto, A. Soundmarks in spoken route guidance. In *Proceedings Intl. Conf. Auditory Displays (ICAD, Montréal, Canada)* (2007), pp. 107–111.

[27] Kang, W., Nam, S., Han, Y., and Lee, S. Improved heading estimation for smartphone-based indoor positioning systems. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on* (September 2012), pp. 2449 –2453.

[28] Keller, F., Willemsen, T., and Sternberg, H. Calibration of smartphones for the use in indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* (November), pp. 1–8.

[29] Kim, J. W., Jang, H. J., and Hwang, D.-H. A Step, Stride and Heading Determination for the Pedestrian Navigation System. *Journal of Global Positioning Systems 3* (2004), 273–279.

[30] Kim, Y., Shin, H., and Cha, H. Smartphone-based wi-fi pedestrian-tracking system tolerating the rss variance problem. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on* (2012), IEEE, pp. 11–19.

[31] Kothari, N., Kannan, B., and Dias, M. B. Robust indoor localization on a commercial smart-phone. Tech. Rep. CMU-RI-TR-11-27, Robotics Institute, Pittsburgh, PA, August 2011.

[32] Ladetto, Q., Gabaglio, V., and Merminod, B. Combining Gyroscopes, Magnetic Compass and GPS for Pedestrian Navigation. In *International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation (KIS), Banff, Canada* (2001), pp. 205–212.

[33] Li, Y., and Wang, J. A robust pedestrian navigation algorithm with low cost imu. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* (Novemver 2012), pp. 1–7.

[34] Lin, M.-W., Cheng, Y.-M., Yu, W., and Sandnes, F. E. Investigation into the feasibility of using tactons to provide navigation cues in pedestrian situations. In *Proceedings of the 20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat* (2008), ACM, pp. 299–302.

[35] Liu, J., Chen, R., Pei, L., Guinness, R., and Kuusniemi, H. A Hybrid Smartphone Indoor Positioning Solution for Mobile LBS. *Sensors 12*, 12 (2012), 17208–17233.

[36] Meier, R. *Professional Android 4 Application Development.* Wrox Programmer to Programmer. Wiley, 2012.

[37] Milette, G., and Stroud, A. *Professional Android Sensor Programming.* Wrox Programmer to Programmer. Wiley, 2012.

[38] Neumann, A. TIOBE-Sprachindex: Android verhilft Java zurück zur Spitze. Online Resource, February 2013. `http://heise.de/-1801281`, accessed 2013/02/21.

[39] Ojeda, L., and Borenstein, J. Personal dead-reckoning system for gps-denied environments. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on* (September 2007), pp. 1 –6.

[40] Park, K., Shin, H., and Cha, H. Smartphone-based pedestrian tracking in indoor corridor environments. *Personal and Ubiquitous Computing 17* (2013), 359–370.

[41] Pielot, M., Poppinga, B., and Boll, S. Pocketnavigator: vibro-tactile waypoint navigation for everyday mobile devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services* (2010), ACM, pp. 423–426.

[42] Pratama, A., Widyawan, and Hidayat, R. Smartphone-based pedestrian dead reckoning as an indoor positioning system. In *System Engineering and Technology (ICSET), 2012 International Conference on* (September 2012), pp. 1–6.

[43] Rai, A., Chintalapudi, K. K., Padmanabhan, V. N., and Sen, R. Zee: zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking* (New York, NY, USA, 2012), Mobicom '12, ACM, pp. 293–304.

[44] Randell, C., Djiallis, C., and Muller, H. L. Personal position measurement using dead reckoning. In *ISWC* (2003), pp. 166–175.

[45] Ruotsalainen, L., Kuusniemi, H., and Chen, R. Heading change detection for indoor navigation with a smartphone camera. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on* (2011), IEEE, pp. 1–7.

[46] Ruotsalainen, L., Kuusniemi, H., and Chen, R. Visual-aided two-dimensional pedestrian indoor navigation with a smartphone. *Journal of Global Positioning Systems 10*, 1 (2011), 11–18.

[47] Scarlett, J. Enhancing the performance of pedometers using a single accelerometer. *Application Note, Analog Devices* (2008).

[48] Schnelle, D., Lyardet, F., and Wei, T. Audio navigation patterns. In *EuroPLoP* (2005), pp. 237–260.

[49] Shin, B., Lee, J. H., Lee, H., Kim, E., Kim, J., Lee, S., su Cho, Y., Park, S., and Lee, T. Indoor 3d pedestrian tracking algorithm based on pdr using smarthphone. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on* (October 2012), pp. 1442 –1445.

[50] Skog, I., Nilsson, J.-O., and Handel, P. Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on* (September 2010), pp. 1–6.

[51] Slabaugh, G. Computing euler angles from a rotation matrix. Tech. rep., City University London, August 1999. `http://www.soi.city.ac.uk/~sbbh653/publications/euler.pdf`.

[52] Stark, A., Riebeck, M., and Kawalek, J. How to design an advanced pedestrian navigation system: Field trial results. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on* (2007), IEEE, pp. 690–694.

[53] Weinberg, H. Using the adxl202 in pedometer and personal navigation applications. *Analog Devices AN-602 application note* (2002).

[54] Wilson, J., Walker, B. N., Lindsay, J., Cambias, C., and Dellaert, F. Swan: System for wearable audio navigation. In *Wearable Computers, 2007 11th IEEE International Symposium on* (2007), IEEE, pp. 91–98.

[55] Ábrahám, E. Course Notes: Modeling and Analysis of Hybrid Systems, April 2011.

List of Figures

List of Tables

List of Abbreviations

AP Access Point

API Application Programming Interface

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DLT Direct Linear Transform

DMM Direct Matching Mode

DR Dead Reckoning

FIR Finite Impule Response

GPS Global Positioning System

GUI Graphical User Interface

HMM Hidden Markov Model

ICS Ice Cream Sandwich

IDE Integrated Development Environment

IMU Inertial Measurement Unit

INS Inertial Navigation System

JNI Java Native Interface

JOSM Java OpenStreetMap Editor

LMM Lookahead Matching Mode

LP Low-Pass

LTL Linear Temporal Logic

MEMS Microelectromechanical systems

MLTL Metric Linear Temporal Logic

NN Neural Network

OHA Open Handset Alliance

OS Operating System

OSM Open Street Map

PDR Pedestrian Dead Reckoning

RANSAC random sample concensus

RFID Radio-frequency identification

RMSE Root Mean Squared Error

RSSI Received Signal Strength Indicator

RWTH Rheinisch Westfälische Technische Hochschule Aachen

TTS Text-to-Speech

WiFi Wireless Fidelity

XML Extensible Markup Language

ZC zero crossing

ZUPT Zero Velocity Update

A

Survey Results - Where do you carry your smartphone?

This chapter contains a more detailed overview of our survey about smartphone carrying loations. We present the complete results in Table A.1 and Table A.2.

Location	Always [%]			Sometimes [%]			Never [%]		
	m	*f*	*both*	*m*	*f*	*both*	*m*	*f*	*both*
Trouser Pocket (front)	80	18	63	15	32	19	5	50	17
Trouser Pocket (back)	5	4	5	8	25	13	87	71	83
Jacket Pocket (outer)	3	11	5	35	61	42	63	29	53
Jacket Pocket (inner)	5	0	4	40	32	38	55	68	58
Belt	0	0	0	0	0	0	100	100	100
Arm	0	0	0	4	0	3	96	100	97
Shirt Pocket	0	0	0	12	7	11	88	93	89
Backpack	4	46	6	24	43	29	72	46	65
Bag	3	61	18	12	32	17	85	7	64

Table A.1 Survey results "Where do you typically carry your smartphone?". Each row describes the answers for a certain location. Three big columns represent the possible answers *always*, *sometimes* and *never*. The location and answer possibilities are split into three different sets - only male (m), only female (f) and all (*both*) participants. The number of participants is 103.
There are not many locations which are widely used. Overall, the most preferred locations are trouser pocket and jacket pocket. We notice a difference between male and female participants at trouser pocket and bag. However, most answers are alike.

Location	Definitley [%]			Maybe [%]			No Chance [%]		
	m	*f*	*both*	*m*	*f*	*both*	*m*	*f*	*both*
Trouser Pocket (front)	88	45	77	5	32	13	7	21	11
Trouser Pocket (back)	11	29	16	19	14	17	71	57	67
Jacket Pocket (outer)	28	64	38	40	18	34	32	18	28
Jacket Pocket (inner)	43	50	45	41	36	40	16	14	16
Belt	3	7	4	37	14	31	60	79	65
Arm	5	0	4	21	36	25	73	64	71
Shirt Pocket	9	4	8	28	36	30	63	61	62
Backpack	23	54	31	33	21	30	44	25	39
Bag	13	82	32	35	14	29	52	4	39

Table A.2 Survey results "What place would also be acceptable?". Each row describes the answers for a certain location. Three big columns represent the possible answers *definitely*, *maybe* and *no chance*. The location and answer possibilities are split into three different sets - only male (m), only female (f) and all ($both$) participants. The number of participants is 103.
The most acceptable locations are trouser pocket (front) and jacket pocket (inner and outer). We also observe good acceptance rates for the bag and backpack location.

B

Heading Ambiguity at Magnetic Field and Accelerations

In order to calculate a heading towards north, we will calculate a rotation matrix which transforms intrinsic coordinates into extrinsic coordiantes. We do this by calculating the world-axis basis vectors out of the acceleration data and the magnetic field data as both act as a fixed point in worl coordinates. We illustrate the calculated vectors in Figure B.1 [51].

B.1 Rotation Matrix

We define these sensor values as follows:

$$A = (a_x, a_y, a_z)^t \tag{B.1}$$
$$M = (m_x, m_y, m_z)^t \tag{B.2}$$

We now assume using the definition presented in Figure 2.3 which defines orthogonal axes. First of all, the accelerations already define our extrinsic z-axis as the gravity directs towards the ground:

$$Down \coloneqq A \cdot \|A\|^{-1} \qquad = A \cdot \sqrt{a_x^2 + a_y^2 + a_z^2}^{-1} \tag{B.3}$$
$$= (d_x, d_y, d_z)^t \tag{B.4}$$

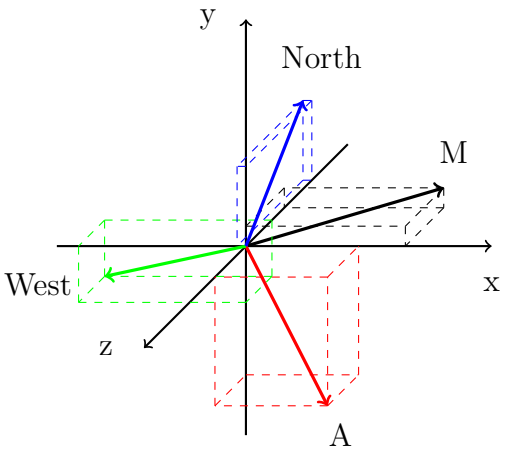

Figure B.1 Schematic Overview: Rotation Matrix Calculation. This figure depicts the used vectors for the calculation of the rotation matrix which transforms intrinsic to extrinsic coordinates. The coordinate system defines the intrinsic coordinates. We denote the direction of accelerations, i.e., gravity, with A (red), whereas the direction of the magnetic field is denoted as M. We also present the resulting directions West (green) and North (blue) which depict the extrinsic coordinate system together with the accelerations vector. Note, that the vectors are generally normalized.

Furthermore we calculate the west-direction which will represent our x-axis in extrinsic coordinates by taking the cross product of the magnetic field with the accelerations which is normalized afterwards.

$$West' \coloneqq A \times M \tag{B.5}$$

$$= \begin{vmatrix} m_x & m_y & m_z \\ d_x & d_y & d_z \\ e_x & e_y & e_z \end{vmatrix} \qquad = \begin{pmatrix} m_x d_y - m_z d_y \\ m_y d_z - m_y d_z \\ m_z d_y - m_y d_x \end{pmatrix} \tag{B.6}$$

$$West \coloneqq West' \cdot \|West'\|^{-1} \tag{B.7}$$

$$= (w_x, w_y, w_z)^t \tag{B.8}$$

Moreover, having the west direction, we can get the direction towards north by calculating the cross product of the west direction and the accelerations. The resulting vector will describe the y-axis of the extrinsic coordinate system.

$$North \coloneqq Down \times West \qquad = (n_x, n_y, n_z)^t \tag{B.9}$$

As we now have all three distinct orthogonal axes, we define the rotation matrix transforming intrinsic to extrinsic coordinates as follows, whereas each column of this matrix defines a basis change:

$$R \coloneqq \begin{pmatrix} w_x & n_x & a_x \\ w_y & n_y & a_y \\ \underbrace{w_z}_{x} & \underbrace{n_z}_{y} & \underbrace{a_z}_{z} \end{pmatrix} \tag{B.10}$$

B.2 Heading

We will show, that extracting the heading towards north is usually ambiguous. We demonstrate this by analyzing 3d rotation matrices in general [51]. Each three-dimensional rotation consists of three rotations around different axes at different angles. We denote these as $R_x(\psi)$, $R_y(\theta)$ and $R_z(\phi)$:

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix}, \qquad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, \tag{B.11}$$

$$R_z(\phi) = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.12}$$

However, these rotations can be combined by applying matrix multiplication. Note, that these rotations are not commutative. We introduce generic identifiers for each matrix entry due to the need of calculating on single cells later on. The results of the combined rotation matrix then looks as follows:

$$R = [R_y(\theta)R_x(\psi)]R_z(\phi) \tag{B.13}$$

$$= \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{B.14}$$

$$= \begin{pmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\theta\sin\phi & \cos\psi\sin\theta \\ \cos\psi\sin\phi & \cos\psi\cos\phi & -\sin\psi \\ \sin\phi\sin\psi\cos\theta - \cos\phi\sin\theta & \sin\theta\sin\phi + \cos\theta\cos\phi\sin\psi & \cos\psi\cos\theta \end{pmatrix} \tag{B.15}$$

As we only want to extract the heading towards north, i.e., the y-axis angle, we first calculate the angle θ using the matrix entry R_{31}:

$$R_{23} = -\sin\psi \qquad \Leftrightarrow \psi = -\arcsin R_{23} \tag{B.16}$$

This equation yields two different valid solutions in general (whenever $R_{23} \neq \pm 1$ hols). We will stick to this case now and exclude those, wehre the solution is unambiguous. This result it the typical reason for the two different valid headings mentioned in Section 4.2.4. Nevertheless, we get two different angles - ψ_1 and ψ_2 which will be exactly 180° into the opposite direction of each other:

$$\xRightarrow{R_{23}\neq\pm1} \psi_1 = -\arcsin R_{23}, \qquad \psi_2 = \pi - \arcsin R_{23} \tag{B.17}$$

Having determiend the first parameter, we can continue with the actually desired rotation angle ϕ which describes the rotation arount the y-axis, i.e., the heading towards north. Taking a closer look to the matrix entries R_{21} and R_{22} shows, that both solely contain the already determined ψ as wel as the heading angle ϕ. After a minor algebraic equivalence transformation, we get the following:

$$R_{21} = \cos\psi\sin\phi, \qquad R_{22} = \cos\psi\cos\phi \tag{B.18}$$

$$\sin\phi = \frac{R_{11}}{\cos\psi} \qquad \cos\phi = \frac{R_{21}}{\cos\psi} \tag{B.19}$$

Due to the ambiguity of sine and cosine, we have to solve this equation system. Note, that the fraction of cosine and sine describe the tangent. As the angle depends on the proportion of the length of the adjacent and the opposite leg, we can calculate it via the atan2 function as follows:

$$\phi = \text{atan2}\left(\frac{R_{21}}{\cos\psi}, \frac{R_{22}}{\cos\psi}\right) \tag{B.20}$$

This general formula will result in a unique solution, but due to the fact, that we have two possible values for θ, we get the following equations:

$$\Rightarrow \phi_1 = \text{atan2}\left(\frac{R_{21}}{\cos\psi_1}, \frac{R_{22}}{\cos\psi_1}\right), \qquad \phi_2 = \text{atan2}\left(\frac{R_{21}}{\cos\psi_2}, \frac{R_{22}}{\cos\psi_2}\right) \tag{B.21}$$

However, the angle θ has the same impact on both legs, but the primary unambiguity is a result of a sign change which also leads to both valid solutions ϕ_1 and ϕ_2.

C

MLTL Examples

We have seen in Section 4.3.4 how we can use Metric Linear Temporal Logic (MLTL) for our path-evaluation. We will now discuss some simple examples.

Reaching the destination. Assuming having reached the destination in a region of about 5m at the actual destination, a formula to express this may look like follows:

$$\mathcal{F}^{[0,5]}\, \mathcal{X}\, \mathit{false} \tag{C.1}$$

Bearing change but nothing different. Let us assume that we want to filter out simple bearing changes, but only if there is nothing special before or after this event in a range of 20m. Then the formula may look like this:

$$\Big(\mathcal{G}^{[0,20]}\, \neg\, (\text{Door} \vee \text{Stairs} \vee \text{LevelChange})\Big) \wedge \mathcal{F}^{[0,20]}\, \text{BearingChange} \tag{C.2}$$

Bearing change and door afterwards, but nothing other in between. To identify the scenario that we have a bearing change (in the next 15m) followed by a door to pass (in the next 10m), a formula for this would be the following:

$$\neg\, (\text{Door} \vee \text{Stairs} \vee \text{LevelChange}) \tag{C.3}$$

$$\mathcal{U}^{\,[0,15]}\, \Big(\text{BearingChange} \wedge \mathcal{X}\, \Big(\neg\, (\text{BearingChange} \vee \text{Stairs} \vee \text{LevelChange})\ \mathcal{U}^{\,[0,10]}\, \text{Door}\Big)\Big) \tag{C.4}$$

Nothing special, but some interesting facts. If there is nothing to happen on the current path, we can check whether we have some doors left or right of the path.

$$\Big(\mathcal{G}^{[0,50]}\, \neg\, (\text{Door} \vee \text{LevelChange} \vee \text{BearingChange} \vee \text{Stairs})\Big) \wedge \mathcal{F}^{[0,50]}\, \text{NearDoor} \tag{C.5}$$

D

Evaluation

This chapter will give more information for the different evaluations presented in chapter 6. These additional results allow a deeper understanding of the discussed results.

D.1 Device Location Determination

We have presented evaluation results on the device location determination approach in Section 6.1. However, a tabularly representation is presented in Table D.1.

		estimated type			
		trousers pocket		jacket pocket	
actual	trousers pocket	23	95.6%	1	4.2%
type	jacket pocket	3	12.0%	22	88.0%

Table D.1 Device Location Estimation Results. The absolute number depicts the amount of determined locations of test runs, whereas the percentage shows the fractional relation. The trousers location is well detected with one false detection sample, whereas the jacket location shows a higher error rate with an absolute number of three false detections which leads to an error rate of 12.0%.

D.2 Flipping

We have already mentiond the problem of flipping in Section 4.2.2 and Section 4.2.4. Due to get more competitive results, we have used flipping against the ground truth for evaluation purposes in general. Due to the fact, that the actual heading is not available, we also investigated on a simple method using Direct Accelerations. However, the overall presented in Figure D.1 results show, that flipping by this approach yields worse results. Moreover, we did not concentrate on this any further as the newly introduces method outperforms the other regarded techniques by far. Nevertheless, we provide error distribution of the flipping results in Figure D.2 and Figure D.3.

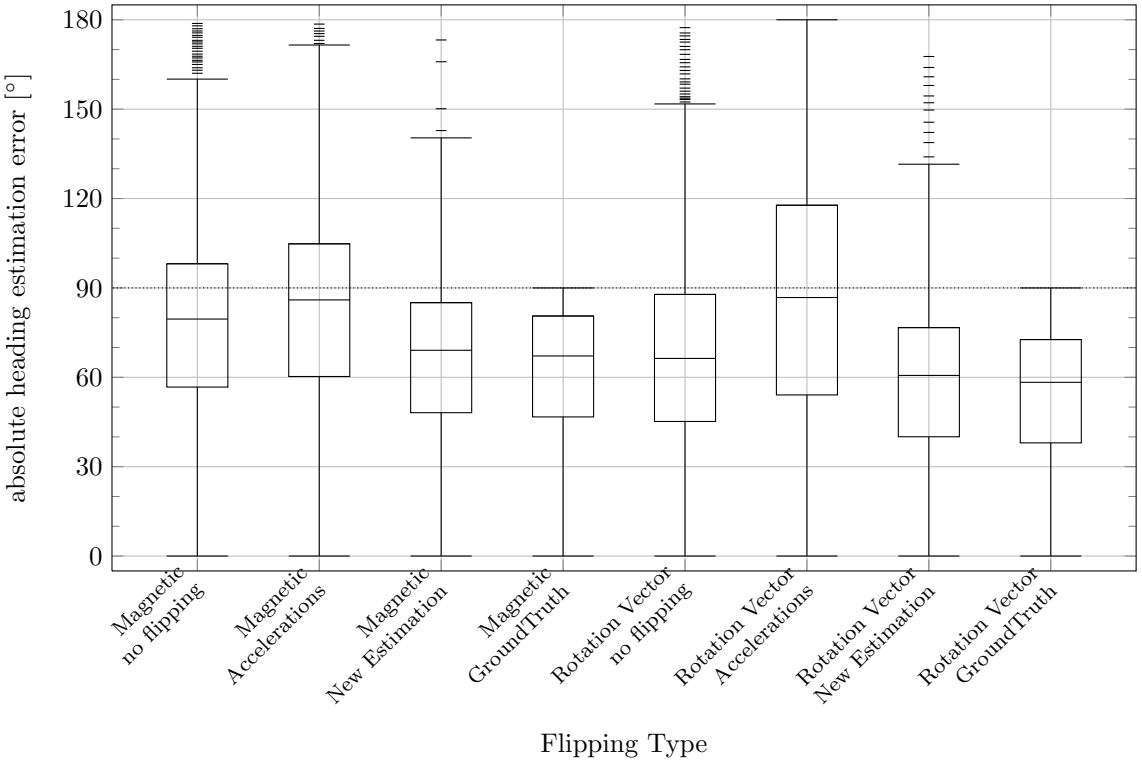

Figure D.1 Flipping Results. This figure shows a comparison of different measures for flipping.
We have discussed, that the Rotation Vector and Magnetic approach may fail in estimating the correct direction (see Section 4.2.2 and Section 4.2.4). For a correct determination if the calculated heading has to be flipped, we have to include movement direction data.
If we do not apply flipping, we notice a high error for Rotation Vector and Magnetic ($q_{.50} \gtrsim 70°$). Flipping by Direct Accelerations leads to even worse results, whereas the Magnetic technique together with the Direction Accelerations performs better than the Rotation Vector-Direction Accelerations combination. However, flipping by our new Estimation and ground truth yield best results ($q_{.75} \lesssim 70°$), whereas the flipping towards ground truth restricts the error to an absolute value of 90° and thus, performs slightly better. In general, the Rotation Vector provides better results than the Magnetic approach.

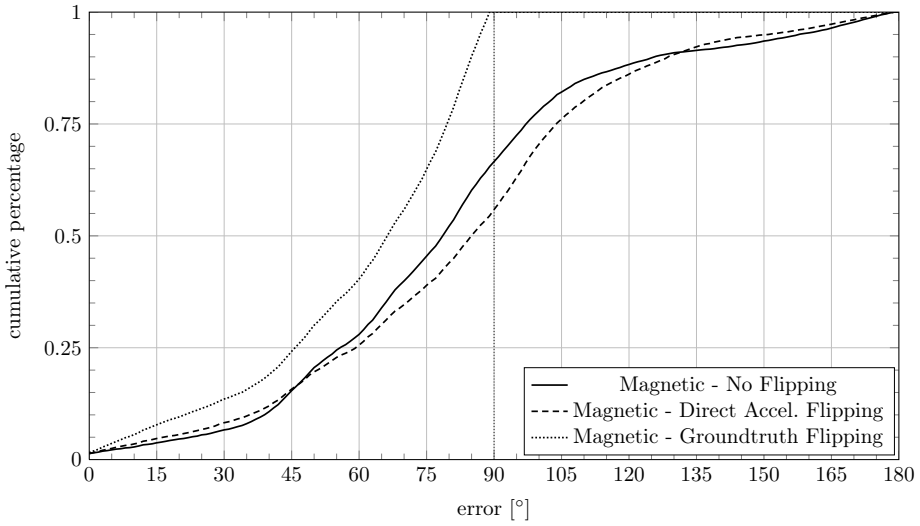

Figure D.2 Flipping Evaluation - Magnetic. This figure shows the complete error distribution of the regarded heading estimation techniques at the example handheld run. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
Flipping the Magnetic values by the ground truth yields best results. No flipping yields even better results than flipping by Direction Accelerations, but the results are quite similar with $q_{.25} \approx 60°$, $q_{.75} \approx 80°$ and $q_{.75} \approx 100°$.

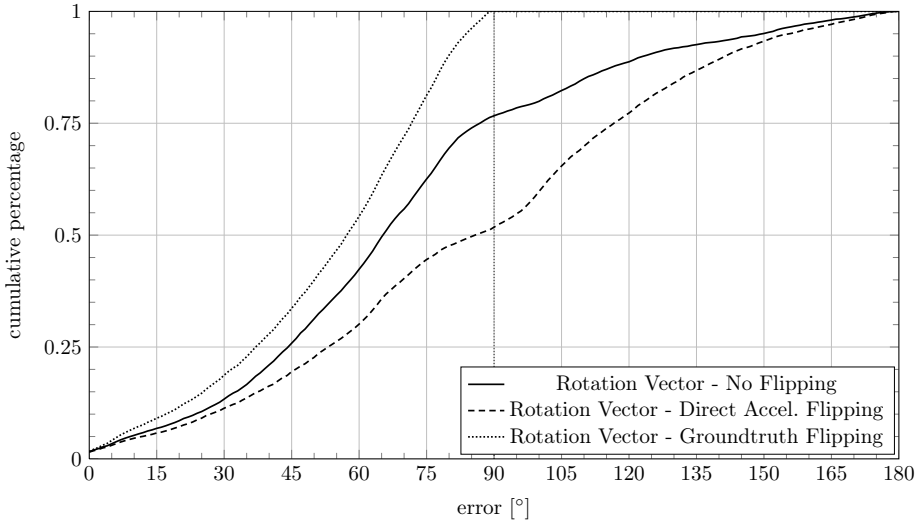

Figure D.3 Flipping Evaluation - Rotation Vector. This figure shows the complete error distribution of the regarded heading estimation techniques at the example handheld run. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
Flipping the Rotation Vector values by the ground truth yields best results. No flipping yields even better results than flipping by Direction Accelerations with a roughly linear increasing amount of errors until $q_{.75} \approx 85°$, whereas a quarter of the estimations show an error of more than 90°. The flipping by Direct Accelerations perform worse at $q_{.50} \approx 90°$ and $q_{.75} \approx 120°$.

D.3 Smoothing

In this section, we primarily discuss the influence of pre-smoothing. Moreover, we provde furhter statistics on post-smoothing.

D.3.1 Presmoothing

The presemoothing uses the unsmoothed data for step detection in order to keep the same step segmentation. However, the actual bearing estimation then uses smoothed data. This means, that we apply a gaussian filter at different sizes on each used dataset before doing any further calculation. This discrete gaussian filter uses a variance of $\sigma = \text{filterwidth}/4$ which normally yields a coverage of about 96% of the continuous gauss function, whereas the filterwidth depends on the target time period. Finally, we normalize the filter to make it numerically consistent. Note, that we apply postsmoothing on the results afterwards.

We present the results of presmoothing on our New Estimation method in Figure D.4. In general, presmoothing has no impact on the Rotation Vector and Magnetic approach, whereas the Direct Acceleration and New Estimation technique show worse results at all presmoothed datasets. In fact, these techniques already involve a kind of average smoothing by taking the mean measured heading of all sensor data items for a complete step sequence.

Further statistics separating our datasets into jacket and trousers location are given in Figure D.5, in Figure D.6. Overall results for the Magnetic, Rotation Vector and Direct Accelerations approach are shown in Figure D.7, in Figure D.8 and in Figure D.9, respectively.

D.3.2 Postsmoothing

We have alread discussed the impact of our postsmoothing method in Section 6.2.3.2. We provide further distinct information of the error distributions separated into the different models in Figure D.10 and Figure D.11.

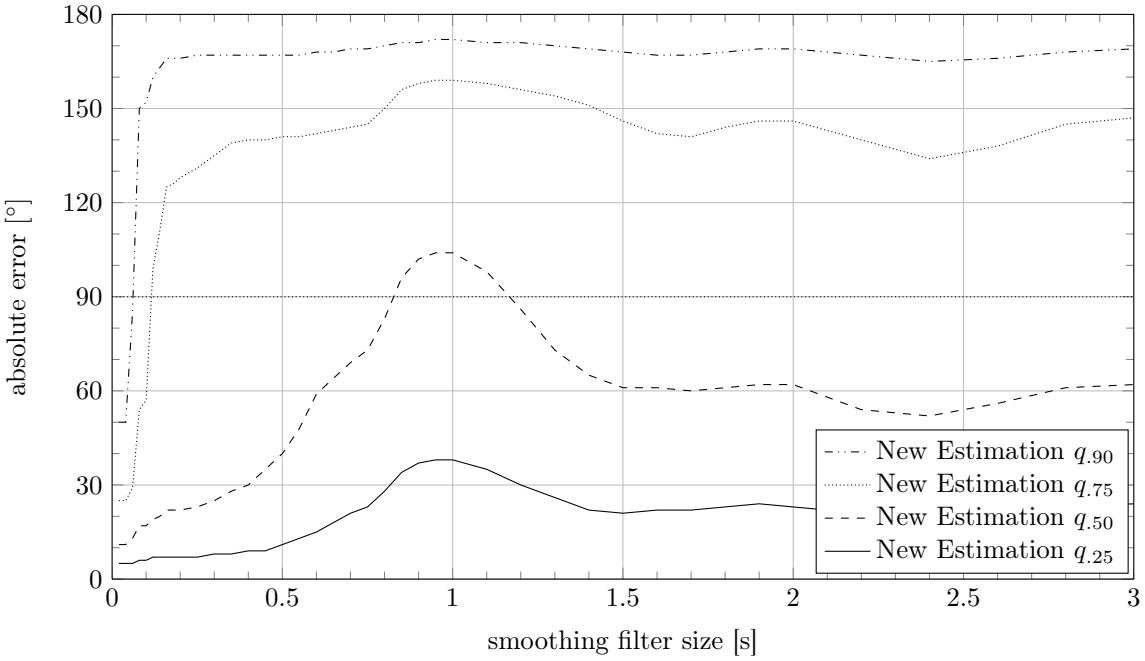

Figure D.4 Estimation Method Behavior at different Presmoothing Magnitudes - New Estimation. This figure shows different quantiles of the error distribution of the New Estimation method at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
We can see, that presmoothing does not improve the performance of our New Estimation method. However, the lower quantile as well as the median of the error distribution show a significant peak around filter sizes of 1s. We assume, that these filter sizes damp the typical major walking frequencies.

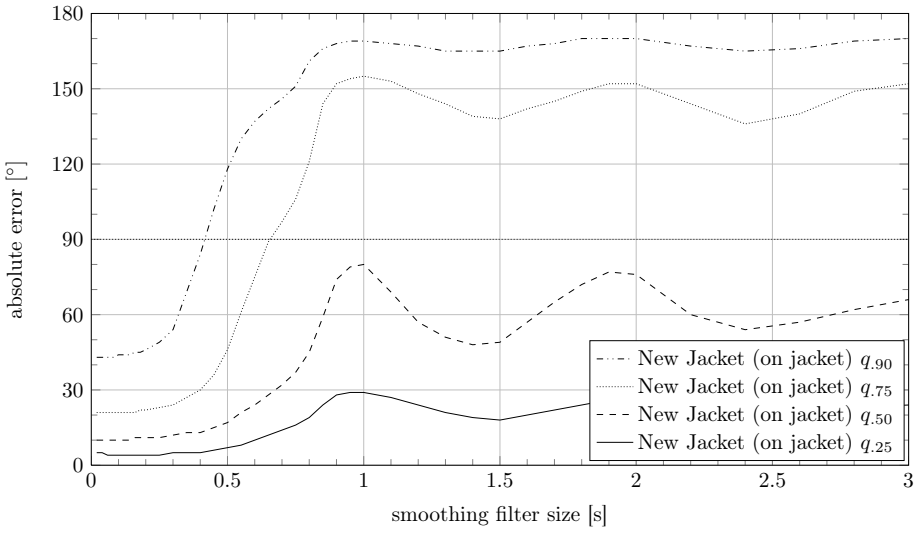

Figure D.5 Estimation Method Behavior at different Presmoothing Magnitudes - New Jacket (only Jacket Datasets). This figure shows different quantiles of the error distribution of the New Jacket method only on jacket datasets at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
The behavior of the different quantiles show, that presmoothing does not improve the performance of the New Jacket method at all. We can observe two peaks - one at filter size $1s$ and a second at filter size $2s$. We assume, that these filter sizes damp the major walking frequencies as already seen in Figure D.4.

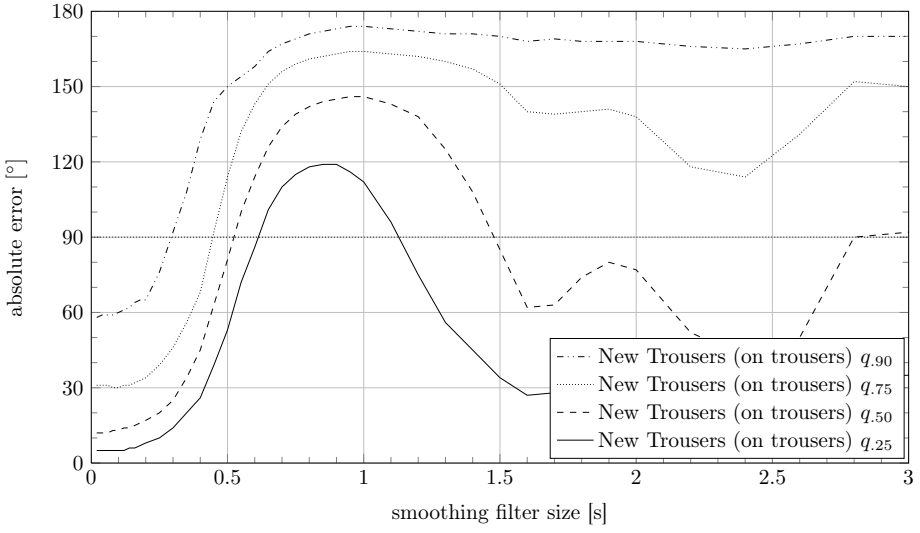

Figure D.6 Estimation Method Behavior at different Presmoothing Magnitudes - New Trousers (only Trousers Datasets). This figure shows different quantiles of the error distribution of the New Trousers method only on trousers datasets at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
The behavior of the different quantiles show, that presmoothing does not improve the performance of the New Trousers method at all. We can observe one extreme peak at filter size $0.7s$. We assume, that this filter size damps the major walking frequencies as already seen in Figure D.4.

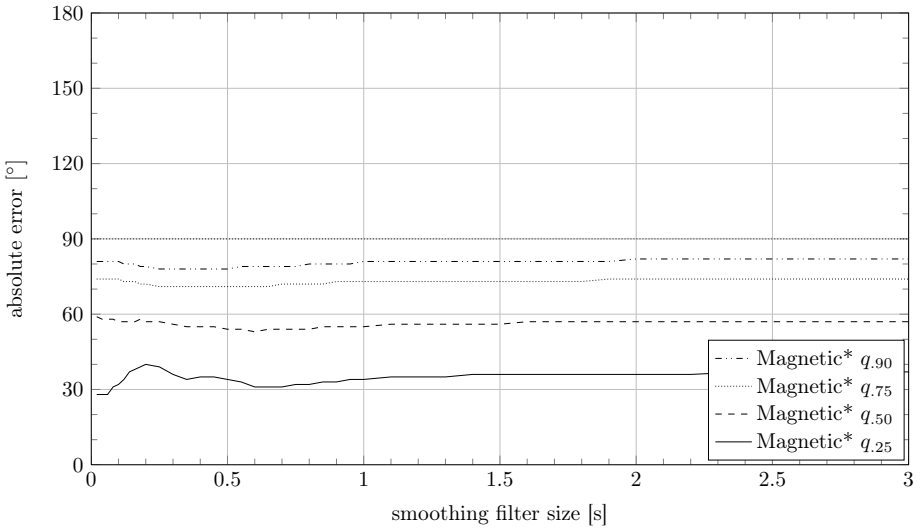

Figure D.7 Estimation Method Behavior at different Presmoothing Magnitudes - Magnetic*. This figure shows different quantiles of the error distribution of the Magnetic method at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
We cannot observe any significant change in performance of the Magnetic technique for different presmoothing values.

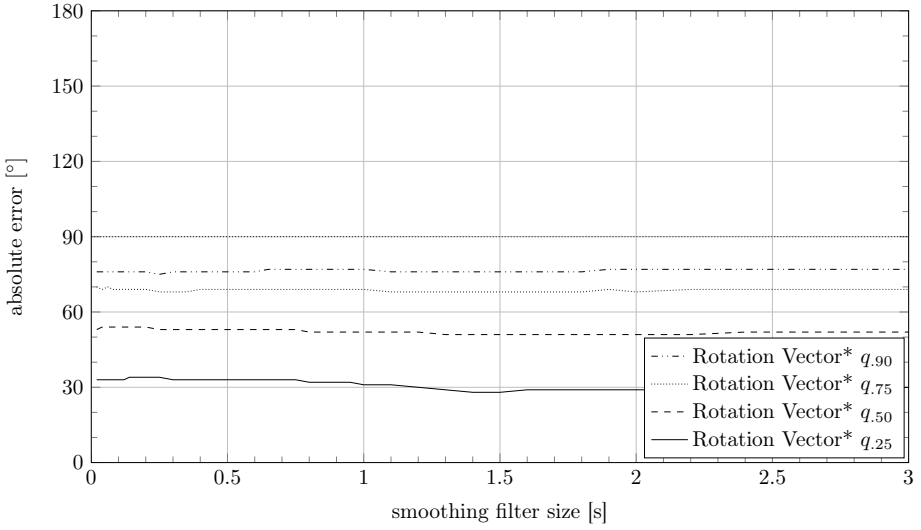

Figure D.8 Estimation Method Behavior at different Presmoothing Magnitudes - Rotation Vector*. This figure shows different quantiles of the error distribution of the Rotation Vector method at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
We cannot observe any significant change in performance of the Rotation Vector technique for different presmoothing values.

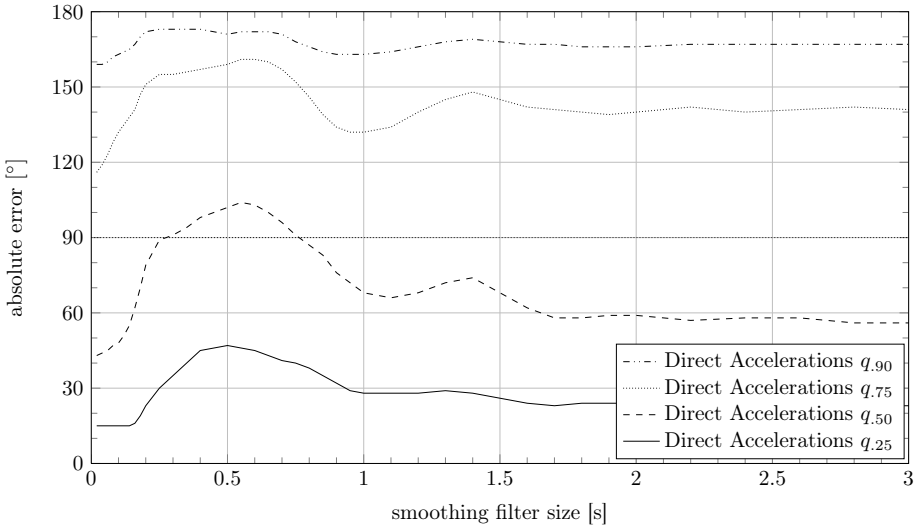

Figure D.9 Estimation Method Behavior at different Presmoothing Magnitudes - Direction Accelerations. This figure shows different quantiles of the error distribution of the Direction Accelerations method at different magnitudes of presmoothing. The y-axis depicts the error value of the quantile, whereas the x-axis denotes the used filter size in terms of time.
Presmoothing does not improve the performance of the Direction Accelerations approach. Furthermore, we see a peak in the statistical values of the error distribution at $0.5s$. However, we currently have no explanation for this behavior.

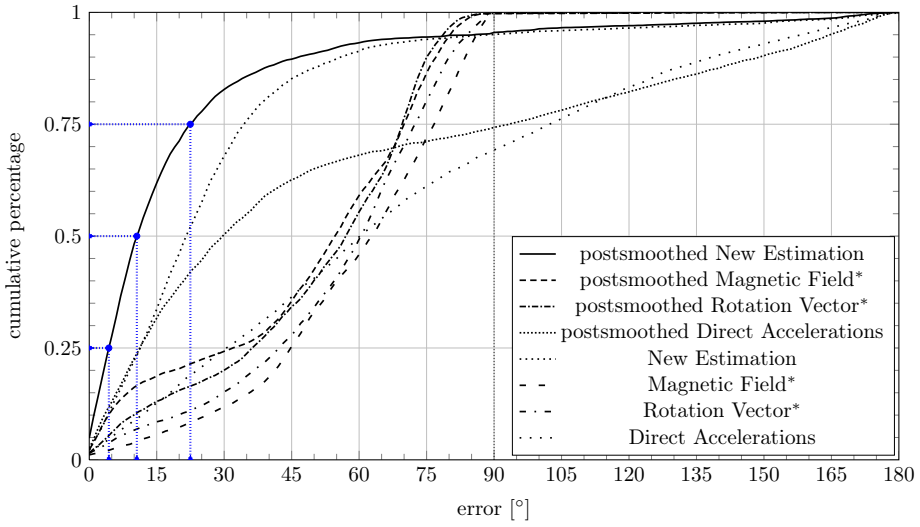

Figure D.10 Model Evaluation: Postsmoothing Jacket. This figure depicts the error distributions of the regarded heading estimation approaches with and without postsmoothing.
We can observe, that postsmooth drastically improves the performance of our New Estimation, pushing down the $q_{.75}$ quantile from 35° to 17°. Furthermore, the Direct Acceleration method is very improved with a change of the $q_{.50}$ from 60° down to 30°. The other methods are also improved, but not as significantly as both mentioned ones.
Note, that we have marked the quantiles of interest (q_{25}, q_{50}, q_{75}) for our New Estimation approach.

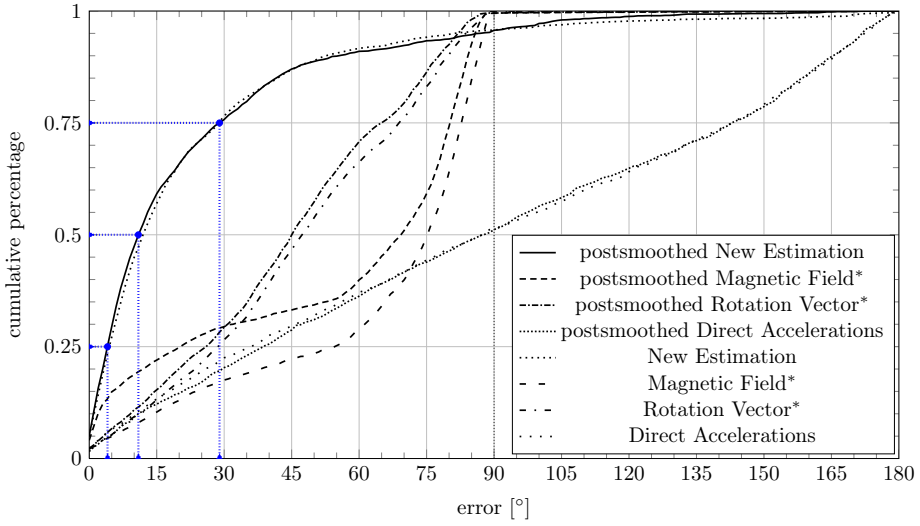

Figure D.11 Model Evaluation: Postsmoothing Trousers. This figure depicts the error distributions of the regarded heading estimation approaches with and without postsmoothing.
The postsmoothing at the trousers model does not improve performance as drastically as at the jacket model. The Rotation Vector method shows slightly better performance, whereas the Magnetic approach shifts its $q_{.25}$ quantile from 55° to 20°. Note, that we have marked the quantiles of interest (q_{25}, q_{50}, q_{75}) for our New Estimation approach.

D.4 Model Parameter Optimization

We have already discussed the optimization results for the distinct datasets in Section 6.2.4, i.e. trousers and jacket, which have resulted in two different models. Evaluation in Section 6.2.5 has shown, that each model does not fit for a good heading estimation for a dataset of the other type.

Nevertheless, we have also calculated the optimization results on the complete dataset in order to get a model that fits both locations. We present the results in Figure D.12. Furthermore, we present results on heading estimation with this combined model in comparison to our New Estimation method in Figure D.15. Note, that we used our device location estimation technique for step detection.

In general, we have best optimization results for the trousers model at a lowest mean squared error per step of 1163.7° which corresponds to an average absolute error per step of ≈ 34°. The jacket datasets result in a lowest mean squared error per step of 2165.2° which corresponds to an average absolute error per step of ≈ 47°. In contrast to this, the combined model shows a lowest mean squared error per step of 2626.5°. This results in an average error per step of ≈ 51°.

However, we can get especially for trousers datasets by far better heading estimations.

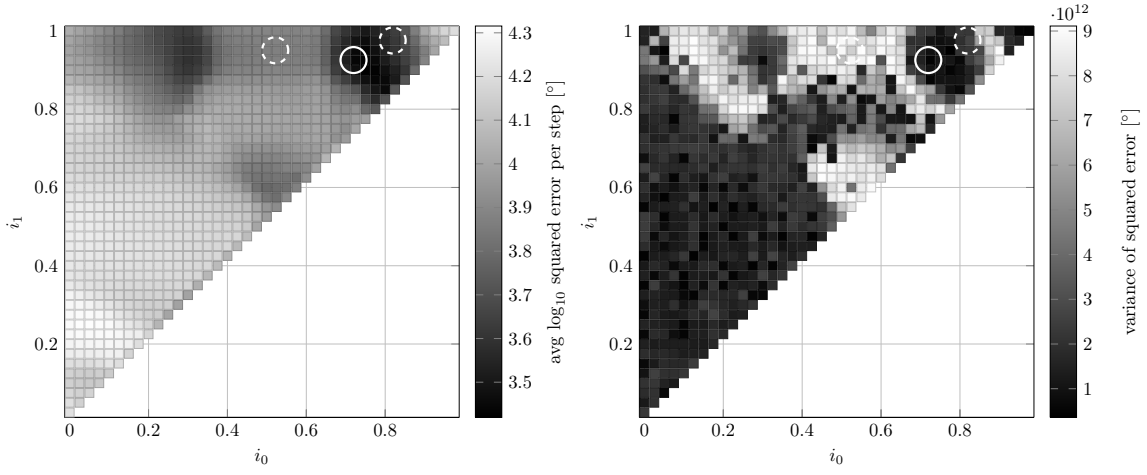

Figure D.12 Model Parameter Determination: Combined. This figure shows the results of model optimization for the all testruns. Both graphics show all valid model-intervals $[i_0, i_1[$ which determine the range of values for the heading estimation per step. Generally, darker quads relate to better results, whereas lighter quads relate to worse results.
Left: Each quad depicts the average $\log_{10}$ squared error per step in degree which is encoded into the gray-value.
Right: Each quad depicts the average variance of the sum of squared errors over all testruns. The results vary highly between subsequent possible discretized interval values. However, we have chosen our final model parameter by taking both values into account which is marked with the white circle at $i_0 = 0.725$ and $i_1 = 0.925$.
Note, that we have marked the optimal parameters with the white solid circle. The dashed circles mark our parameters for the distinctive models.

D.5 Model Evaluation

We have already discussed in Section 6.2.5 the evaluation on our newly introduced models. However, we provide a more detailed overview on the error distributions for the jacket model in Figure D.13 and for the trousers model in Figure D.14.

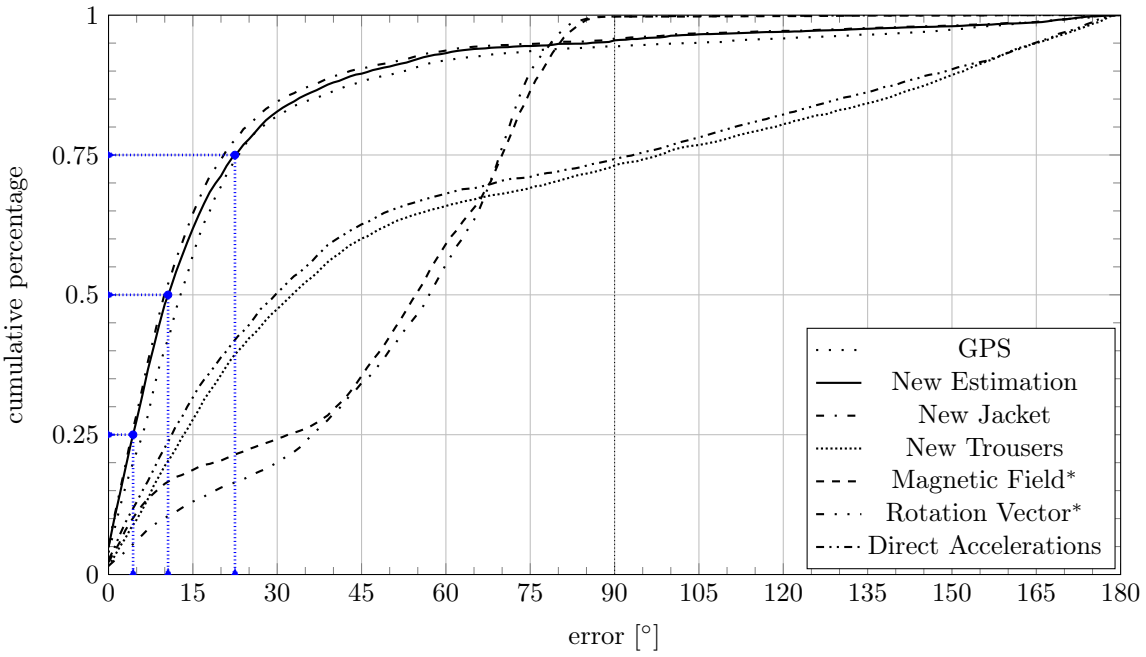

Figure D.13 Model Evaluation: Cumulative Error Distribution - Jacket. This figure shows the complete error distribution of the different regarded heading estimation techniques on all jacket datasets including Global Positioning System (GPS) as a reference. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
The restriction to use the jacket model as well as the New Estimation technique are almost similar to GPS - or even better. The Direct Accelerations method and our trousers model perform acceptable with $q_{.50} \lesssim 30°$, but the Rotation Vector and Magnetic approach outperform both at a percentage of about 65% with an error below 70°.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the New Estimation technique.

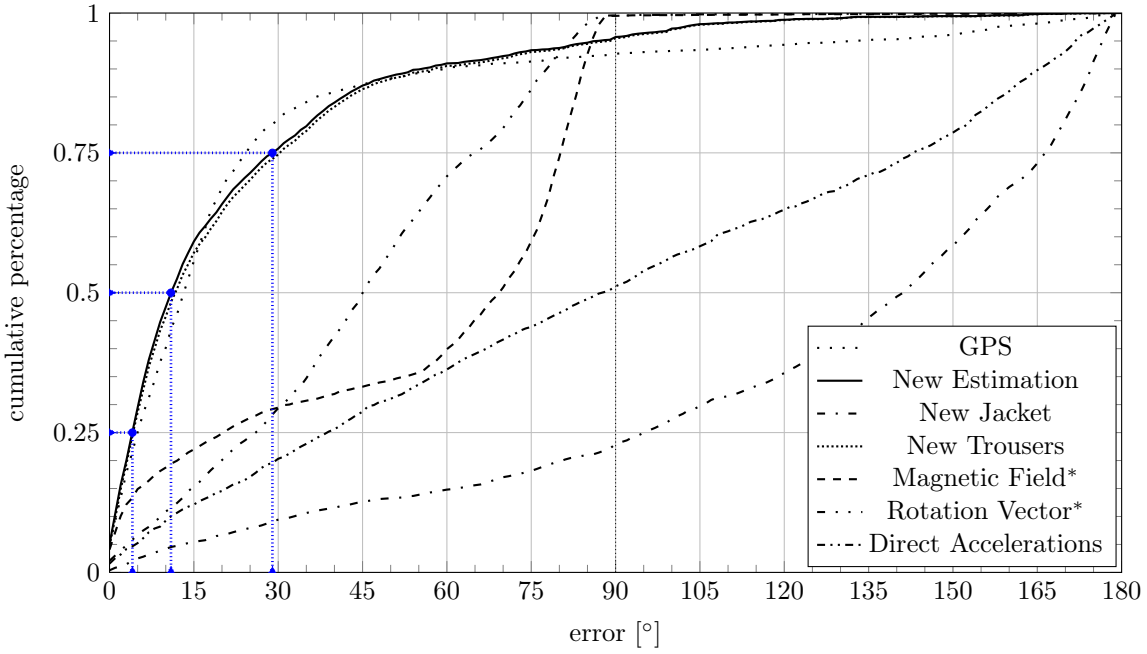

Figure D.14 Model Evaluation: Cumulative Error Distribution - Trousers. This figure shows the complete error distribution of the different regarded heading estimation techniques on all trousers datasets including Global Positioning System (GPS) as a reference. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error. The restriction to use the trousers model as well as the New Estimation technique are almost similar to GPS, but outperforms it at a percentage of about 90% with errors below 45°. The Direct Accelerations method and expecially our jacket model fail to provide a reliable heading estimation with a median of about 90° and 140°, respectively. Due to flipping, the Magnetic approach provides accurate results, but most errors are in a range of 60° to 90°. In contrast to this, the Rotation Vector performs significantly better with an even error distribution between 0° and 90°. Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the New Estimation technique.

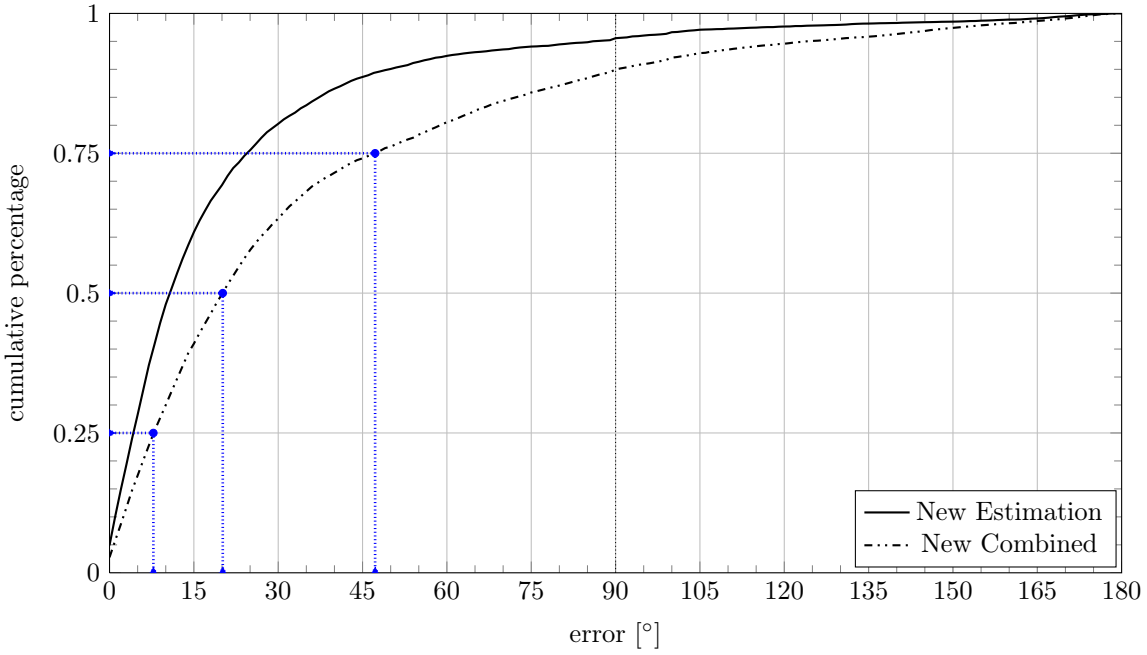

Figure D.15 Model Evaluation: Cumulative Error Distribution - Combined. This figure depicts the complete error distributions of the Combined model. We can see, that this approach provides a reliable heading estimation at an upper quantile of $q_{.75} < 50°$, but its results are inferior to our twofold New Estimation technique.

D.5.1 Outliers

We present results for falsely location estimated datasets in Figure D.16. Besides the statistical data, we provide a comparison between the falsely used model parameters in Figure D.17.

Moreover, to give a better overview, we also provide a boxplot of the errors at heading estimation with our New Estimation method for the outliers having an upper quantile of over 45° in Figure D.18.

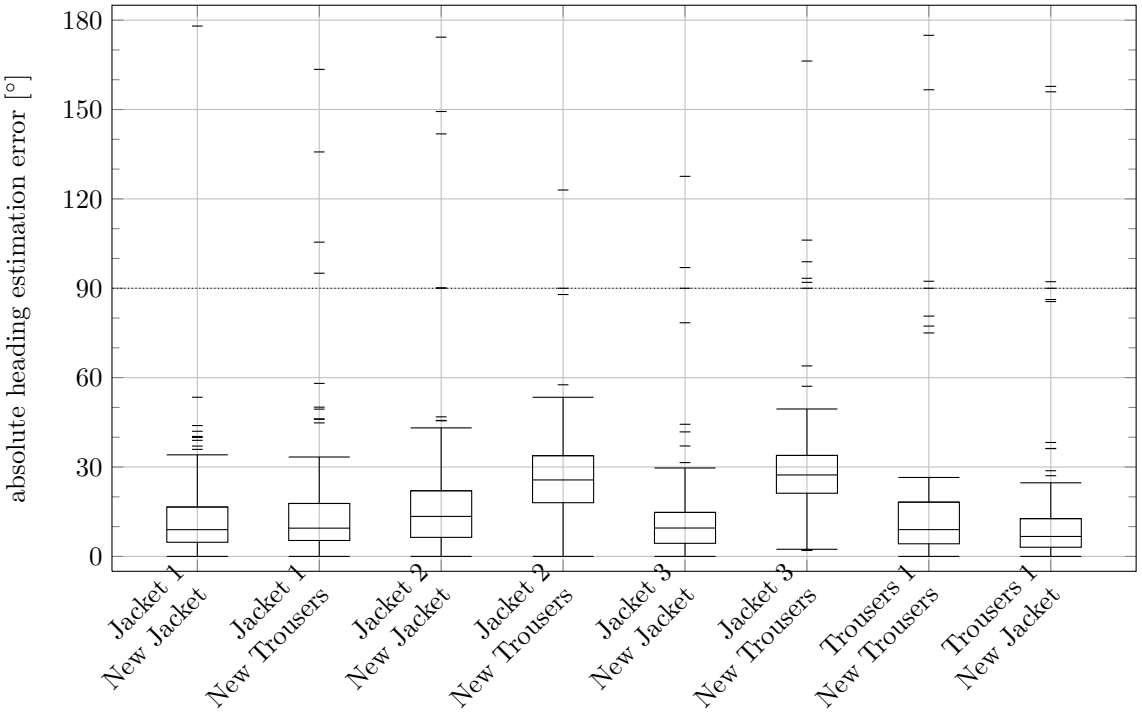

Figure D.16 False Model Selection: Comparison of Outliers. This figure shows a boxplot of all false detected datasets. For each dataset, we calculated the heading estimated twice, using the New Trousers and New Jacket model.
Generally, all results are good and show a small amount of outliers. However, the trousers model does not fit as good as the jacket model on the actually jacket datasets. The jacket model shows even better results than the trousers model on the falsely determined trousers dataset.

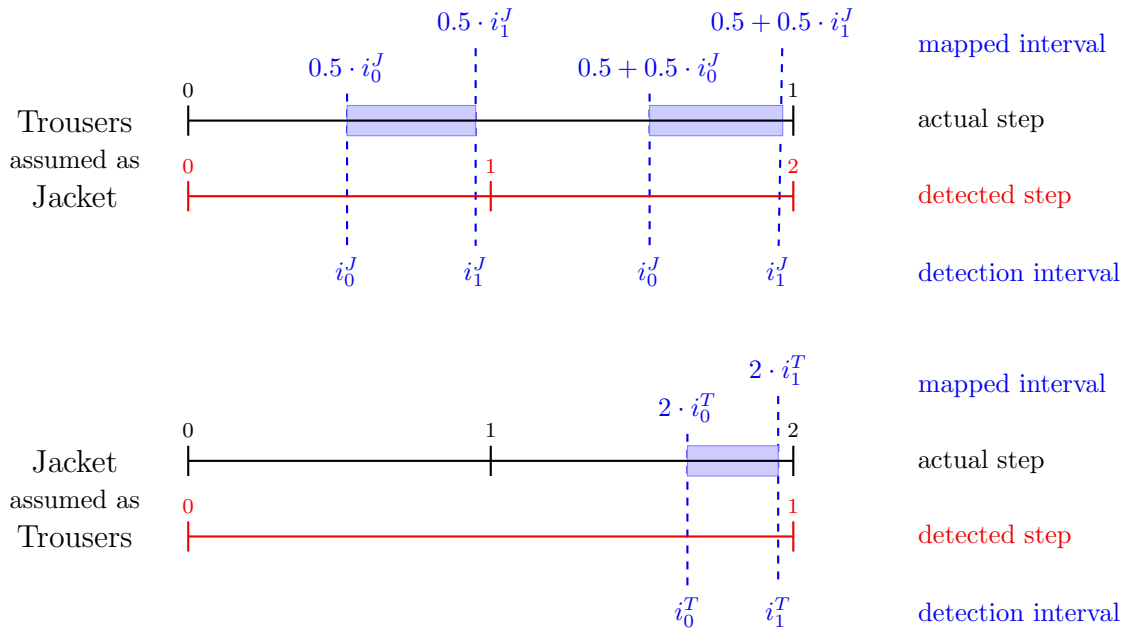

Figure D.17 Model Parameter Comparison - Trousers vs. Jacket model. This figure shows a generic parameter comparison between the trousers and jacket model under the assumption, that the step detection at the trousers model misses each second peak due to the distinction between secondary and primary leg. The black horizontal line depicts the time of the step period in terms of the actual model. The red horizontal line shows the time of the detected step period (i.e. the period of the assumed model). We have colored the used and transformed intervals blue. Note, that the shown intervals correspond qualitatively to our determined model.
Top: We can see, that a trousers dataset processed with our jacket model shows the doubled amount of steps - i.e. per actually one step (which would be the primary step), we detect two (primary and secondary). Moreover, our algorithm estimates a heading with the interval parameters from the jacket model for both of these detected steps. The blue rectangles mark the used intervals of the actual step period.
Bottom: We can see, that a jacket dataset processed with our trousers model shows only the half amount of steps - i.e. each second step is missed due to the higher step timeout. This results in a heading estimation for each two-step sequence. However, we marked the used interval at each second step blue.

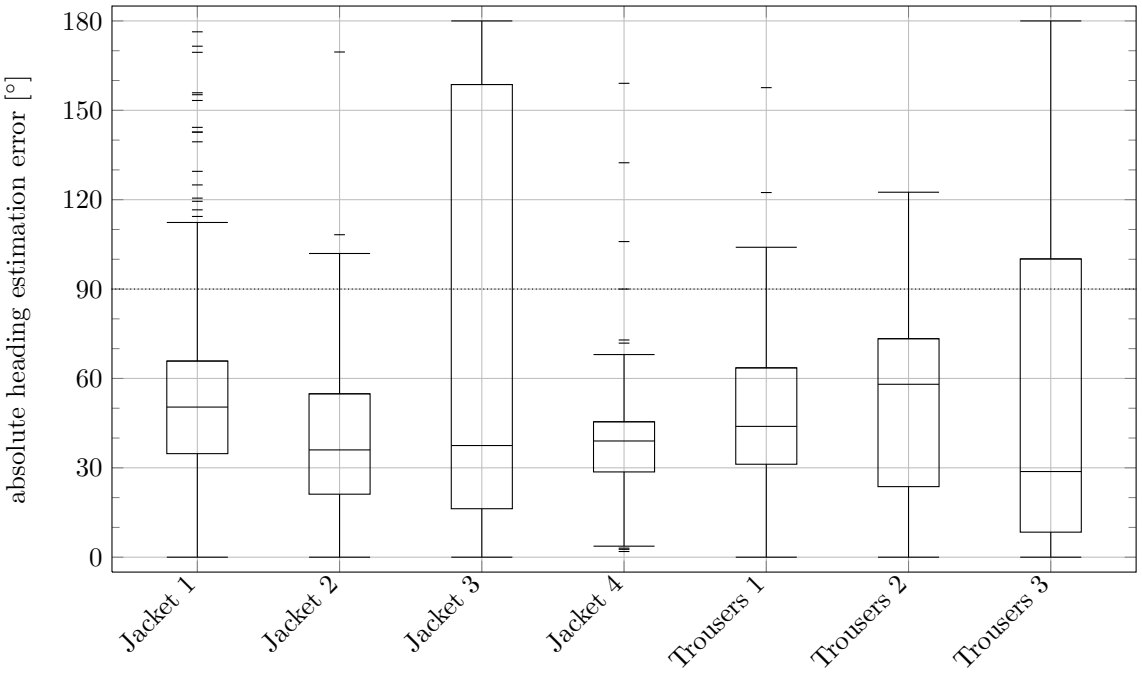

Figure D.18 False Model Selection: Comparison of Outliers. This figure shows the error distributions of out New Estimation approach at the estimation outliers having $q_{.75} > 45°$..
Out estimation method works quite good on most of these datasets with their median below 50° and an upper quantile mostly below $\approx 70°$. However, two exceptions are the Jacket 3 and Trousers 3 datasets which both provide a very good estimation on half of the data. In comparison, the estimation is a lot better at the Trousers 3 dataset in compared to the Jacket 3 dataset as its upper quantile settles at $\approx 100°$, whereas the Jacket 3 has the upper quantile at $\approx 160°$.

D.6 Model Evaluation at different Sampling Rates

We have already presented evaluation results of the different headings estimation techniques in Section 6.2.9.3. We provide more details on the Magnetic, Rotation Vector and Direct Accelerations approach at lower sampling rates in Figure D.19, D.20 and Figure D.21, respectively.

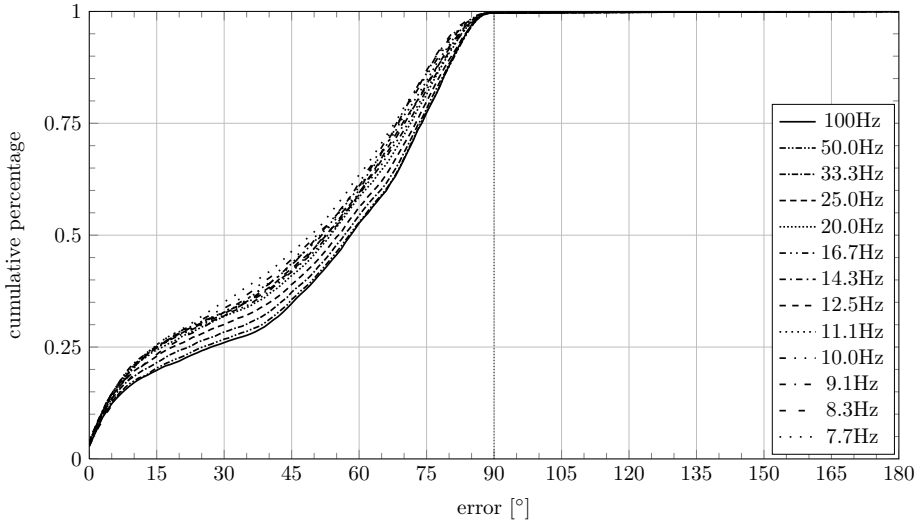

Figure D.19 Model Evaluation at different Sampling Rates: Cumulative Error Distribution - Magnetic*. This figure depicts the error distributions of the Magnetic approach at different sensor data sampling rates.
The general error distribution stays similar over all different frequencies. However, decreasing the sampling rate leads to better results for the Magnetic technique.

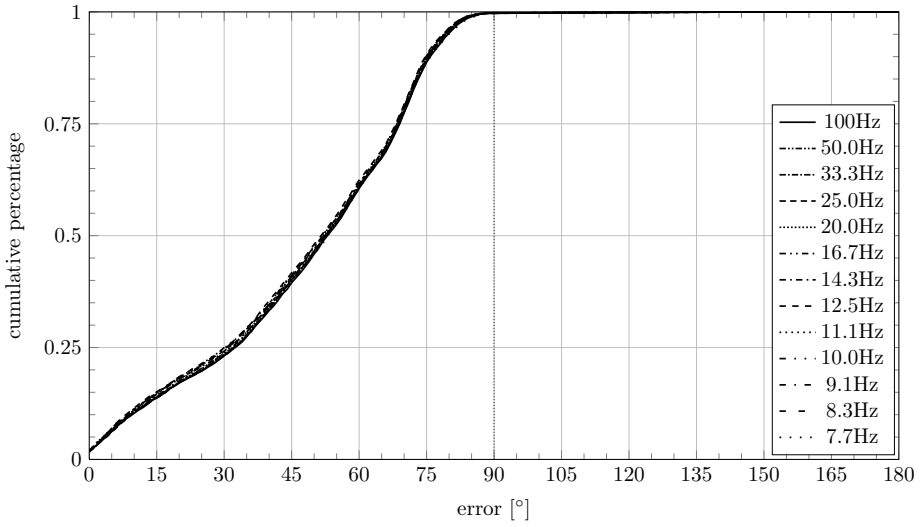

Figure D.20 Model Evaluation at different sampling rates: Cumulative Error Distribution - Rotation Vector*. This figure depicts the error distributions of the Rotation Vector approach at different sensor data sampling rates. The Rotation Vector technique remains robust over all regarded frequencies.

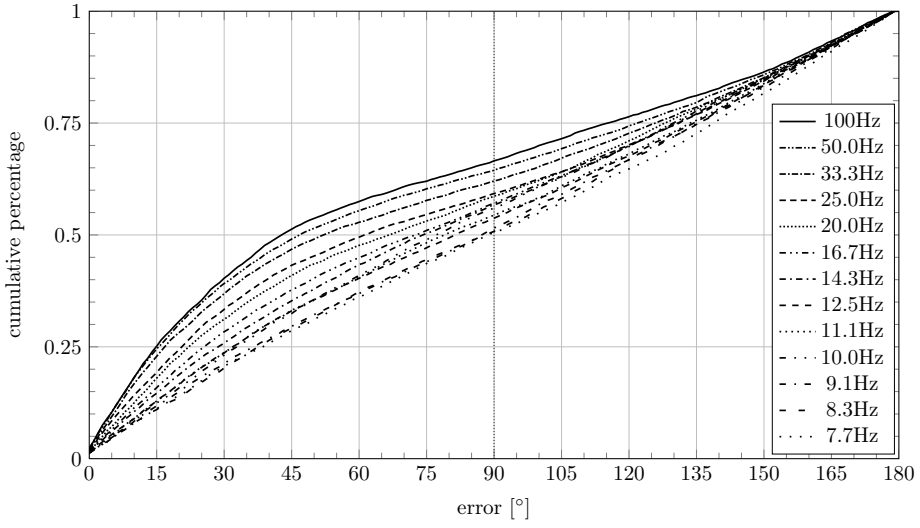

Figure D.21 Model Evaluation at different sampling rates: Cumulative Error Distribution - Direct Accelerations. This figure depicts the error distributions of the Direct Accelerations approach at different sensor data sampling rates. By decreasing sensor data, the amount of errors increases gradually until reaching a uniform distribution. In general, this method does not perform as good as any other presented one.

D.7 Comparison of Online and Offline variants

We have already discussed the comparison between our online and offline algorithm variant in Section 6.2.6. However, we provide a more detailed overview on the error distributions in Figure D.22.

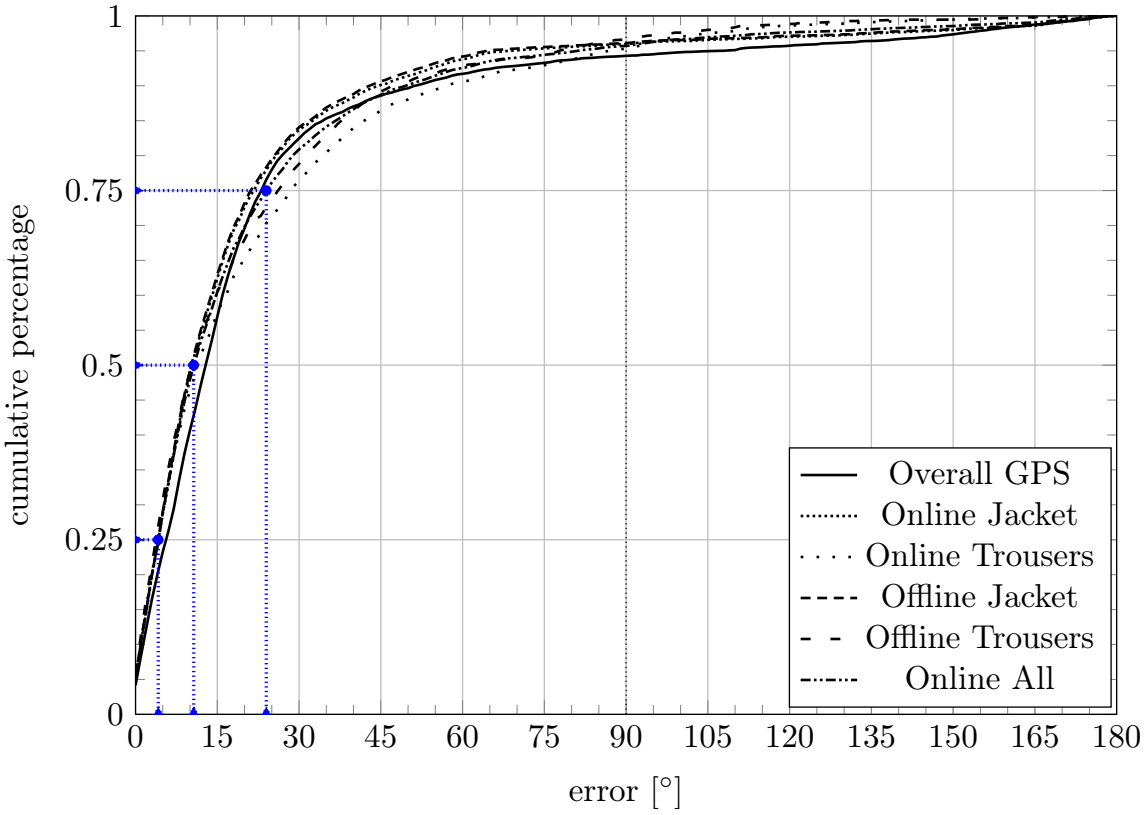

Figure D.22 Cumulative Error Distribution: Online vs. Offline. This figure shows the error distribution of our New Estimation technique in the online and offline variant. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error. The used datasets differ - the Global Positioning System (GPS) reference is calculated from all datasets, the Jacket approach only considers jacket datasets and the Trousers approach just uses trousers datasets, whereas we furthermore give a combination which consideres all dataset as well.
The error distributions are very similar. The online methods perform generally slightly worse than the offline variants due to the local knowledge restriction for model determination.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the online approach over all datasets.

D.8 Examplepath Evaluation

Although a single examples does not show an overall behavior, they allow a deeper understanding of the heading estimation and the related problem. We provide more statistical data on the presented trousers pocket example run presented in Section 6.2.8. Moreover, we give an examples at the jacket pocket and handheld device location, wheras both have been performed on the same path shown in Figure 6.12.

D.8.1 Additional Examples: Jacket Pocket & Handheld

The results of the example runs at the Jacket Pocket location are presented in Figure D.23, wheras the results for the handheld example run are shown in Figure D.24.

Moreover, we provide additional statistics for both example runs. Boxplots are shown in Figure D.25 and in Figure D.26, whereas we provide error distributins in Figure D.27 and in Figure D.28, respectively.

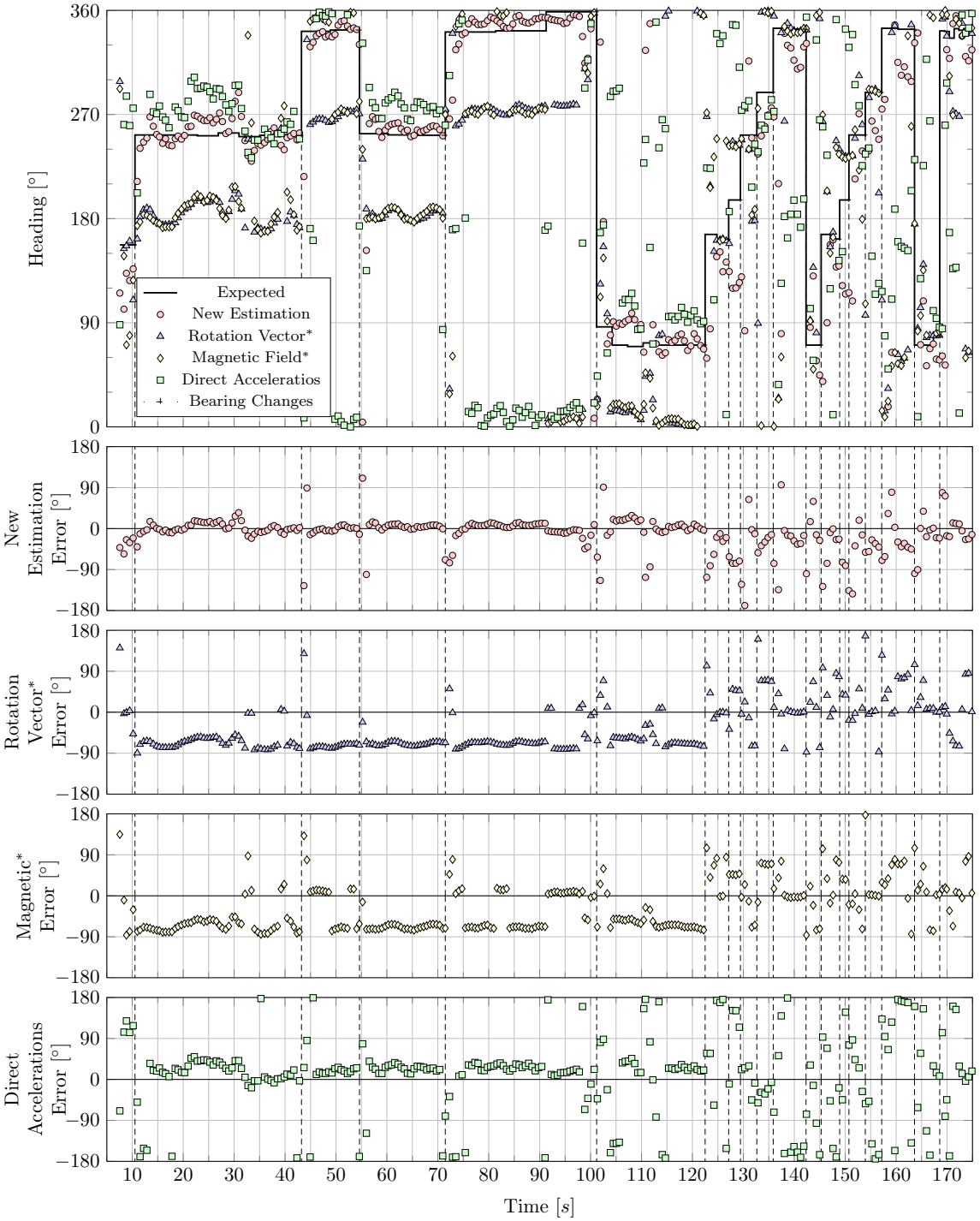

Figure D.23 Example Run indoors - Results of Jacket Pocket. This figure shows the results of our example run of the path depicted in Figure 6.12 with the smartphone in the (outer) jacket pocket. The x-axis describes the passed time of the run.

Top: The upper graph's y-axis describes the current heading in degree. We illustrate the expected heading values extracted from map data and estimated heading information of the regarded techniques. Furthermore, we marked significant heading changes by vertical dashed lines.

Center to Bottom: The other four subgraphs show an exact evaluation of the estimated headings, where the y-axis represents the error.

The New Estimated performs well at low error rates except for higher errors at each bearing change. Although we use flipping, Rotation Vector* and Magnetic* heading estimation show significantly wrong estimated values at mostly about 90°. Direct Accelerations provide a much better heading estimation, but is not as good as our New Estimation method.

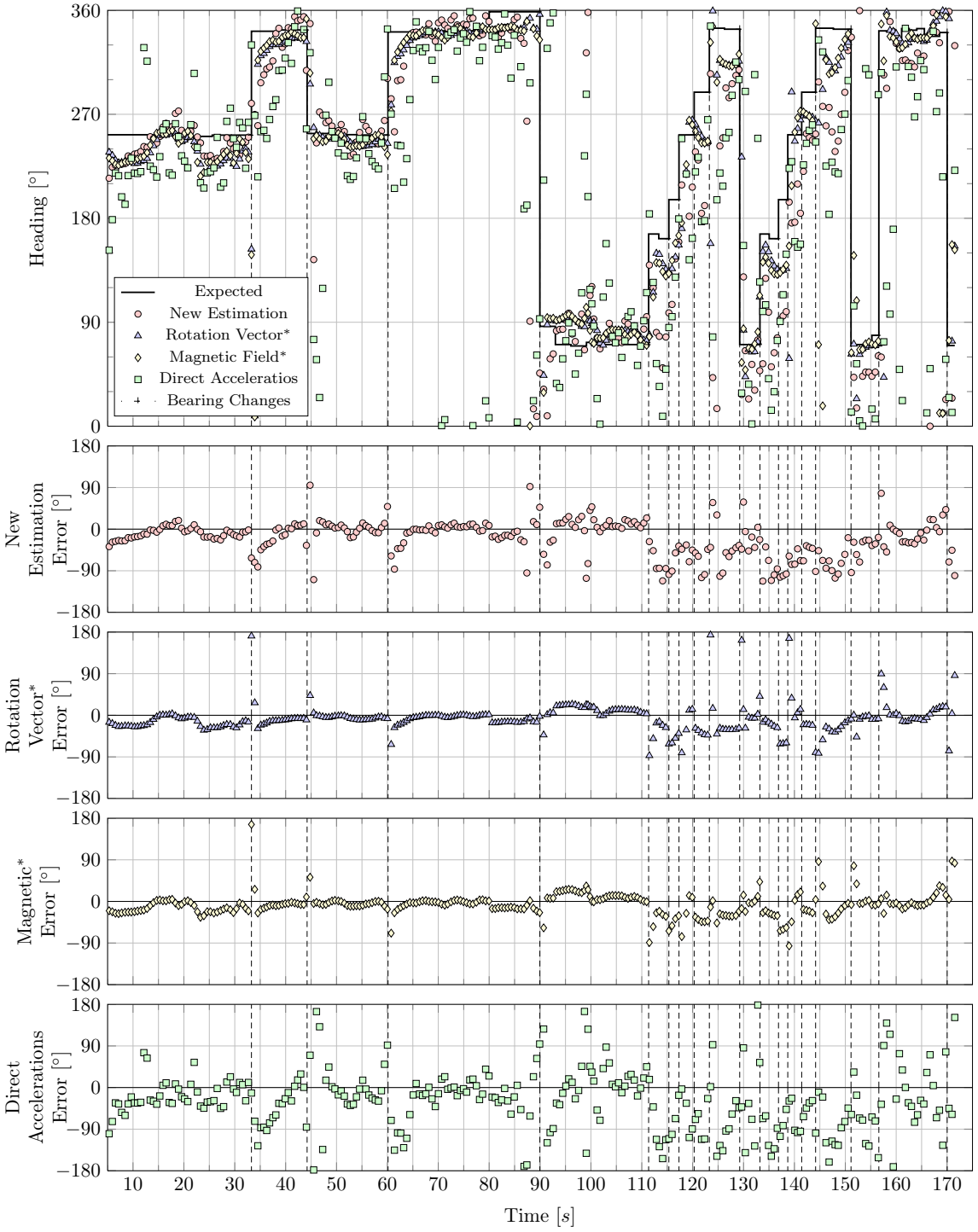

Figure D.24 Example Run indoors - Results of Handheld. This figure shows the results of our example run of the path depicted in Figure 6.12 with the smartphone in the handheld location. The x-axis describes the passed time of the run.
Top: The upper graph's y-axis describes the current heading in degree. We illustrate the expected heading values extracted from map data and estimated heading information of the regarded techniques. Furthermore, significant heading changes are marked by vertical dashed lines.
Center to Bottom: The other four subgraphs show an exact evaluation of the estimated headings, where the y-axis represents the error.
Both methods, the Rotation Vector as well as the Magnetic approach provide very accurate results and outperform our New Estimation technique which used the jacket model. The Direct Acceleration approach shows a strong variance and most errors.

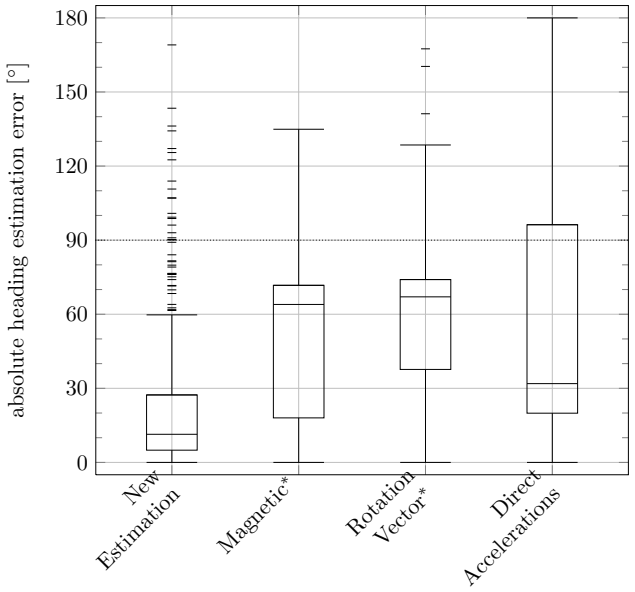

Figure D.25 Example Run Evaluation: Jacket. This boxplot shows the error distribution if the regarded headings estimation methods at a testrun in the jacket device location.
Our New Estimation approach provides the best results with $q_{.75} < 30°$. The Magnetic and Rotation Vector perform quite similar, whereas the Magnetic method shows better performance in the lower quantile. The Direct Accelerations show an accurate estimation for half of the estimated headings with $q_{.50} \approx 30°$, but shows a wider overall distribution than all other methods.

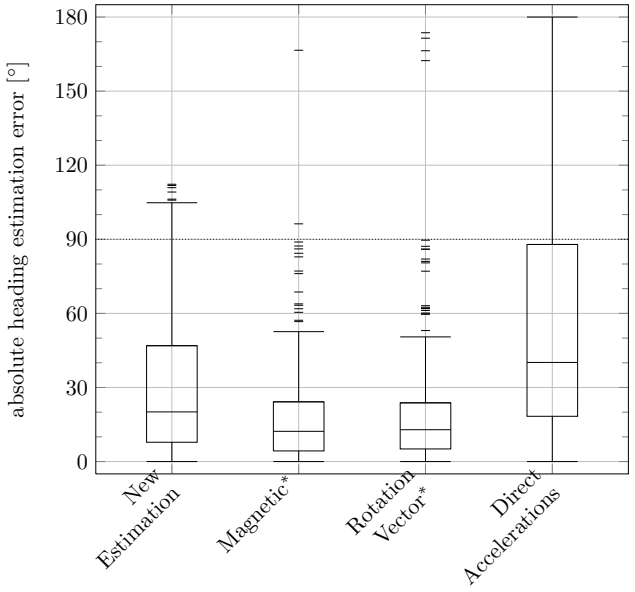

Figure D.26 Example Run Evaluation: Handheld. This boxplot shows the error distribution if the regarded headings estimation methods at a testrun in the handheld device location.
Both, the Rotation Vector and Magnetic approach perfrom very good with low error rates with $q_{.75} < 30°$ and up to four outliers beyond 90°. Our New Estimation method, which operates with the jacket model also performs acceptable with $q_{.75} < 50°$. The Direction Accelerations approach provides the worst results with $q_{.50} \approx 40°$ and $q_{.75} \lesssim 90°$.

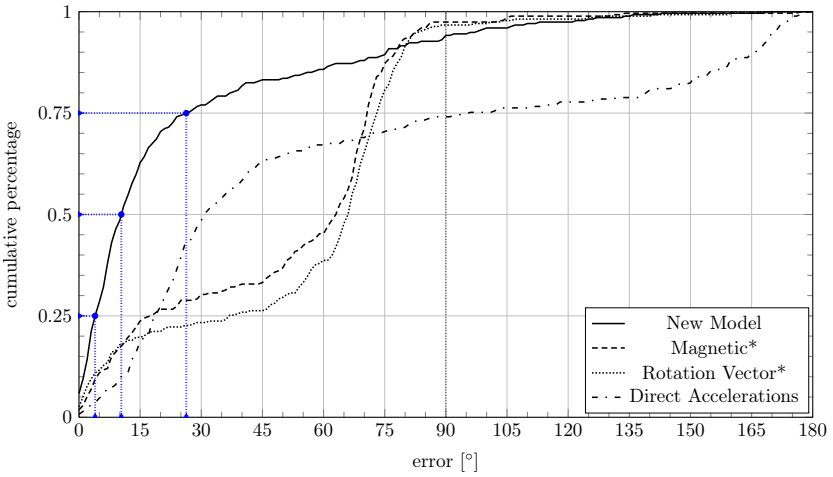

Figure D.27 Example Run Evaluation: Cumulative Error Distribution - Jacket. This figure shows the complete error distribution of the regarded heading estimation techniques at the example handheld run. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
Out New Estimation technique shows very good performance with $q_{.75} \approx 25°$. Both, the Magnetic and Rotation Vector technique fail to give accurate results, but seem to be biased as the largest amount of errors is in an interval of 45° to 75°. The Direction Accelerations provide accurate results for roughly 65% of the estimations, but shows a significant amount of errors greater than 90°.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the New Estimation technique.

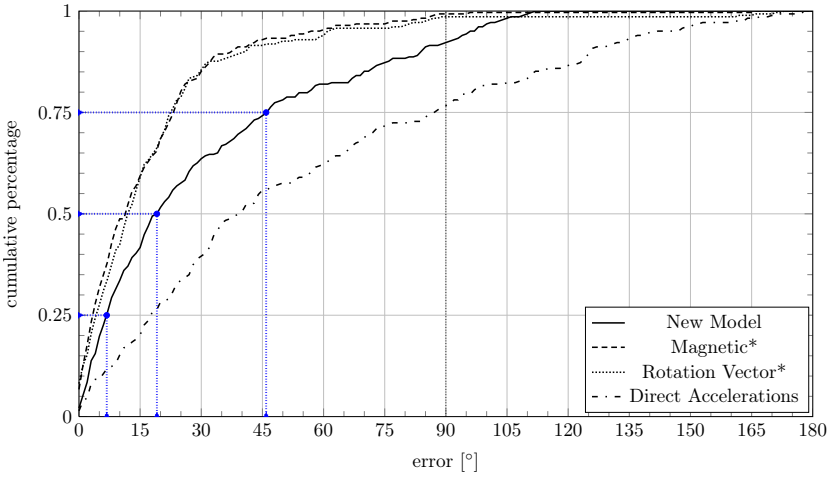

Figure D.28 Example Run Evaluation: Cumulative Error Distribution - Handheld. This figure shows the complete error distribution of the regarded heading estimation techniques at the example handheld run. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
Both, the Rotation Vector and Magnetic method yield very accurate results with $q_{.75} < 25°$ and $q_{.90} < 75°$. Our New Estimation also performs good with $q_{.75} \approx 45°$. The Direct Accelerations approach is not as accurate as the other methods.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the New Estimation technique.

D.8.2 Additional Statistics for Trousers

We have presented a test run at the trousers location in Section 6.2.8. We provide further statistical data in a boxplot in Figure D.29, wheras the corresponding error distributions is depicted in Figure D.30.

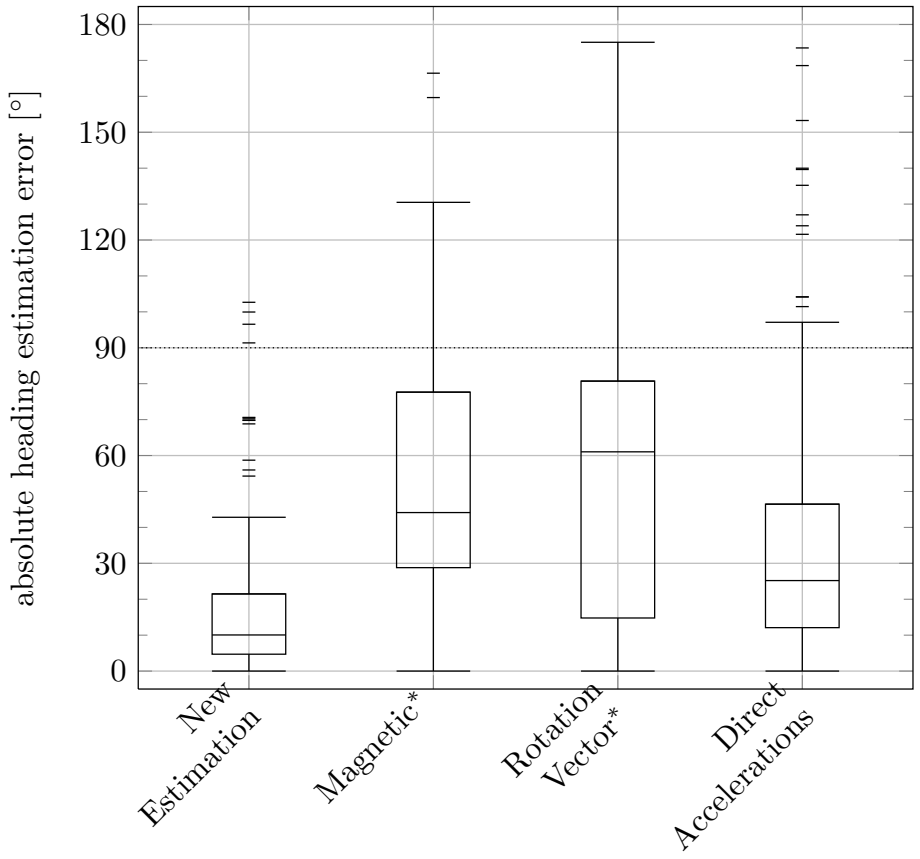

Figure D.29 Example Run Evaluation: Trousers. This boxplot shows the error distribution if the regarded headings estimation methods at a testrun in the trousers device location.
We get best results by our New Estimation approach with $q_{.75} \approx 20°$, but it also shows several outliers with errors up to 115°. The Direct Accelration technique outperforms the Magnetic and Rotation Vector method with $q_{.75} < 50°$.

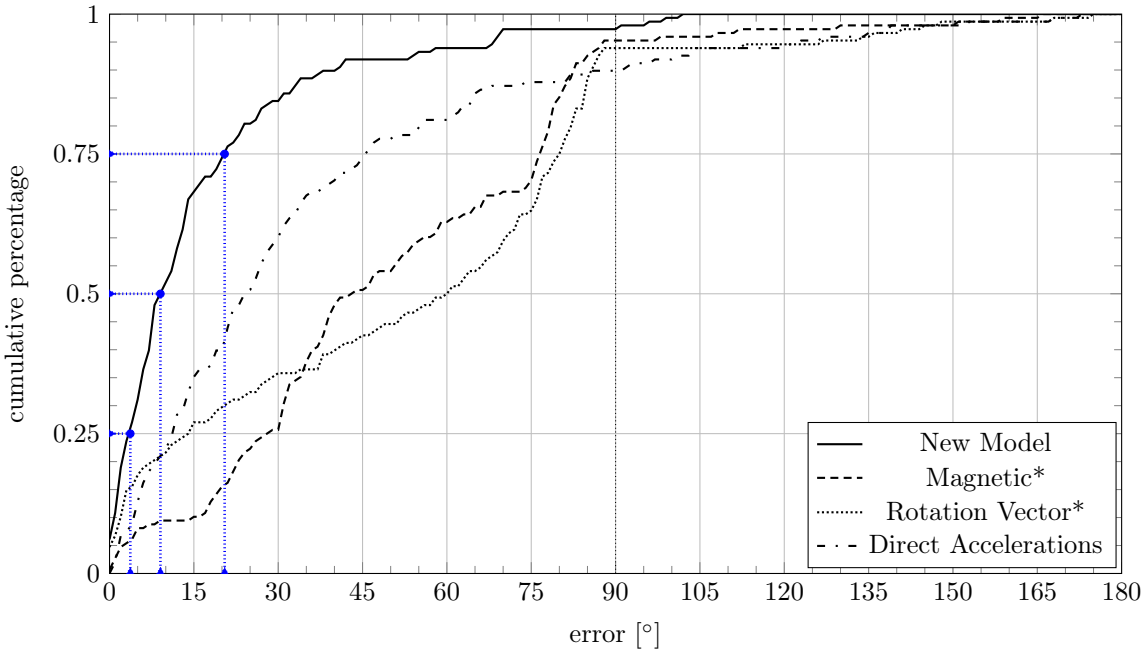

Figure D.30 Example Run Evaluation: Cumulative Error Distribution - Trousers. This figure shows the complete error distribution of the regarded heading estimation techniques at the example handheld run. The x-axis describes the absolute error, whereas the y-axis describes the relative amount of measures below the corresponding absolute error.
Our New Estimation technique shows very good performance with $q_{.75} \approx 20°$. The Direction Accelerations method also provides very accurate results with $q_{.75} \approx 45°$ and outperforms the Magnetic and Rotation Vector approach.
Note, that we have marked the quantiles of interest ($q_{.25}$, $q_{.50}$, $q_{.75}$) of the New Estimation technique.