Masterarbeit

Development of a Location-Based Information and Navigation System for Indoor and Outdoor Areas

---

zur Erlangung des akademischen Grades
Master of Science

vorgelegt dem
Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen

Nils Becker
im März 2014

Referent:	Prof. Dr. Michael Jäger
Korreferent:	Sebastian Süß

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Gießen, den 12. März 2014

Nils Becker

Abstract

Modern smart phones offer the computing power, connectivity and sensors needed to assist their users in daily life. Therefore, the development of the mobile campus information system (MoCaInfo) has been initiated in 2011, to provide a digital helping hand to students and university staff at a continuously growing and changing campus. Besides location-based information, the system offers point-to-point navigation in buildings and at the campus area. In contrast to other location-based information systems, MoCaInfo focuses on assisting visually impaired people, as well as sighted people.

In this thesis, the requirements for a location-based information and navigation system are analyzed, also considering the special requirements which arise due to the assistants of visually impaired users. This includes a low-vision user interface for the mobile application and particular navigation instructions.

Furthermore, a library for mobile indoor and outdoor navigation has been developed, since existing solutions barely support indoor navigation. For this, as much existing approaches as possible have been used and integrated into the system, to build upon established components. To these components belong Google Maps and OpenStreetMap with its toolchain. In order to assist visually impaired users as well, additional navigation information, e.g. about the condition of the floor, is offered.

With the purpose of providing a point-to-point navigation system, a positioning system is essential, too. It uses Wi-Fi signals for absolute positioning in buildings, in combination with relative positioning information gathered by analyzing data of the smart phone's accelerometer, magnetometer and gyroscope. Due to the conjunction of this positioning information, an average error of 1.67 meters has been achieved.

Contents

List of Figures

Part I.

Project Overview

1. Introduction

The following work describes the development of the mobile campus information system ("MoCaInfo") which has been developed at Technische Hochschule Mittelhessen - University of Applied Sciences. MoCaInfo's major goal is providing location-based information to students, lecturers and university staff. That information might be a floor plan, opening hours of the faculty office or the cafeteria menu.

In addition, MoCaInfo includes a point-to-point navigation system, especially to enable new and abroad students as well as visually impaired people to find their way at a continuously growing and changing campus.

MoCaInfo consists of a bunch of different software components which are divided into end-user front-ends, administrative front-ends and back-ends. At the current state of development, the end-user part has been realized as a mobile Android application and an HTML5 web application. The administrative tasks can either be fulfilled with the help of a web application or desktop application.

The software itself has been elaborated by students in different classes starting with "Project Mobile Campus Information System" in 2011, followed by two "Location-based Services" classes.

The following chapters will gain an insight into the different approaches which have been developed for the several problems. At this, the focus is on mobile user interfaces for visually impaired people, indoor and outdoor positioning and point-to-point navigation.

The generic masculine is used in the thesis, due to a better readability.

1.1. Aim of the Thesis

Todays smart phones offer decent CPU and memory resources as well as various sensors, enabling developers to implement location-based information systems. However, today's information systems have deficits when it comes to point-to-point navigation. Main reason for this is the lack of standard indoor positioning techniques. Furthermore, the needs of visually impaired people are barely considered in existing location-based information systems.

Therefore, the aim of this thesis is to provide possible solutions for the mentioned problems and share the experiences made during the development of a university location-based information system. One distinguishing feature of the proposed system is an indoor and outdoor point-to-point navigation system. The system has to be usable at current Android smart phones. In order to achieve such a navigation system, one of the problems to be solved is indoor positioning. At this, it is important to use sensors and connectivity technologies, which are available in today's smart phones.

Furthermore, a system for outdoor and indoor map creation, rendering and pathfinding is needed. Particular requirements for these components arise because visually impaired people shall be guided by the navigation system as well. This user group is often ignored by existing systems, even though they benefit even more from a navigation system than sighted users as they are not able to orientate with the help of usual signposts. For this reason, much additional information has to be gathered and considered by the components involved in navigation.

This thesis builds upon the previous insights of many students and co-workers who worked at the MoCaInfo project in the past years. Those insights are combined with own thoughts and other current research findings.

1.2. Structure of the Thesis

The document is divided into five major parts, which are briefly introduced in the following enumeration.

I. Project Overview gives a brief introduction into the MoCaInfo project, explaining the stakeholders and resulting requirements for the information and navigation system. Section 3 describes the components and data model, needed to realize the previously defined requirements. The part ends with an explanation of MoCaInfo's content model.

II. Mobile Application addresses the actual end user interface of MoCaInfo, the mobile Android application. The part begins with a brief introduction. Followed by an explanation of the mobile application's data synchronization feature, which enables users to use most features without a network connection. After that, the pros and cons of a user interface which follows the universal design approach and a dedicated low-vision user interface are discussed.

III. Navigation deals with the problem of indoor and outdoor navigation. At first existing map and navigation solutions are evaluated, in order to find out if and how they can be used in the given scenario. Section 10 explains the approach used for implementing an indoor and outdoor navigation system, using as many established technics and libraries as possible. The following sections outline additional information which have to be considered in order to provide a good navigation experience for visually impaired people. The navigation part ends with a description of the navigation component's user interface and its software architecture.

IV. Indoor and Outdoor Positioning gives an overview about existing positioning technologies and approaches. At this, it is distinguished between absolute positioning approaches, which are able to determine the absolute position on earth and relative positioning approaches, which are only able to detect changes in position. After this general explanation, the actual positioning system which has been developed for MoCaInfo is introduced. To these belongs a Wi-Fi fingerprinting approach, which is explained in section 17.5. Those Wi-Fi positions are reckoned with a relative positioning component, which is made of a step detection algorithm and a compass. Additionally, NFC and GPS information are used for position determination when available. The different algorithms and approaches are explained in section 18.

V. Conclusions and Future Work summarizes the insights about mobile user interfaces for visually impaired people as well as indoor and outdoor navigation and positioning. The thesis ends with thoughts about future work for further improvements, followed by some final words how indoor positioning and navigation may change within the next couple of years.

2. Requirements

The following section describes the stakeholders involved in *MoCaInfo*. Based on those stakeholder descriptions, functional and non-functional requirements for the system are defined.

The general purpose of *MoCaInfo* is to provide students, staff and guests of Technische Hochschule Mittelhessen with valuable, often location-based, information. To achieve this basic goal, client applications for end users as well as administrative applications for data acquisition need to be designed and implemented.

As the overall development of *MoCaInfo* uses an agile approach, there is no contract style list of requirements or anything similar. Instead user stories have been described verbally, using the classical approach: "*As a <role> I want to <feature/ability of the system> so that <value the role receives from the feature>*" [1, p. 102ff].

Before user stories and requirements can be defined, it is common practice to analyze the stakeholders who are involved in the software system[1, p. 119ff].

2.1. Stakeholders

At first glance there are two stakeholders, *end users* who use the system and *administrators* who maintain the information system and its data. Looking more deeply into these stakeholders one notices that a more detailed separation is necessary. *End users* can be seen as a group which contains more precisely defined stakeholders.

2.1.1. End users

students are the biggest end user group for a mobile campus information system. Because of their age, they can be seen as belonging to the so-called group of *digital natives*. Today's students are used to handle interactive web and smart phone applications to look up information or communicate with each other. Because of the daily usage of the internet and smart phone applications, students expect an interactive and responsive user interface, which presents the information in an appealing way.

visually impaired or even blind students and staff members are the second large user group, which is often disadvantaged by common information systems. Instead of seeing information, visually impaired users have to be able to request and understand information by using their remaining senses. Therefore, the focus of an application has to be on audible and tactile feedback.

university staff may also benefit from a centralized information system. In opposite to the students, it has to be assumed that many staff members are not pretty familiar with using mobile applications.

2.1.2. Administrators

An information system is only as good as its information. This information has to be created and maintained. Administrators are a privileged group of people which is allowed to edit or create new information. They do not necessarily have a technical background. Therefore, applications have to be as simple as possible and well documented.

2.2. Functional Requirements

Functional requirements define functions of the system or a system's component. The following functional requirements have been extracted from user stories which are not described in this thesis.

2.2.1. Information System

room information - the system has to provide information about rooms, whereas the kind of information depends on the room type. For example, a lecture schedule should be shown for lecture halls. Information about an office might list the staff members who usually work in the office and their contact information. Additionally, opening hours for the deanery and the cafeteria's menu are potentially useful room-depended information.

Since information is the data which is shown to the end-user, in this thesis the term is often used equivalent to the term *content*.

points of interest (POI) - beside rooms, there are other points of interest. Such POIs are, for example, copying machines, beverage dispensers or seating accommodations. A list of nearby POIs has to be available, based on the user's current position.

contact person - a pretty common situation for students but for university staff as well is to find the correct contact person for a particular problem. Students may need a formula for their health insurance or a staff member needs opening permissions for a certain room. Based on the given keyword, e.g. "opening permissions", the information system shall find the appropriate contact persons. The search result shows the according contact and office information of the found staff member.

non location-based content - most content in the system is location-based, whereas there is location-independent content, as well. Examples for location-independent contents are newsfeeds and general information about the university, study paths or semester breaks.

bookmark - the user has to be able to bookmark POIs and content for a fast future access.

user feedback - the user has to be able to enter comments or ratings for certain content or POI elements.

data import - the most important part of an information system is information. Therefore, the system needs to be able to import or directly access existing data such as lecture schedules and contact lists.

user interface for visually impaired users - the mobile application has to be usable by visually impaired or blind users. Often they use tools like software magnifiers to enlarge existing screen elements. Such tools are barely usable at Android devices. For this reason, it might make sense to implement a user interface which is customizable and scalable. Therefore it is necessary that the user is able to change font styles, size and colors to make the app fits to his individual limitations.

2.2.2. Navigation System

point-to-point navigation - the system has to provide the possibility to navigate from one point on the campus to another. The navigation targets are points of interest.

shortest path - the computed route has to be short and reasonable. This includes the usage of side entrances and other shortcuts.

audio-visual navigation instructions - navigation instructions, as known from car navigation systems, have to be determined and presented to the user in an audio-visual manner.

navigation instructions for visually impaired people - visually impaired and blind people need additional information about a walking route which have to be provided by the system. Such information are obstacles like a fire door that crosses the way, the floor's condition and others.

position determination - the user's position has to be estimated by the system, no matter if the user is outdoor or indoor. The accuracy has to be reasonable, in order to be able to provide point-to-point navigation.

2.3. Non-Functional Requirements

offline availability - as many data as possible should be offline available because there are some parts on the campus where neither a reliable Wi-Fi nor a mobile connection can be established. Regarding that lots of data

like room information and contact persons are valid for certain months or even years, further approves the reasonableness.

low resource consumption - the mobile application has to be able to run on lower midrange devices without limitations. Therefore, using efficient algorithms and technologies to achieve the intended features is necessary. Furthermore, old Android versions, which are still wide-spread, have to be supported.

maintainability - is important for any software project, but in this case it is particularly important because many students only work at the project for several months. Therefore, it should be easy for students to make themselves familiar with the project and its source code. To achieve this, any source code has to be written in Java because it is the first learned programming language at Technische Hochschule Mittelhessen.

intuitive user interface - the user interface has to follow well-known user interface design patterns in order to make the usage as intuitive as possible.

3. Architecture

The requirements listed and explained in chapter 2 influence the software architecture. Therefore, the first step of creating the architecture is to describe logical components, needed to fulfill those requirements. Afterwards, the single components are transferred into a logical architecture. The chapter ends with an introduction to the system's data model.

3.1. Component Overview

The term component does not define how it is actually implemented. In case of MoCaInfo, some components are realized as standalone applications others have a programming library behavior.

Content Management System enables privileged users to create, edit or dismiss content which is available within the client applications.

Database stores data which is needed for the information system and components like *positioning* and *navigation*. The data itself is inserted by the *content management system* and the *data import* component.

Data Import uses existing data sources and stores them into the *database*. To achieve this, the existing data has to be transformed into a specific data structure.

Data Synchronization synchronizes data from the global database to a locally available database. This is needed because the smart phone application has to be as network independent as possible.

Map Editor enables privileged users to create indoor maps of buildings. Those maps include meta-information about buildings and its levels as well as

information needed for indoor navigation. Furthermore, the map editor supports gathering of outdoor areas.

Positioning is needed to determine the user's position, no matter whether the user is inside or outside a building. This component is used by the *smart phone application* and the *navigation* component.

Navigation inside buildings and on outdoor facilities is provided by this component. It consists of methods for calculating routes, navigation instructions and possibilities to visualize navigation-dependent data like maps.

Smart Phone Application shows information stored in the *database* and enables users to navigate to points of interest.

Web Application shows information stored in the *database* and enables users to show a route to points of interest.

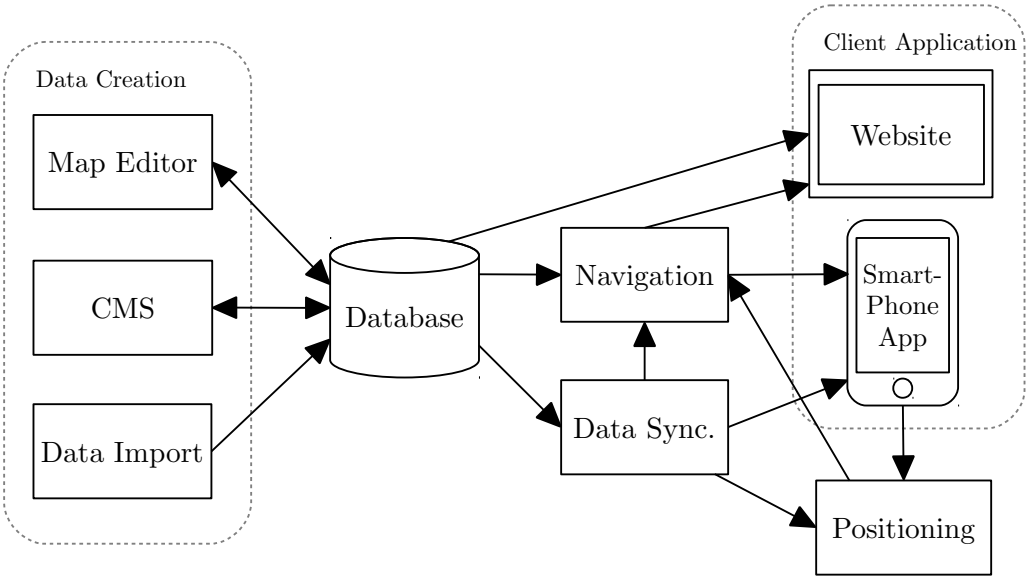


Figure 3.1.: Logical Component Overview

Figure 3.1 shows the connections between the different components described above. Each arrow illustrates a communication between the connected components, whereas the arrowhead points out the communication direction. The *database* acts as an intermediator between the components. The only component which does not directly access the database is the *smart phone application*. It accesses all necessary data via the *data synchronization* component. Due to

that approach, the smart phone application does not depend on an ongoing connection to the *database*.

A more detailed description of the single components can be found in the following chapters.

3.2. Logical Architecture

A common approach for structuring an information system's architecture is the usage of a multilayered logical architecture. It is a pretty straight forward way, to make sure that business logic and user interfaces are separated. This is important in order to keep a growing system understandable and maintainable. The layers used for *MoCaInfo* are based on a best practice for domain-driven design, which is described among others by Eric Evans[2, p. 68ff].

Evans recommends the following four layers:

Presentation Layer is responsible for presenting information to external actors and interpreting the actor's commands. External actors are usually human end users, but in some cases, the external actor is another computer system.

Application Layer (*Service Layer*)[1] provides services which establish a set of operations and coordinates the application's response in each operation. The layer itself is kept thin. The actual business logic is implemented in lower layers. Therefore, the *application layer* only defines an application's boundary and its set of available operations from the perspective of interfacing client layers.

Business Layer (*Domain Layer* or *Model Layer*) is the heart of any software because it contains the business logic. However, technical details, like storing data are delegated to the lower *infrastructure layer*.

Infrastructure Layer provides general capabilities to enable the higher layers fulfilling their tasks. A common feature provided by the *infrastructure layer* is the storage of data.

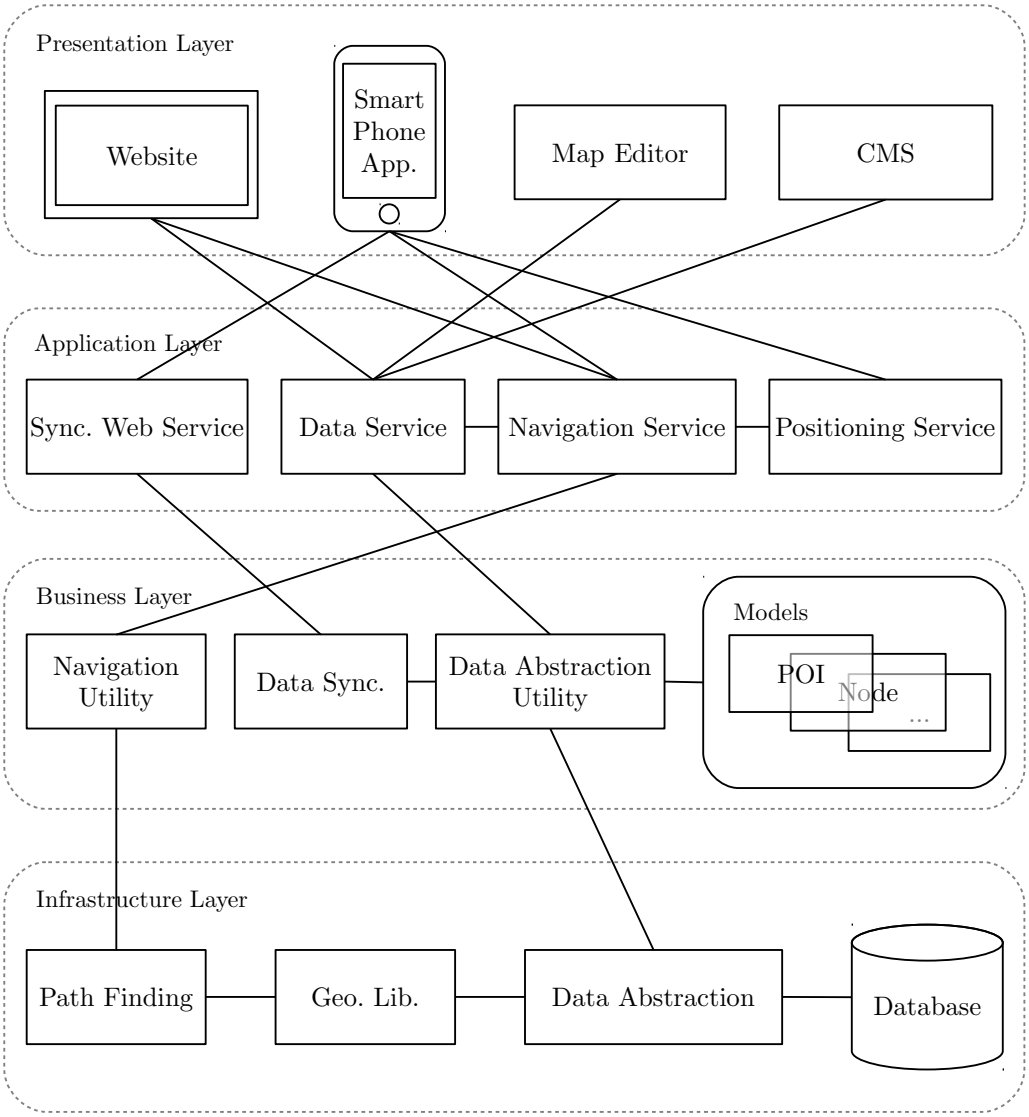


Figure 3.2.: Layers Architecture

To achieve a layered software architecture, the components introduced in section 3.1 have to be assigned to one of these layers. Figure 3.2 shows the components already explained components, in addition to some new ones. The lines between components illustrate a usage relationship, e.g. the *smart phone application* uses the *navigation service*.

[1] see [3, p. 138]

The *presentation layer* holds all applications which are visible to end users: the *website*, the *smart phone application*, the *map editor* and the *content management system*.

The *application layer's* components act as interfaces between the actual business logic provided by components in the *business layer* and components from the *presentation layer*. The technical realization of these interfaces slightly differ from component to component. In case of the *synchronization web service* which is the interface for the incremental data synchronization, operations are provided via a RESTful web API. Other components like *data service* just abstract direct access to other components from a developer's point of view. *Navigation service* is used by the *smart phone application* and the *website* to compute and visualize routes and other navigation-related information. *Positioning service* is used by the *smart phone application* to determine the user's current position.

The *business layer's* components implement the business logic. Together with other components from the *infrastructure layer*, the *navigation utility* finds the shortest path between two POIs and offers other navigation related operations to the upper layer. Another important part of the *business layer* is the data model, which is represented by a couple of components on the right side of the figure. It should be noted that figure 3.2 only shows the model components *POI* and *node*, but the complete architecture contains a lot of additional models, e.g. to represent *content*, *Wi-Fi fingerprints* and many more. The *data synchronization* component implements methods which are needed for an incremental data synchronization. The *data abstraction utility* provides functions to load and modify model components.

The lowest layer offers pretty general, low level operations, like a *pathfinding* algorithm or tools to convert between different geographical location formats. The *data abstraction* component implements the actual database access.

3.2.1. Architectural Pattern: Layers

For a better understanding of the architecture described above, this section summarizes and explains the general properties of the *layers* pattern. The description is based on Buschmann et al.[4, p. 31ff].

The probably best-known example of a layered architecture is the ISO/OSI 7-Layer model. The layered approach is considered to be better practice than implementing a protocol as a monolithic block because of several benefits. First of all, development tasks in a layered architecture can be distributed to different developers or teams more easily because the different parts of different layers are more or less independent, if the pattern is used pervasive. Having semi-independent parts, also enables an easier exchange of individual components at a later date because the section of code, which has to be replaced, is clearly delimited.

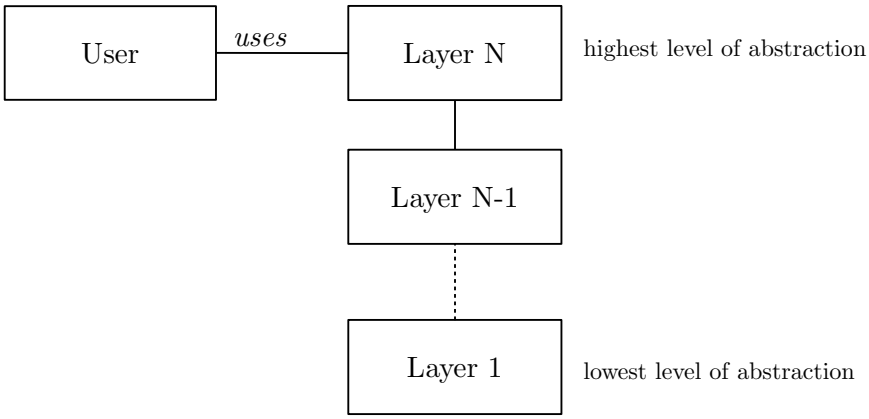


Figure 3.3.: Architectural Pattern: Layers

A typical system, which is predestined for a *layers* architecture, has different levels of abstraction. The uppermost layer starts with a pretty high-level of abstraction, which usually acts as an interface to the user. The level of abstraction decreases at lower layers. The higher levels rely on operations, provided by the lower level. As shown in figure 3.3 it is an essential principle of the pattern that any element of a layer depends only on elements in the same layer or on elements of the layer below.

A single layer does not necessarily have to be composed of one single component. Often a layer is a complex entity, consisting of different components which may even communicate with each other. To keep the advantage of a loose coupling, it is important to abstract the communication between components at different layers by a layer interface.

3.3. Data Model

The following section gives an overview on the system's data model. As it is used across all components, it can be seen as the lowest common factor of the different components.

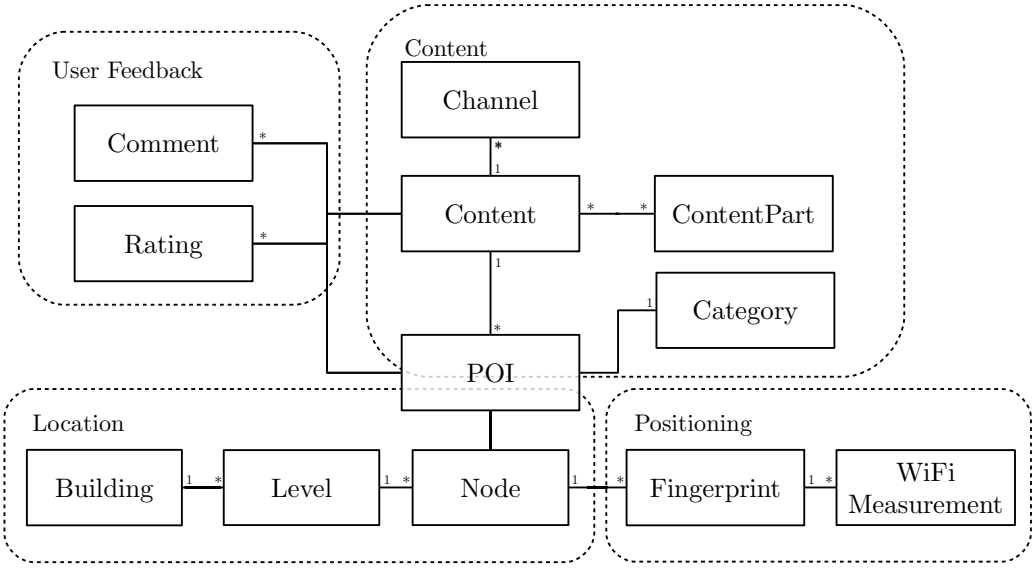


Figure 3.4.: Entity-Relationship Diagram of MoCaInfo's Data Model

Figure 3.4 visualizes the data model's main parts. For reasons of clarity, the entity-relationship diagram does not show any attributes or named relationships. The actual implementation of the data model is realized by a MySQL database on the server side and an SQLite database on the mobile device, which are kept in sync.

The different tables can be classed into four different logical categories. The first one is *user feedback* which holds *comments* and *ratings* about *content* and *POIs*.

Another logical category represents *content*, which also includes an identically named table. *Content* may be assigned to so-called *channels*. Each *channel* represents a collection of content about a certain topic or addresses a certain interest group. For example, there is a channel for all computer science students which contains recent events about lecture cancellation and guest lectures. The detailed architecture and meaning of *content* is explained in chapter 4. A *POI* is represented by a describing *content* and a *node* which holds the geographic location. Therefore, the *POI* can be seen as a connector between the categories *content* and *location*.

The actual *location* of a *node* is described by its geographical location and the building plus level, whereas outdoor nodes are member of a pseudo building.

The positioning system uses a Wi-Fi fingerprint approach, details described in section 17.5. The according data is stored in the database tables *fingerprint* and *Wi-Fi measurement*.

4. Content Model and Managment

One of the most important aspect of an information system is the information which it provides. In case of MoCaInfo, any information is considered to be content. This chapter describes the content's data model, followed by a short introduction of the content management system.

MoCaInfo's content model, on the one hand, has to be flexible in order to describe different kinds of information, but on the other hand it has to prescribe a structure. This structure is needed, to visualize content in similar ways, no matter whether it is a room schedule, the cafeteria menu or something else.

Furthermore, content has to be reusable. For example, a professor's digital business card has to be shown as part of his office's room information and in a list which shows all lecturers of the faculty.

The following section describes the concept of *content parts* followed by an introduction to the content management system.

4.1. Content Parts

In order to achieve the goals of flexibility, structure and reusability a data model has been designed, which defines *content* as a set of *content parts*. At this, *content* is the object which is assigned to a POI or accessible via information channels, and *content part* is the object which holds the actual information.

A content part consists of a header, an optional rich content and textual content. The textual content can either be plain text or use HTML markups for basic formatting. Rich content is defined by an URI and usually refers to an HTTP resource, e.g. an image or a video. Furthermore, the rich content can

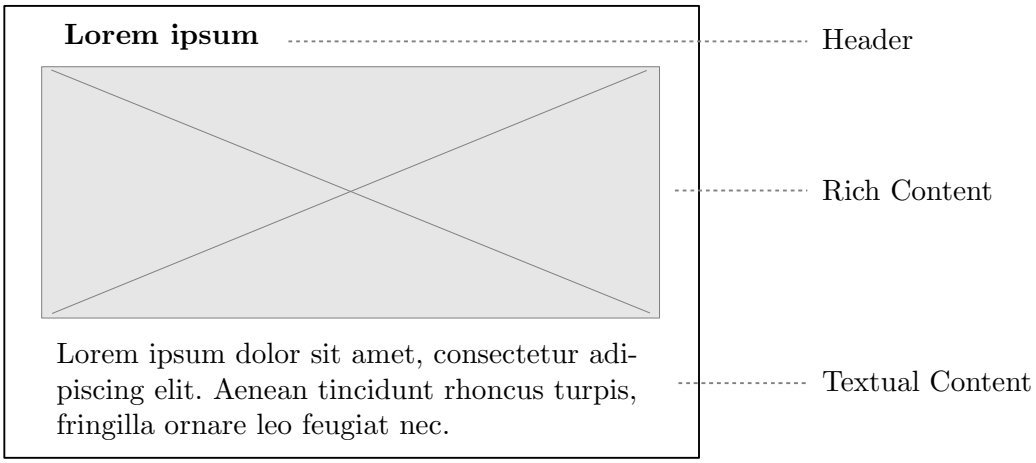


Figure 4.1.: Sketch of a Content Part, Using the *Big Picture Template*

point to internal resources, e.g. to show a floor map with the currently selected POI highlighted.

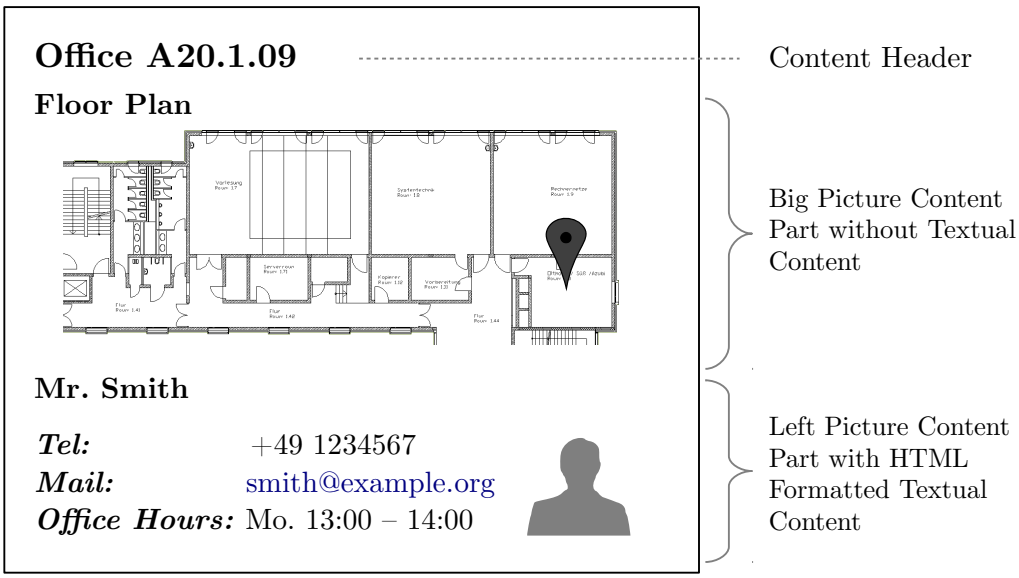


Figure 4.2.: Example Content Composed of Two Content Parts

The arrangement of the rich content element and its size is defined by a template. Currently there are three templates. One template showing a screen width filling rich content. It is called the *big picture template*. Two other templates only have a small square for rich content, either left or right beside the textual content.

Besides flexibility and structuring, content parts contribute to offline availability. As headers and textual content are stored in the database, it is offline available on the mobile device because of the data synchronization mechanism described in section 6. Only external rich content elements, such as images, are loaded on demand and may require a network connection. But even those rich content elements are cached, which makes them offline available when they have been loaded once before.

An example of a content object consisting of two different content parts is illustrated by figure 4.2.

4.2. Content Management System

In order to create, edit and delete content, a content management system (CMS) has been developed. The CMS is web-based so that content administrators only need a web browser to access the CMS.

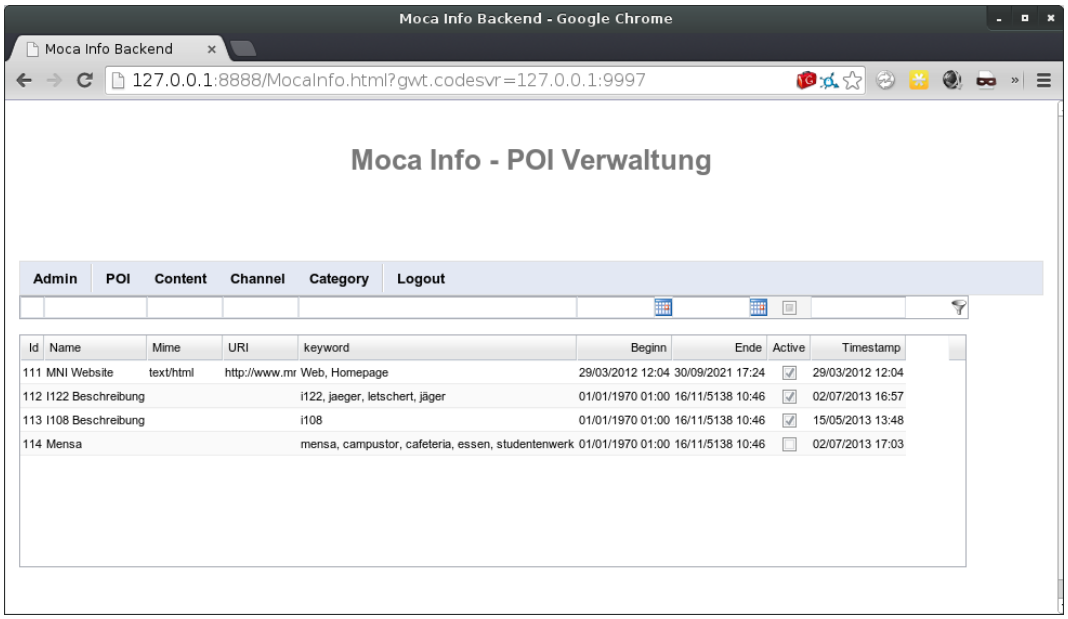


Figure 4.3.: Screenshot of the Content Management System

At the current state of development, the content management system is not much more than a database editor, which enables privileged users to insert or

update elements of the *content_part* and *content* database tables. In a future version, the CMS will assist the creation of certain content types, in order to make sure that similar content is gathered in the same way. Candidates for unified content types are for example, room information, digital business cards and general POI information.

4.2.1. Google Web Toolkit

The web-based CMS has to be written in Java, since one of the non-functional requirements imposes that all MoCaInfo projects have to be implemented with this programming language. Therefore, the Google Web Toolkit (GWT) has been used for development. GWT allows web application developers, to write plain Java sourcecode without taking care of web technologies. The GWT compiler creates HTML, CSS and JavaScript files for the front-end and JavaEE servlets for server-sided application parts. Typical candidates for such servlets are components which interact with server resources, such as the database.

The communication between the client's webinterface and servlets is asynchronous, using the AJAX technology. At sourcecode level, those asynchronous method calls are achieved by implementing a `RemoteServiceServlet` and a `RemoteService`. The client code calls an asynchronous interface `RemoteService` which in turn delivers data by triggering a callback method. In figure 4.4, this class is `DBServiceAsync`. Interesting from a developers point of view is the fact that the asynchronous interface does not need an implementation at all. The implementation is done by the GWT framework automatically. Figure 4.4 shows the Java classes involved in client-server communication, in a simplified UML diagram.

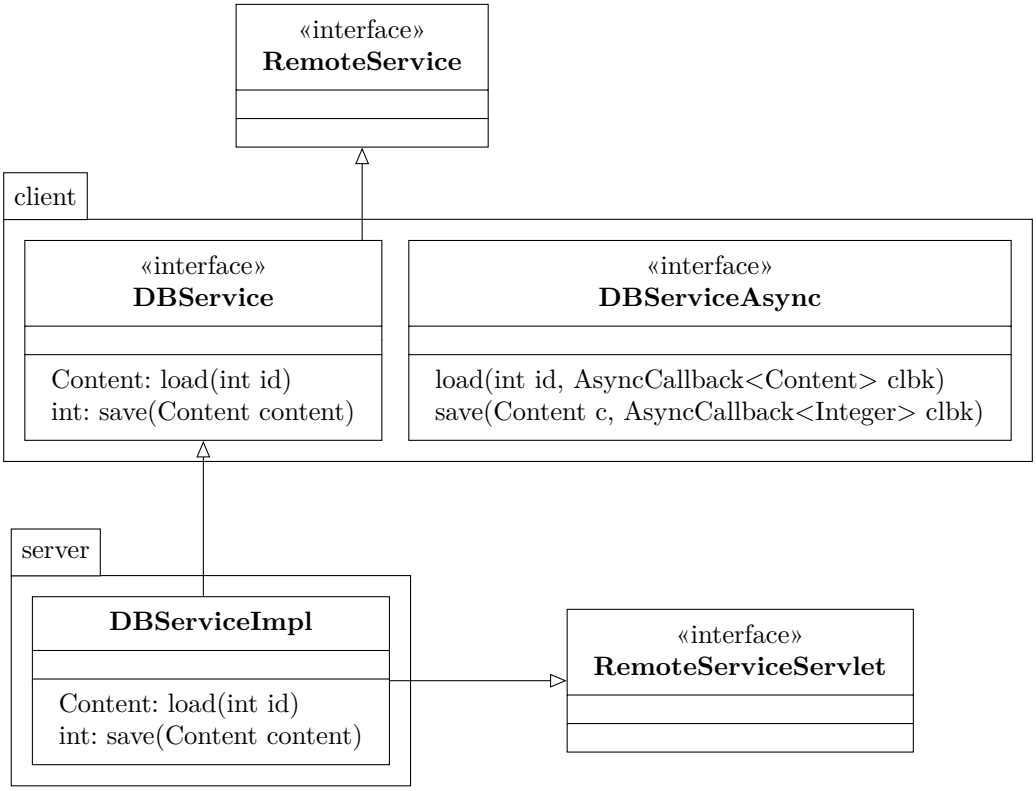


Figure 4.4.: Asynchronous Client Server Communication with GWT

Part II.

Mobile Application

5. Introduction to the Mobile Application

Two applications exist which provide the system's information to the users - a website and a mobile application. The website is nice to get a brief overview about certain information, but the mobile application is able to present the location-based information where it is needed. Furthermore, some of the proposed features, such as point-to-point navigation, cannot be achieved with a website.

Android is the targeted platform of the mobile application for various reasons. First of all, Android is the mobile operating system with the largest market share[5]. Second of all, Android devices offer all sensors and connectivity technologies which are needed to fulfill the requirements explained in section 2. In addition, Android applications are written in Java, and there are lectures at the university addressing Android development. Thus, there are students with the precognition needed to develop such an application from scratch.

An alternative, which has been considered, is a hybrid cross-platform approach. Such cross-platform applications are implemented using HTML5, JavaScript and CSS. By means of special frameworks, it is even possible to access native APIs, e.g. to be able to use the device's NFC reader. But those web-based solutions lack of techniques to support visually impaired end users. Furthermore, web and hybrid applications offer a worse responsiveness and user experience compared to native apps[6].

The application's first prototype has been developed by Artur Klos and Patrick Winter, former students of Technische Hochschule Mittelhessen, during their conclusion phase in 2011. Patrick Winter describes the development in detail in his bachelor thesis *"Entwicklung eines mobilen Campusinformationssystems"*[2][7]. This chapter summarizes his thoughts, in addition to the further

[2]german for: *"development of a mobile campus information system"*

development since August 2011. The second major part of the mobile application, beside the information system is navigation and localization, which is described in chapter IV starting at page 70.

The following sections use an Android-specific terminology. Those terms are described in the glossary at the end of this thesisDue to a better flow of reading.

6. Incremental Data Synchronization

One of the mobile application's key features is the ability to provide as much information as possible without having an active network connection. This is important because Wi-Fi and cellular network connections cannot be assumed to be available all the time. For this reason, the server's database has to be synchronized with the Android client's database whenever a network connection is available. This chapter describes the incremental data synchronization, including some issues which appeared during the development.

The data described in section 3.3 is stored globally in a MySQL database at a server. In order to be able to use one data abstraction library for server-sided and client-sided applications, the same database structure is needed on the server and the Android client. Since Android does not support MySQL databases, the mobile application uses a SQLite database which is perfectly supported by the Android framework.

Due to the two different database systems a MySQL database dump has to be converted into a SQLite database. This database dump has to retain the database structure as well as the data sets.

6.1. Database Dump

A common approach, described and discussed in many newsgroups is the shell script *mysql2sqlite.sh*[3], created and published by a user called *esperlu*. The shell script produces a MySQL database dump and uses a lot of regular expressions to map the MySQL syntax to an SQLite compatible one. The resulting SQL statements are used to create the corresponding SQLite database with the help of SQLite command line utilities.

[3] `https://gist.github.com/esperlu/943776`

This approach worked well in the MoCaInfo environment at an early state of development, only having some non-representative sample contents in the database. With a growing amount of data, errors occurred either when creating the initial MySQL database dump or when creating the SQLite database from the modified dump. Reasons for these errors are some special characters and quotation marks inside the MySQL database's content tables.

Improving the regular expressions which modify the MySQL dump is one possibility to fix this issue. But, this approach further increases the complexity of the already pretty confusing and hardly maintainable regular expressions.

Therefore, an approach has been developed which does not need to change the datasets with the help of regular expressions. Thus, the database's data is separated from the database's structure. The structure is created by *mysql2sqlite.sh* as described above, but the MySQL dump is used for modification, only includes the database's structure without any data. This results in an empty SQLite database with the same database structure. The data is transferred by a tiny application which connects to the MySQL database server and the SQLite database. It selects all data from the MySQL database and inserts it to the empty SQLite database.

6.1.1. Performance Issues

Selecting the data from MySQL and inserting it row by row results in a critical performance issue with SQLite. Within a test environment, it took more than 10 minutes to copy 1500 rows from MySQL to SQLite. This issue is caused by SQLite's transaction management. SQLite is designed to wrap any SQL insert into one atomic transaction[8]. As a result, inserting 1500 rows generate 1500 database transactions. By default, a SQLite transaction waits until the data is safely stored on the hard disk's surface before it completes.

According to SQLite's FAQ[8] the maximum number of transactions per second is about 60, at current 7200 rpm hard drives. In MoCaInfo's test environment, the number of transactions per second is even limited to 2.5.

To get rid of this problem, multiple inserts have to be combined into one transaction. Figure 6.1 shows the tremendous performance improvement. An

additional test with 50.000 database rows shows that the difference in performance between 5000 and 50000 inserts per transaction is just about 12%. Therefore, the nightly MoCaInfo database dump is created with 5000 inserts per transaction in order to keep the memory consumption at a low level combined with a reasonable runtime.

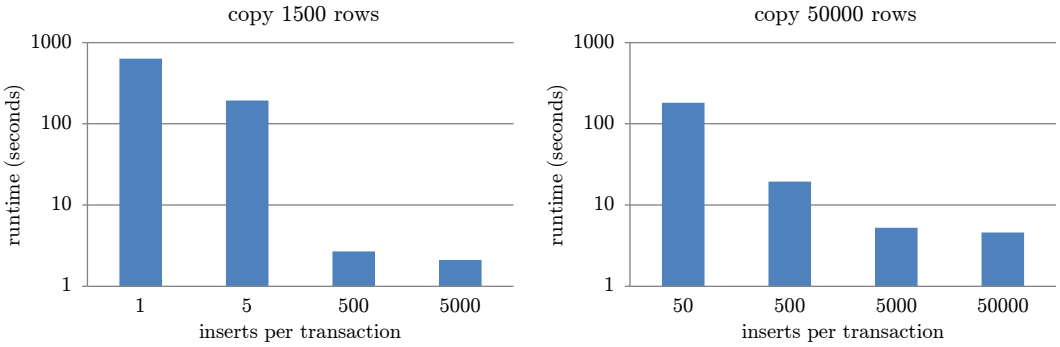


Figure 6.1.: Performance Overview, Inserting 1500 and 50000 Rows Into SQLite Database

6.2. Incremental Synchronization

Generating an SQLite database from the server's MySQL database is only one part of data synchronization. As mentioned earlier, the data synchronization has to be incremental. This prevents the mobile application from downloading a complete database dump of several megabytes just because a few data sets have been changed or added.

To implement this feature, the database needs to remember whether data has been added, changed or deleted. Therefore, any database entry has a *timestamp* field which stores the creation or modification time. The *timestamp* is stored as a UNIX timestamp which represents the time as the number of seconds elapsed since January 1st, 1970 00:00:00 UTC.

The *timestamp* field enables the system to figure out whether data is new or changed, but this cannot be applied to deleted entries. In order to keep track of deleted database entries there is a dedicated database table called *deleted_entry*. Whenever an entry is deleted from any other table, an entry

including the table's name and the entry's id is added to the *deleted_entry* table. Like any other table, *deleted_entry* has a *timestamp* field, to store the deletion time.

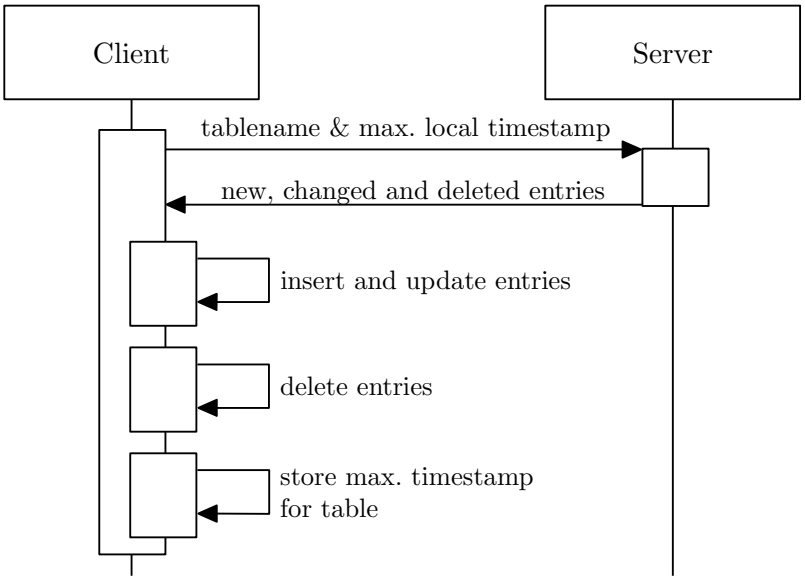


Figure 6.2.: Data Synchronization Between Client and Server

As shown in figure 6.2 the Android client sends a request to the server, including the database table's name and the maximum timestamp of the table's local copy. With this information the server is capable to collect all new, changed and deleted entries by searching all entries, which are newer than the local timestamp. The changed and deleted entries are send back to the client. The received information is used by the client to update the local database. After that process, the requested database table on client and server are in sync.

6.2.1. GSON

In a former development version of MoCaInfo, XML was used to send the information about changed entries to the client application. Therefore, the server loaded the data from the MySQL database and built a well formed XML document, by concatenating strings. At the client side, a SAX parser interpreted the XML string and wrote the data to the local SQLite database.

Both, the Android application and the web service which creates the XML string, are written in Java.

This XML solution has some problems, especially regarding the maintainability. Whenever the server's XML structure changes, the client's SAX parser needs to be adjusted accordingly. The problem is that developers will not get any compile-time errors even if the SAX Parser and the XML generator are incompatible. As a result, incompatibilities and resulting errors only appear at runtime. A general programming paradigm is to catch errors as early as possible because earlier errors are easier to handle. Therefore, compile-time errors are preferred to run-time errors[9].

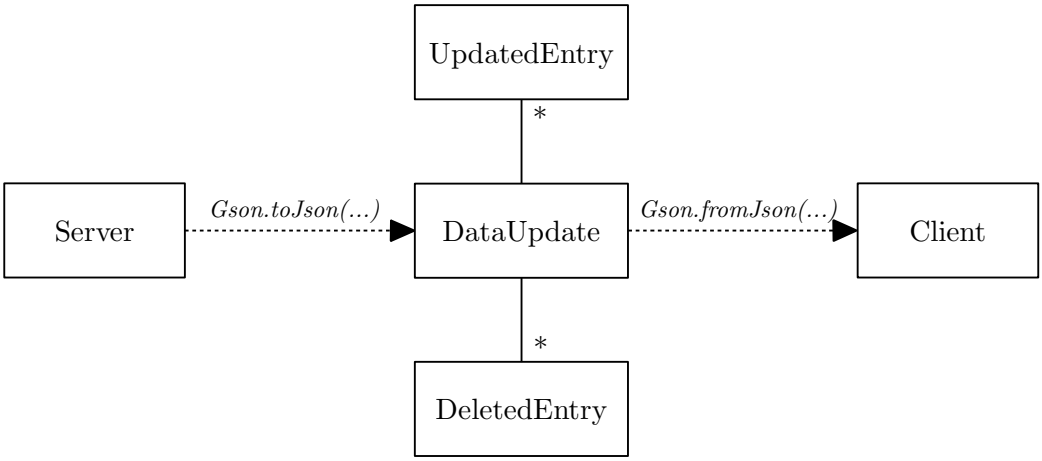


Figure 6.3.: Data Synchronization using GSON

A solution which shows compile-time errors needs to share Java classes instead of an abstract XML string. For this purpose, Google developed a library called google-json[4], in short GSON. It is an open source project which converts Java objects to JSON strings and vice-versa. Instead of creating an XML string manually at the server side and parsing it manually at the client side, a GSON-based solution uses the same Java classes at client and server. The actual data transport format is a JSON encoded string. The whole encoding and decoding is done by GSON, so there is no additional work for the application developer. Figure 6.3 shows the concept of using the same Java classes at server and client without the need to develop an encoder or a parser.

[4] `https://code.google.com/p/google-gson/`

7. User Interface

This chapter describes the mobile application's user interface. Two different approaches for user interfaces for visually impaired people are discussed. Afterwards, the graphical user interface is introduced followed by a description of a textual user interface. The chapter ends with an evaluation of the chosen user interface approach.

The user interface of an application is the system by which human users interact with the machine. A user interface has to provide the application's features in a feasible and appealing way. In case of MoCaInfo's mobile application, there are few additional things to consider, regarding the accessibility, which is ignored by most mobile applications.

7.1. User Interfaces for Visually Impaired People

As mentioned in section 2 visually impaired or blind people have totally different requirements for a user interface compared to sighted people. Sighted users tend to like fancy, colored user interfaces with fluid animations and transitions. For a better look and feel, buttons are often replaced by nice looking icons without any textual description. Those kinds of user interfaces, however are not at all appropriate for visually impaired users.

In fact, there are two different approaches to design user interfaces for visually impaired. The first possibility is to consider Android's accessibility features when designing the graphical user interface for sighted people. Blind people then should be able to use the usual graphical interface with the help of gestures and screen readers[10]. This concept is also known as *Universal Design*[11]. As a second approach an application can provide two different user interfaces, whereas one uses so-called *low-vision* controls instead of graphical ones.

7.1.1. Low-Vision User Interface

According to Sierra and Togores[12] the best way to afford a good user experience for visually impaired and blind people is to design the app directly for this user group. They say that the result usually is much better then designing an application for sighted people and try to make it usable for visually impaired by using accessibility tools like screen readers afterwards.

A usable accessible design uses *low-vision* controls. Those controls are designed to be larger than usual controls for sighted users and tend to use a pretty high contrast, whereas white on black is the best contrast for most visual impairments. The different controls and lists have to be usable through gestures. Furthermore, haptic feedback and text-to-speech should be used whenever it is possible. Besides that, low-vision controls should be customizable in size and color to fit best to the users' capabilities and disabilities.

These statements match with those of Niehaus[13], a former student of Technische Hochschule Mittelhessen, who is visually impaired himself. Beside the idea of large, high contrast controls, gestures and text-to-speech, he also states some possible incompatibilities which might occur between the low-vision user interface and other installed accessibility utilities. A common example he mentions is a conflict between a screen reader and an app which does self-voicing. One the on hand, self-voicing applications assist users by using text-to-speech, a classical feature of a low-vision user interface. On the other hand, screen readers try to read out the currently selected UI element. In many cases this means that a text is read twice, which is a really disturbing user experience.

7.1.2. Universal Design

Universal Design is a term which has been coined by Ronald L. Mace an architect and product designer. In one of his last works, he explains his ideas together with Molly Follette Story and James Mueller[11]. The concept of universal design describes a design which addresses each human, no matter what age, size, abilities, talents and preferences he or she has. To make product designers understand the various human abilities, he grouped them into cognition, vision, hearing and speech, body function, arm function, hand function

and mobility. By considering the different capabilities and needs, a product should be easily usable by all potential users, without the need to design a distinguished product for each user group.

7.1.2.1. Android Design Patterns: Accessibility

The Android design patterns by Google were created in accordance with universal design principles[10]. Together with the following Android accessibility features and tools, applications which are appealing for sighted users should be usable by visually impaired users, too.

TalkBack is a pre-installed screen reader. It uses spoken feedback to describe the results of actions, such as launching an app or events such as notifications.

Explore by Touch is a system feature that works together with TalkBack. It allows users to touch the device's screen and hear what is under their finger via spoken feedback.

Accessibility Settings allows modification to display and sound options, such as increasing text size or changing the speed of text-to-speech output.

In order to work correctly, this requires that application developers and designers follow some principles. The first principle from the user's perspective is *"I should always know where I am"*[10]. To accomplish this principle, the user needs feedback while navigating through an application in order to be able to create a mental model of where he or she is. For most users, this can be achieved by a visual and haptic feedback during the navigation, such as labels, icons or touch feedback. As described in section 7.1.1 users with low vision benefit from verbal descriptions and large visuals with high contrast. Furthermore, the navigation should be intuitive, which can be achieved by designing well-defined and clear task flows with minimal navigation steps. Especially major user tasks should be reached within a minimum amount of steps or commands. In order to be able to control the application via accessibility gestures, any task needs to be accessible via focusable controls.

In addition, touch targets should have a size of 48 dp at least. This is important for sighted users to be able to touch the desired button and for visual impaired users to be able to find it via *Explore by Touch*.

In order to let *TalkBack* read out reasonable descriptions for any kind of important user interface element, a *content description* has to be applied. A *content description* is particularly needed for touch controls which only use an icon without any textual description. In this case, the *content description* is used for text-to-speech.

To sum things up, an accessible user interface should have an intuitive navigation, touch targets should not be too small and control elements should be labeled meaningfully.

7.1.2.2. Implementation

One of the key concerns for developers or project managers regarding accessibility is the additional effort necessary to achieve a good result in comparison to the relatively low amount of benefiting users.

When looking at the recommendations mentioned in the last section, one notices that most advices also improve the user interface for non-visually impaired users so they should be regarded anyway. Other accessibility needs like labeled user interface elements and a focusable navigation can be achieved by a few simple changes[14].

As most user interfaces for Android are defined with XML, it is also possible to define labels or content descriptions for UI elements. Those will be read out by *TalkBack* when the appropriate element is focused.

```
1 <ImageButton
2     android:id="@+id/add_note_button"
3     android:src="@drawable/add_note"
4     android:contentDescription="@string/add_note"/>
```

Listing 7.1: Content Description in Layout XML file

For a flexible usage, it is also possible to change an element's content description at runtime by using the method `setContentDescription`.

In order to control an application via the directional pad (D-Pad) or gestures, control elements have to be focusable. This is the default behavior for elements like *Buttons* but it is not for UI elements which also can be used for non-navigable purposes like *ImageViews* or *TextViews*. In this case, the `focusable` attribute has to be set to `true` explicitly. Furthermore, the focus order can be set explicitly by setting which element is below, above, left or right to the current element. Listing 7.2 shows an example of two focusable views, whereas one is set focusable explicitly. In addition, the focus order is defined.

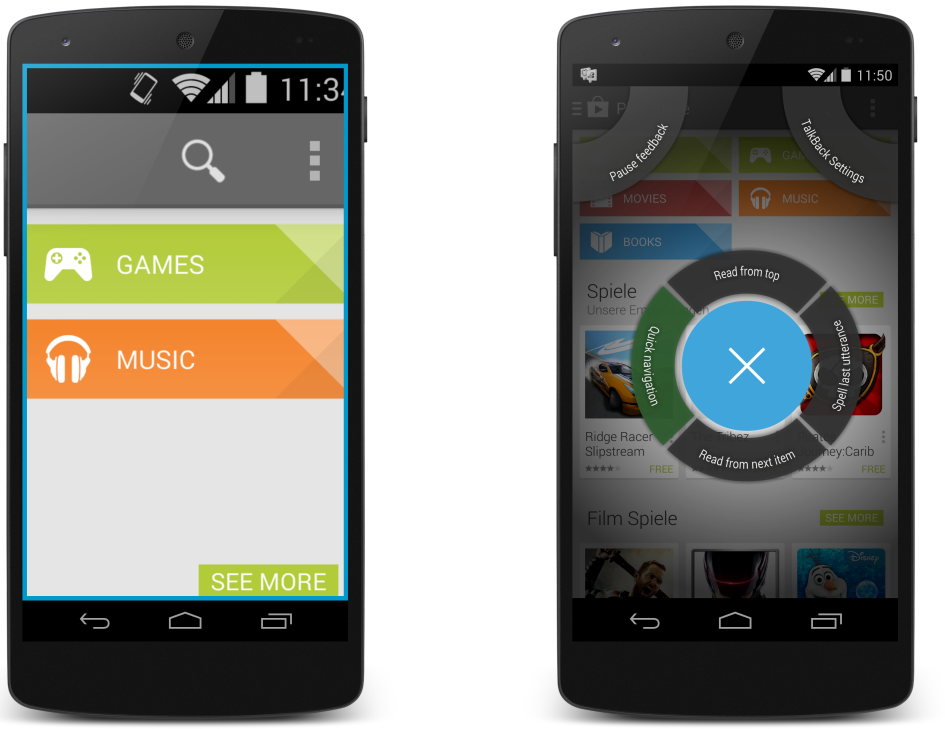
```
<LinearLayout android:orientation="horizontal"
        ... >
    <EditText android:id="@+id/edit"
        android:nextFocusDown="@+id/text"
        ... />
    <TextView android:id="@+id/text"
        android:focusable="true"
        android:text="Hello, I am a focusable TextView"
        android:nextFocusUp="@id/edit"
        ... />
</LinearLayout>
```

Listing 7.2: Focusable Elements in XML

7.1.2.3. History of Accessibility in Android

Android is an operating system with lots of improvements over its short lifetime, but the topic accessibility obviously was put in second place for a long period of time[15]. From Android 1.6, released in 2009, till Android 3 released in the mid of 2011 there were no major accessibility enhancements. There was a text-to-speech API and the user was able to switch to a *large text mode*.

With Android 4.0 released in December 2011, touch exploration has been introduced, important for devices without a D-Pad. With Android 4.1, *services* are able to set an activity's focus which is necessary for the also introduced gesture navigation. Furthermore, the focused view is indicated by a yellow rectangle, regardless whether the developer took care of it or not. In addition, Android 4.1 is the first Android version supporting Braille input and output devices.



(a) Magnification within Google's Play Store

(b) Quick Context Menu for Accessibility Settings

Figure 7.1.: Screenshots of Android's Accessibility Features

Android 4.2 added magnification. Users are now able to zoom in to applications' user interfaces by a triple tap. Application developers do not need to change anything in their applications, to let magnification work correctly. In addition to that, a global quick context menu for accessibility settings is available via a gesture. No matter which application is currently shown on the device, the user can access important accessibility features like *repeat last utterance* or *pause feedback* with a single tap.

7.2. Graphical User Interface

When Artur Klos[16] and Patrick Winter[7] defined the user interface for MoCaInfo in 2011, the current version of Android was 2.3 (Gingerbread). As mentioned in the previous section, Android's accessibility features were quite undeveloped at that time. Therefore, a barrier-free application had to implement accessibility itself. Under these conditions, it is reasonable why Klos and

Winter designed two different user interfaces instead of choosing a *universal design*. This section describes the graphical user interface for sighted people and its design patterns.

7.2.1. Design Patterns

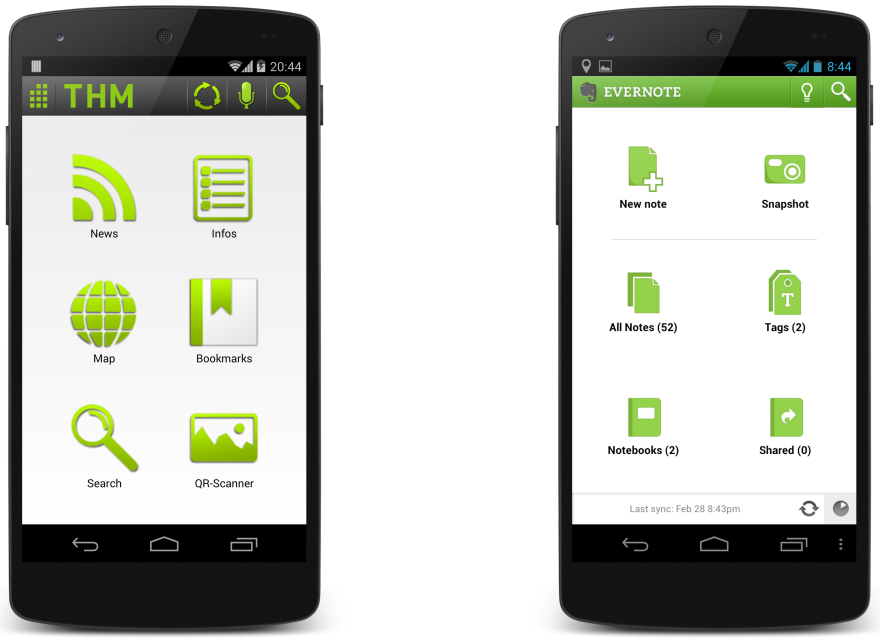
Even for user interfaces, which do not take care of accessibility, it is highly recommended to use user interface design patterns. Design patterns became common practice since Gamma et al. released the book *Design Patterns: Elements of Reusable Object-oriented Software*[17] in 1995. The design patterns described in it are targeting the software design at source code level, but the principle of patterns has been adapted for user interface design as well. A design pattern describes a reusable solution for a defined problem class within a given context. User interface design patterns do the same for user interfaces. Users, as well as developers, benefit from them. From a user's perspective, applications which use well-known UI design patterns, are easier to use because the user probably already has used a similar user interface before. From a developer's perspective UI design patterns are important for various reasons. One reason is that UI design patterns are usually highly accepted by users if used in the right context. This leads to a better user satisfaction. Furthermore, a developer does not need to reinvent the wheel. In most cases, there are libraries which enable developers to implement UI design patterns with a minimal amount of effort

UI design patterns for Android are described by Google itself[18] as well as by various authors who have a focus on mobile user experience and mobile user interfaces, e.g. Lehtimaki. The following descriptions are based on one of his books[19].

7.2.1.1. Dashboard

The *Dashboard* is one of the oldest Android UI design patterns. It is usually used for an app's landing page. The screen is designed simple, showing not more than six large icons. Some additional information like recently updated items can be shown optionally.

The design pattern is recommended for applications with more than one logical part. Any dashboard icon provides a direct link to the appropriate part of the application. In case of MoCaInfo candidates for dashboard icons are *news*, *nearest POIs*, *bookmarks*, *navigation*, *search* and *QR code scanner*.



(a) MoCaInfo (b) Evernote

Figure 7.2.: Examples of the Dashboard UI Design Pattern

7.2.1.2. Action Bar

The *Action Bar* is a styled top bar that consists of the app icon and contextual action buttons. Besides the action buttons which are directly shown, there can be additional actions within an overflow menu. An implementation of this pattern was introduced in Android 3.0. Applications targeting older Android versions needed to use a third party library. Recently, Google made the `ActionBar` classes available in their Android support library. Today, the Action Bar is one of the most prominent recall values of Android user interfaces as it is used by nearly any application.

Besides the better look, compared to the old window title bar, the Action Bar gives the user a better sense of the app's structure. The contextual action buttons can be used to execute actions, but also to indicate the state of an

application, e.g. if it is currently loading data or not. Furthermore, Action Bars can be supplemented with a search capability. Both, loading indication and searching of is used in MoCaInfo.

(a) MoCaInfo (b) Google Play Store

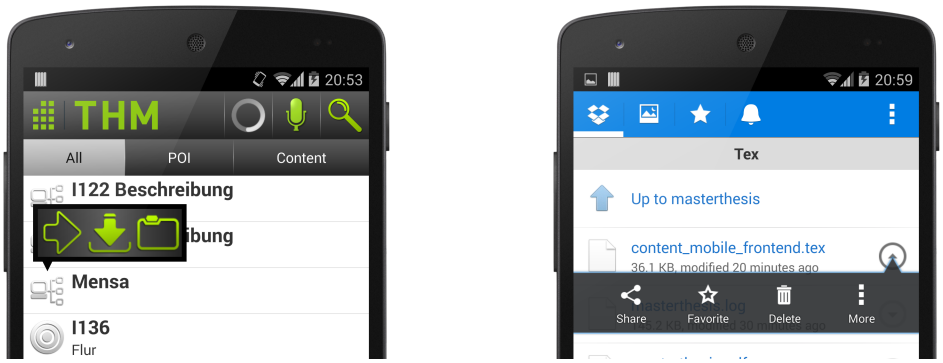
(c) MoCaInfo (Search) (d) Google Play Store (Search)

Figure 7.3.: Examples of the Action Bar UI Design Pattern

7.2.1.3. Quick Actions

The *Quick Actions* design pattern is used to show additional actions for one or a few items on the screen. It is often used together with a list of elements, whereas *Quick Actions* are used to provide contextual actions for each list item. Most likely, actions such as *delete*, *edit* or *move* are provided. In MoCaInfo, users can *bookmark* a POI or *navigate* to it.

Usually the *Quick Actions* list is an overlay, often with the shape of a balloon. It is displayed whenever the appropriate icon of the list element is touched.



(a) MoCaInfo (b) Dropbox

Figure 7.4.: Examples of the Quick Actions UI Design Pattern

7.2.1.4. Pull-to-Refresh

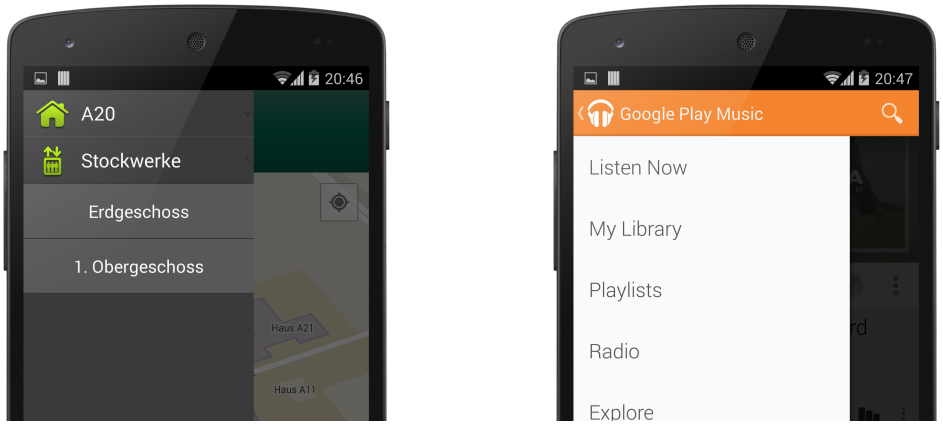
Pull-to-Refresh is a slightly controversial design pattern, more prominent in iOS than in Android. The *Pull-to-Refresh* pattern allows users to refresh a list, by pulling it down when it is already scrolled all the way to the top. *Pull-to-Refresh* is also often used the other way round, refreshing the list when pulling up the scrolled to bottom list. This depends on the order of the data within the list.

MoCaInfo's usage of this pattern is a little bit different. Instead of *Pull-to-Refresh* it could be called *Pull-to-Load-More*. When the user is viewing a list of the nearest POIs, the list is limited to a certain amount of entries. If the user reaches the bottom of the list, he will be able to load more POIs by pulling the list up.

7.2.1.5. Navigation Drawer

The navigation drawer is a panel that transitions in from the left edge of the screen. It usually displays navigation options. An advantage of the navigation drawer compared to other navigation patterns is that it does not occupy any size of the screen if it is not needed. This invisibility is also its main disadvantage. As a result, users might miss the navigation drawer, thus some of the app's features. Therefore, it is recommended to show the navigation drawer at first start so that the user notices that there is one.[20]

In MoCaInfo, the navigation drawer is used to select a building and one of its levels to show it at the map.



(a) MoCaInfo (b) Google Play Music

Figure 7.5.: Examples of the Navigation Drawer UI Design Pattern

7.3. User Interface Structure

Figure 7.6 shows the structure of the graphical user interface. Any box represents an Android Activity, except for *POI/content list* which has a more abstract meaning. It just visualizes that the connected boxes like *news*, *surrounding*, etc. list POI and content elements.

As described in section 7.2.1.1, the *dashboard* pattern suits perfectly for an application with 2 to 6 parts. Therefore, the *home* screen has icons which lead to six other screens. If the user decides to open the *map* screen, he will see his current position, at an indoor or outdoor map. This depends on whether the user is inside a building or not. *News*, *surrounding*, *bookmarks* and *search* are pretty similar. All screens show a list of POIs and content elements. Only the selection of elements differs between the screens. From those list views, the user is able to show a more detailed view of a chosen POI or content element. As QR codes are directly linked to a POI or content element, the *QR Scanner* opens the appropriate view when a code has been recognized successfully.

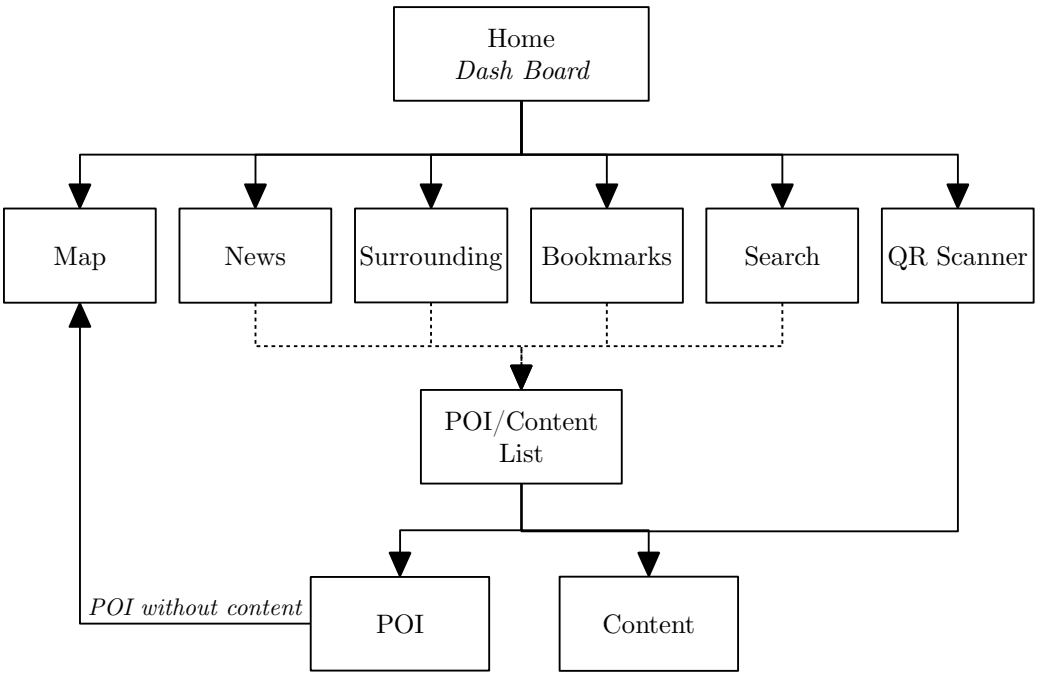


Figure 7.6.: Wireframe of Graphical User Interface

7.4. Textual User Interface

The textual user interface follows the principles described in section 7.1.1. The structure of activities does not differ from the structure for the graphical user interface. There is no need to simplify the hierarchy because the existing one is already pretty flat. This keeps the additional programming effort relatively low.

The structure of the single activities and views is completely linear. This has a bunch of advantages. First of all, it is easier to control because users only need to navigate vertically instead of navigating in multiple directions. Furthermore, a seamless scaling in size is way easier to implement because no rearranging of UI elements is needed.

As the individual visual impairments differ a lot, the user can customize the textual user interface for his needs. Text sizes, colors and the navigation type can be changed.

7.4.1. Navigation

Beside the better readability for users with remaining eyesight, the textual interface needs to be easily navigable, even by blind users. Depending on the user's preferences and device, there are different possibilities to achieve this.

7.4.1.1. Virtual Directional Pad

A common approach, even to control usual graphical user interfaces, is a virtual directional pad. The virtual D-pad allows a user to change the currently selected UI element. In combination with *Talk Back*, which presents the current element acoustically, even blind users are able to use the application. This assumes that a visual impaired user is able to use the virtual D-pad.

7.4.1.2. Gesture

An alternative to a virtual D-pad is gesture-driven navigation. By swiping up or down, the user changes the currently selected UI element. Due to the linear hierarchy, the user is able to picture the logical structure to himself easily. It should be noted that gestures need to have a simple, linear shape. More complex shapes, which includes circles, are hard to reproduce by many visual impaired users [13, p.73f].

7.4.1.3. Hardware Buttons

A problem with touchscreens for visually impaired users is the lack of haptic feedback. Therefore, users either need to discover the screen with the help of a screen reader systems or they need to fix some tactile object upon the screen at an appropriate position. Many visually impaired smart phone users glue sellotape at the position of the virtual D-pad in order to be able to hit the touchscreen buttons reliably.

As an alternative, some devices offer hardware D-pads or even hardware qwerty keyboards which are superior to touchscreen-only devices for visually impaired users. There are no official numbers about the amount of available devices, but

looking at current Android devices by major manufacturers, devices with hardware D-pads or even keyboards are rare. According to the German comparison shopping site *gh.de* only 18 out of 712 listed Android devices offer a hardware keyboard, whereas only one device comes with Android 4.0, a version which has been released in October 2011. All other devices are shipped with even older versions of Android[5]. It should be noted that this does not necessarily represent the whole Android device market, but it shows that this kind of devices are rare and getting even less popular over the last years.

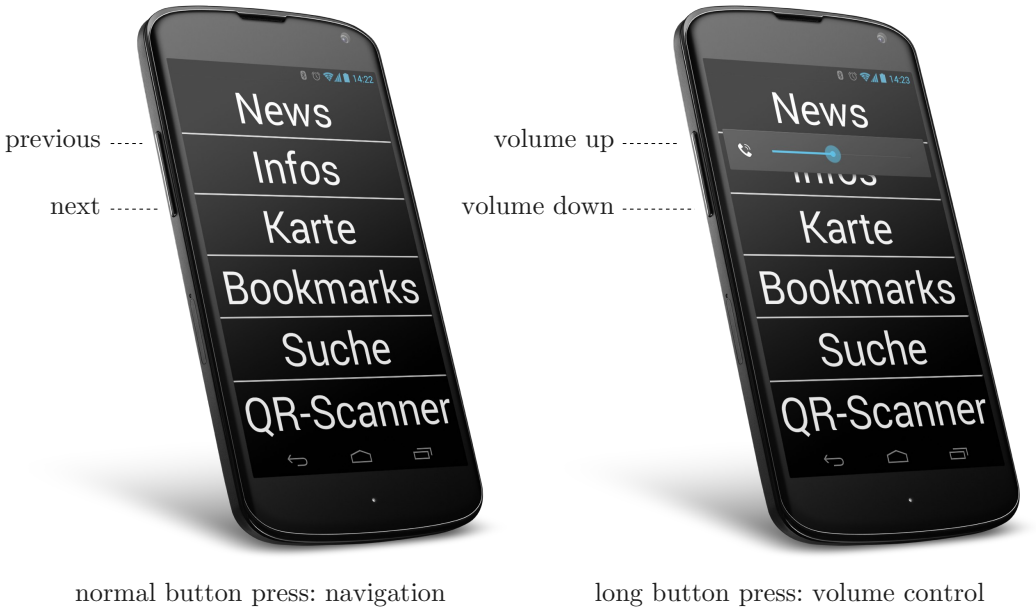


Figure 7.7.: Navigation with Volume Control Buttons

One workaround for a better usability for devices without dedicated hardware buttons is the usage of the volume control buttons. Those buttons exist at nearly any Android smart phone to date. They are usually realized as hardware buttons with haptic feedback. Instead of changing the volume, the currently selected UI element is changed. Admittedly, this workaround leads to a problem because the user is not able to change the audio output's volume anymore, which is particularly important for users who rely on text-to-speech for navigation. Therefore, a second workaround has been developed. Users are able to change the volume by long-pressing the appropriate volume button.

[5]Effective August 13, 2013

7.5. Interim Conclusion

Having two separate user interfaces, one for blind and visually impaired users and one for sighted users results in an additional development effort and complicates the maintainability. As Android's accessibility features are getting better and probably will further improve in the future, the additional expenditure seems to be unjustifiably high. Considering that only users with a remaining eyesight benefit from a dedicated low-vision user interface compared to a universal design, the low-vision UI is getting even more unattractive.

On the other hand, the numbers of the German Federal Office of Statistics clarify that the group of users with a remaining eyesight is bigger than one might expect at first glance. According to their report[21] 102,789 visually impaired people, aged 18 to 65, were living in Germany at 2011, 31th December. Only 22,973 of them were classified as blind. The remaining 79,816 people either have a high grade visually impairment or other visually impairments. This means, about 77% of the second largest target group potentially benefit from a dedicated low-vision user interface.

Because 77% of a target group cannot be ignored, *MoCaInfo* will stick to the approach of two different user interfaces, one for sighted and one for visually impaired users.

Part III.

Navigation

8. Introduction to the Navigation System

Besides a location-based information system, MoCaInfo provides a point-to-point indoor and outdoor navigation system for pedestrians. This part of the thesis analyzes existing mobile navigation systems for the usage in an indoor and outdoor environment. Chapter 10 describes the implementation, which is based on two well-known map and navigation systems. The following chapters describe navigation instructions and the user interface, taking the special needs of visually impaired users into account. The part ends with a description of the software architecture.

The user experience of an indoor and outdoor navigation system should be comparable to today's car navigation systems. This includes a map which shows the current position, as well as spoken and visual navigation instructions. Figure 8.1 shows the main components, involved in a navigation system to provide these features. The realization of these components is described in this part, except positioning. A detailed description about indoor and outdoor positioning can be found in part IV.

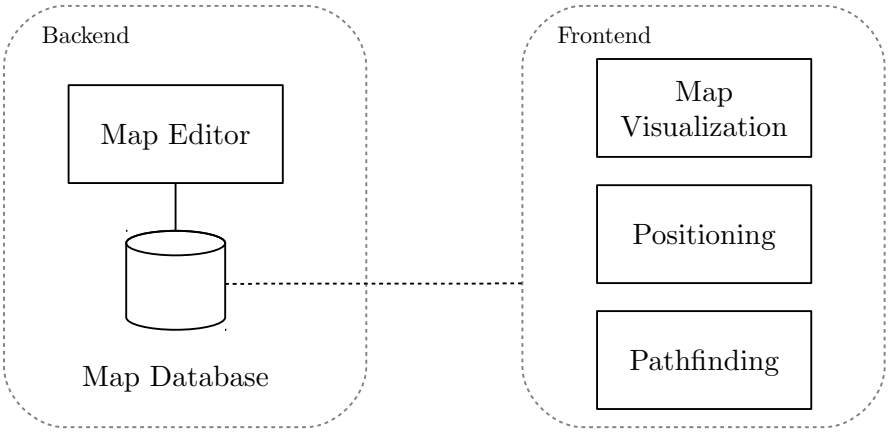


Figure 8.1.: Overview of a Navigation System's Components

9. Existing Mobile Navigation Systems

The development of a navigation system, including pathfinding, map rendering and map editing from scratch is a lot of effort. Considering that point-to-point navigation is just a single part of the overall project, solutions based on existing tools have to be considered to reduce the development effort. Therefore, existing navigation tools are evaluated for extensibility to support indoor and outdoor navigation.

9.1. Google Maps

As MoCaInfo targets Google's Android platform, the first navigation system which comes into someone's mind is Google Maps. Since 2012, Google Maps also provides indoor map rendering. For a few buildings, even indoor navigation exist[22].

A floor plan is uploaded as a picture, to enable Google to create a Google Maps indoor floor plan. Those floor plans are publicly available. Furthermore, there is no possibility to change existing floor plans or accelerate the release of floor plans by building owners. As the floor plans of the university must not be publicly available and editing has to be possible at any time, Google's Indoor Maps are not usable for MoCaInfo. An additional exclusion criterion is the miss of navigation instructions for blind and visually impaired users.

9.2. OpenStreetMap

OpenStreetMap (OSM) is a project, founded in 2004, with the goal to create a free and open geo database, which covers the whole world. To achieve this goal, OpenStreetMap motivates users to contribute geo and map data to the

project. With more than 1.4 million users who contributed data, this project is one of a kind[23].

Currently, indoor maps are not included in the OpenStreetMap standard, but the community elaborated an indoor proposal which will probably become part of a future OpenStreetMap version[24]. Nevertheless, existing tools allow to create indoor maps, based on this draft standard, today. Even though, none of the common open source map rendering tools support indoor maps yet.

9.2.1. Data Structure

The data structure of OpenStreetMap consists of three elements only: *nodes*, *ways* and *relations*[25].

Node is the basic element of any object at a map. It has a global geo coordinate and an id for referencing. Besides that, arbitrary key-value-properties can be assigned to a node. Those attributes declare how to interpret the node. Examples for standalone nodes are park benches or autotellers.

Way is a connection of *nodes*. As the name implies, they represent streets, sidewalks and other drivable or walkable paths. In addition, ways are also used to illustrate landmarks like seas or forests, as well as the shapes of buildings. In those cases, the way is usually closed which means that the last node of the way, points to its first node. As for nodes the actual interpretation of a way is given by the assigned key-value-properties.

Relation acts as an organizing element which may holds *nodes*, *ways* and other *relations*. In usual maps they are, among others, used to assign different ways to a highway with a certain identifier. Relations are also pretty important for indoor mapping, which is explained in section 9.2.2.

This quite simple data structure is stored in XML files, whereas *node*, *way* and *relation* are represented by XML tags. Their key-value properties are represented by child tags named *tag*, having an attribute k whose value stands for the attribute name and an attribute v which represents the value.

9.2.2. Indoor Proposal

The indoor proposal, also called IndoorOSM[24], is a possible scheme for mapping indoor facilities in OpenStreetMap. It is currently in *draft* status, which means that it is not included in the OpenStreetMap standard yet.

In the following, the main parts of the tagging scheme are explained. It should be noted that several tags, which are not needed for the indoor maps in MoCaInfo, are left out for reasons of clarity and comprehensibility. Those left out tags serve mainly to describe the exterior view of buildings.

9.2.2.1. Building

A building is represented by a *relation* whereby the tag *type* of the relation is set to *building*. The shape of the building is, as usual in OSM maps, a closed way. In IndoorOSM, this is also a member of the building relation. Furthermore, any level of the building is a member of the building relation. In addition to those relations, other describing tags are relevant:

key	**description**	**example**
`building`	is the relation a building?	`building=yes`
`building:levels`	number of levels	`building:levels=4`
`building:min_level`	minimum level	`building:min_level=-1`
`building:max_level`	maximum level	`building:max_level=4`
`name`	name of the building	`name=Main Building`
`type`	type of the relation	`type=building`

9.2.2.2. Levels/Floors

Each level of a building is mapped as an OSM relation, which is a member of the building relation. Basically, the level relations refer to OSM ways which represent parts of the level such as rooms, stairways, corridors in addition to navigation information like footpaths.

key	description	example
`level`	number of the level. 0 is the ground level	`level=1`
`level:usage`	usage of the levels	`level:usage=academic`
`name`	name of the level	`name=First Floor`
`type`	type of the relation	`type=level`

9.2.2.3. Parts of Buildings

Rooms, corridors, lecture halls, etc. are so-called *building parts* in IndoorOSM. A *building part* is basically a way, which describes the shape of the object. The according tag declares whether the object is a *room*, *hall*, *corridor* or *vertical passage*. Rooms usually include at least one node with a *door*-tag which represents the entry point to the room. Those door nodes have to be connected to the way network to be able to navigate to the rooms.

key	description	example
`buildingpart`	type of part of the building	`buildingpart=room`
`name`	name of the building part	`name=Audimax`
`ref`	the reference number of the building part, usually a room number	`ref=A20.1.36`

9.2.2.4. Example

The following example represents a building with three floors. For reasons of clarity, only one floor and a few parts of the building are described in detail.

```
<osm version='0.6' generator='JOSM'>
  <!-- building -->
  <relation id='1370729' ...>
      <member type='relation' ref='1370727' role='level_-1' />
      <member type='relation' ref='1370728' role='level_0' />
      ...
      <tag k='building' v='yes' />
      <tag k='building:levels' v='3' />
```

```
9       <tag k='building:max_level' v='1' />
10      <tag k='building:min_level' v='-1' />
11      <tag k='name' v='A20' />
12      <tag k='type' v='building' />
13  </relation>
14  <!-- floor 1 -->
15  <relation id='1370725' ...>
16      <member type='way' ref='94551494' role='buildingpart' />
17      ...
18      <tag k='level' v='1' />
19      <tag k='level:usage' v='academic' />
20      <tag k='name' v='First Floor' />
21      <tag k='type' v='level' />
22  </relation>
23  <!-- room -->
24  <way id='94551494' ...>
25      <nd ref='1098227358' />
26      <nd ref='1098226969' />
27      <nd ref='1098227303' />
28      <nd ref='1098226902' />
29      <nd ref='1098227358' />
30      <tag k='buildingpart' v='room' />
31      <tag k='name' v='1.07' />
32  </way>
33  <!-- door of room -->
34  <node id='1098226902'lat='50.587165546307155' lon='8.682308673862533' ...>
35      <tag k='door' v='manual' />
36  </node>
37 </osm>
```

Listing 9.1: Example of a Building in OpenStreetMap's XML notation

9.2.2.5. Extension for Visually Impaired Users

As section 11.1 explains, navigation especially for visually impaired and blind users requires some additional information for a good level of assistance. Therefore, additional describing attributes for ways are needed.

Because many of these information are directly used to create navigation instructions, they need to be gathered in multiple languages. Therefore, these attributes end with a two-lettered ISO country code. The currently used attributes for navigation look as follows:

key	description	example
`nav:floormaterial :[en|de]`	describes the material of the floor	`nav:floormaterial :en=tile`
`nav:barrier:[en|de]`	describes a barrier	`nav:barrier:en= flowerpot`
`door:openingdirection`	describes a door's opening direction as azimuth	`door:opening direction=180`

9.2.3. Map Editor: JOSM

JOSM is the most often used OpenStreetMap editor for outdoor maps. It is also capable of indoor map creation. A lot of community contributed plugins further simplify the map creation[26].

The general proceeding of creating an indoor map is to download an existing OSM outdoor map of the targeted area. In many cases, outlines of buildings still exist. By means of an overlaid satellite picture, it can be checked whether the buildings outline is accurate.

Secondly, an existing image of the floor plan is added as a picture layer in JSOM. This image can be traced manually to add walls, doors, stairways and any other needed floor plan information.

JOSM also includes a filtering capability. Those filters enable the user to explicitly show or hide nodes, relations and ways depending on their attributes. In this way, the user is able to show a certain level without getting disturbed by other over- or underlying levels. The screenshot in figure 9.1 shows an image layer and the traced result including walkable ways. The lower right shows the filter panel, which has been set to hide the building's level 0 and 2.

To enable a point-to-point or rather POI-to-POI navigation, any point of interest has to be connected to a way, usually of the type *highway=footway*. Those ways are represented by the green lines in the screenshot.

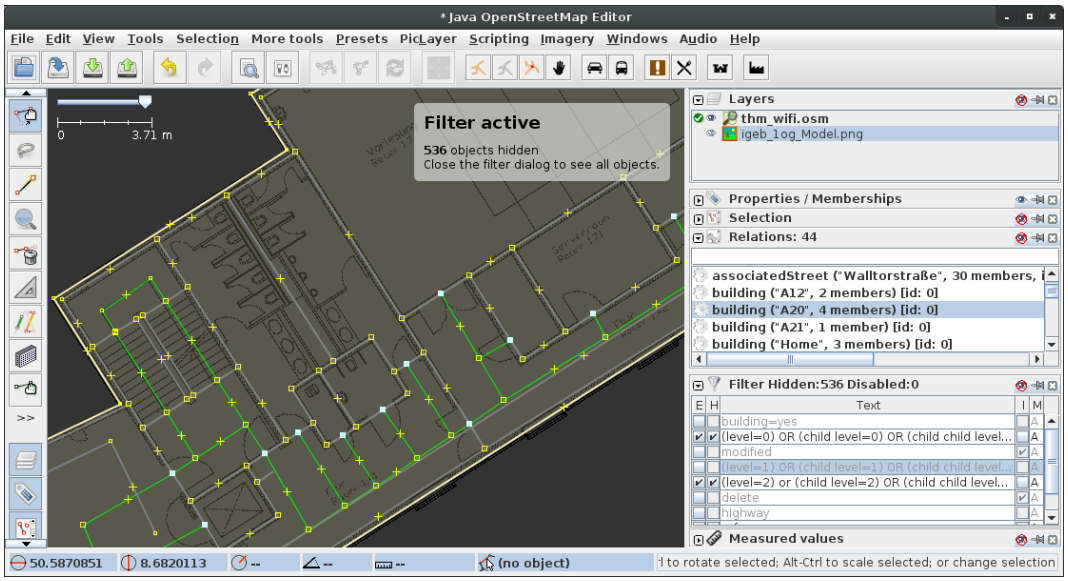


Figure 9.1.: Screenshot of JOSM

9.2.4. Existing Renderer

Even though no OpenStreetMap rendering engine with indoor support exists, there are a lot of open source rendering engines for outdoor maps. In this section, two projects targeting the Android platform are introduced and evaluated regarding extensibility for indoor usage.

9.2.4.1. Mapsforge

Mapsforge is an open source project, providing ad-hoc map rendering. The API is similar to the Google Maps API. Even so, there is a rewrite in process which will probably lead to some major API changes. The map format is binary and created via a converter tool called *osmosis* in combination with a plugin provided by Mapsforge's developers.

One of the advantages of Mapsforge compared to many other projects is the on device rendering. Many other OpenStreetMap viewers depend on pre-rendered tiles, which pretty inflexible and leads to huge application file sizes.

The major problem of Mapsforge and finally the reason why it could not be used for MoCaInfo is the map file format. On the one hand, the binary format

is quite nice because a 2MB OpenStreetMap XML file becomes a 200kb binary file, but on the other hand, this leads to a loss of information and accuracy which is unacceptable for indoor purposes. Because of the loss of accuracy, rectangular rooms become crooked, or a corridors become slightly longer than the building's shape.

9.2.4.2. OsmAnd

OsmAnd is an Android application for point-to-point navigation. It is open source, which enables other developers to use the rendering engine. Similar to Mapsforge, OsmAnd uses a native map format called *.obf*. Those files contain the drawing information in an own vector graphics format. The rendering result itself looks really professional, and the API is easy to use.

To enhance the performance, OsmAnd caches rendered parts of the map in image files. This principally useful feature does not support multi-layered maps and therefore, no floors. For that reason, the cache would have to be disabled, accepting a bad performance. As an alternative, OsmAnd could be patched to support multi-layered rendering. As the rendering and caching components are strongly coupled, this would affect a lot of sourcecode. But even then, whole tiles of the screen would have to be reloaded when the drawn floor changes, which is not a good user experience.

Finally, the necessary changes of OsmAnd would be much of an effort with a moderate result.

10. OpenStreetMap meets Google Maps

The evaluated OpenStreetMap rendering engines are not flexible enough to extend them with indoor rendering features, with a reasonable effort. The differences between a usual single layered street map and a multilevel building map demand strong changes of the renderer's architecture and map formats.

Anyways, OpenStreetMap is worth using because of the existing toolchain, like JSOM for editing the map and libraries for pathfinding. However, the development of an OpenStreetMap outdoor and indoor renderer from scratch would be overkill.

An alternative to this approach is using an existing outdoor map solution and draw all necessary indoor information like floor plans on top of it. Because Google Maps offers a good map rendering performance on Android devices, and an API for drawing custom overlays, it is a perfect fit for such an approach.

10.1. Map File

The map file follows the IndoorOSM approach described in section 9.2.2 with one additional layer. This layer is a background layer which is drawn between the GoogleMaps map and the building overlay. It makes sure that the existing GoogleMaps building shape is overlaid, as it is usually not as accurate as the building shape provided in the OpenStreetMap file. The background layer is stored as a closed way in the OpenStreetMap file.

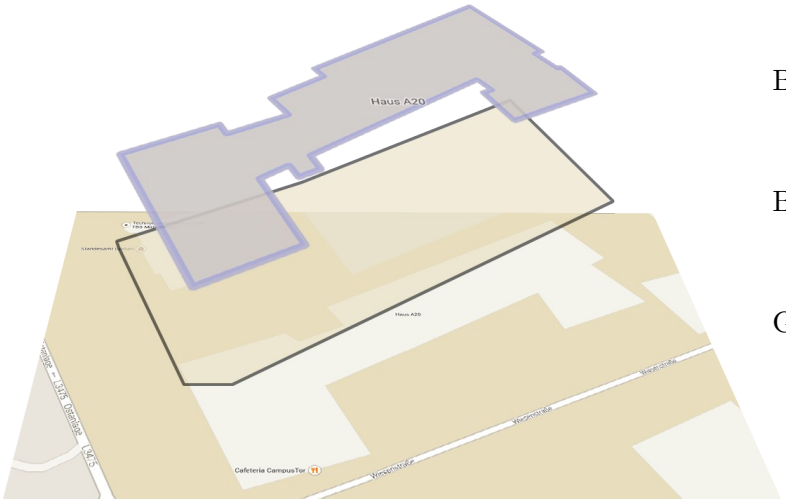


Figure 10.1.: Background Layer between Google Maps and Building Overlay

10.2. Reading Data: SpatiaLite

As OpenStreetMap's map format is based on XML, it would be possible to parse the file to extract the needed information. But there are more sophisticated open source approaches available whereas on of them is used for MoCaInfo's indoor navigation system: SpatiaLite.

SpatiaLite is a spatial DBMS, implemented as an extension for SQLite. Thus, the map data is accessed via SQL queries. Any relation between nodes, ways and relations are mapped. Furthermore, it is possible to create virtual tables to access elements of a certain OSM type easily. In case of MoCaInfo there are virtual tables for *buildings*, *building parts*, *levels* and the *background overlays*. Beside usual SQL-queries, SpatiaLite offers SQL functions to compare geometrical objects. Due to different comparative operators, it is possible to check if geometrical objects are overlapping, intersecting or touching. Such geometrical objects are points, represented by OSM nodes, polylines, represented by unclosed OSM ways and polygons represented by OSM closed ways.

To create the SQLite tables from an OSM XML file, a utility called *osm_convert* is available. It has been slightly patched to produce the virtual tables for indoor usage and skip those, usually needed for car navigation.

10.3. Pathfinding: SpatiaLite

Besides, accessing the static OSM data and filtering by geometric constraints, SpatiaLite offers pathfinding, as well.

For the best possible performance, the walkable ways are converted into a static network which represents the graph between the nodes. The generation is done by the tool *spatialite_osm_net*. This tool can be configured to weight different kinds of ways differently. For example, a network for hampered or wheelchair users, could totally ignore stairways in order to be sure that only suitable ways are considered for pathfinding.

At network generation time, it can be chosen whether SpatiaLite finds the shortest path using the well known Dijkstra or A* algorithm. To find the shortest path between two nodes, just a simple SQL query is needed.

```
1 SELECT * FROM footway_net WHERE node_from = ? AND node_to = ?
```

Listing 10.1: Example of an SQL Query to Select the Shortest Path Between Two Nodes

Compared to other solutions, which need to generate the network at runtime from the database's information, the precomputed binary network has massive performance advantages.

10.4. Data Mapping: MoCaInfo and SpatiaLite

Points of interest, buildings and other data gathered in an OpenStreetMap XML file and converted to a SpatiaLite database need to be accessed by other MoCaInfo components, as well. Therefore, this information needs to be stored in the global MoCaInfo database, following MoCaInfo's data model, explained in section 3.3. To achieve this, a tiny data mapping tool was implemented which copies buildings, levels and POIs from the SpatiaLite database to MoCaInfo's MySQL database. To identify this information for later updates, the MoCaInfo data model has been extended to store the OpenStreetMap id of any element, in addition to the id used internally.

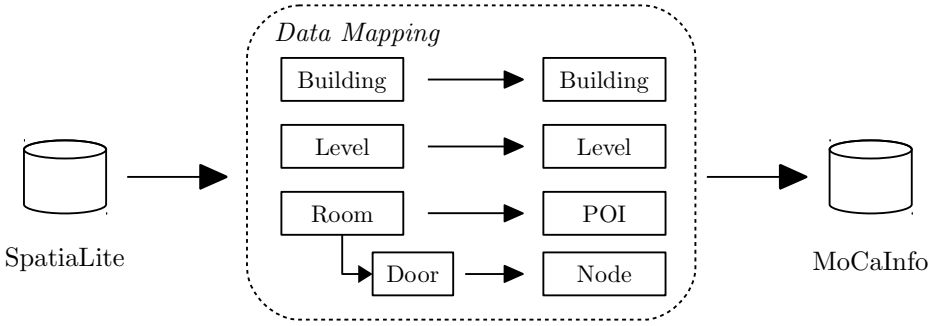


Figure 10.2.: Data Mapping Concept

Buildings and levels can easily be mapped to the according database tables in MoCaInfo. Rooms are mapped to point of interests in MoCaInfo, whereas the door's geo coordinate is used for the POI's node.

Due to this data mapping, there is no direct dependency between the navigation component, and other components of MoCaInfo. As a result, it is possible to use the navigation component in other projects not related to MoCaInfo.

11. Navigation Instructions

Point-to-point navigation systems use navigation instructions, which are represented by symbols and an audiovisual explanation, to guide the user to the target destination. Pedestrian navigation systems slightly differ from car navigation systems. In contrast to car navigation systems, pedestrian navigation systems do not need to take care of roundabouts or multi-lane roads. On the other hand, additional instructions dealing with stairways and building entrances have to be considered.

The following enumeration lists navigation instructions needed for a reasonable indoor and outdoor pedestrian navigation system.

turn directions are the most common navigation instructions in point-to-point navigation systems. In detail, there are instructions for slightly, normal and sharp turns to the left or right direction.

stairways are navigation instructions which are dedicated for indoor pedestrian navigation. The navigation system also show the user whether the stairway goes up or down.

elevators are another form of vertical passages which have to be described if they are on the user's way.

entrances indicate to the user that he enters a new building.

11.1. Additional Instructions for Visually Impaired Users

As already stated in section 2 and section 7.1, visually impaired and blind people need to have some additional information in order to be able to bene-

fit from a navigation system. Niehaus describes several additional supporting navigation instructions for blind and visually impaired users[13].

doors are important for blind users especially if they are located within the route. For an optimal assistance, the kind of door is needed also. Currently differentiated door types are room doors, manual fire doors, automatic doors and revolving doors. Furthermore, the opening direction acts as a reference point for users with low vision. With the help of this reference point, the user can check whether the current situation is consistent with the navigation system's instructions.

Another example where the information about doors comes in handy is when the targeted door is at a corridor. In this case, the system is able to tell the blind user that the targeted room is, for example, the fifth door on the right hand side. The visually impaired user can count the doors with his blindman's stick. Figure 11.1 illustrates the two different navigation instructions, depending on whether corridor and door information are taken into account or not.

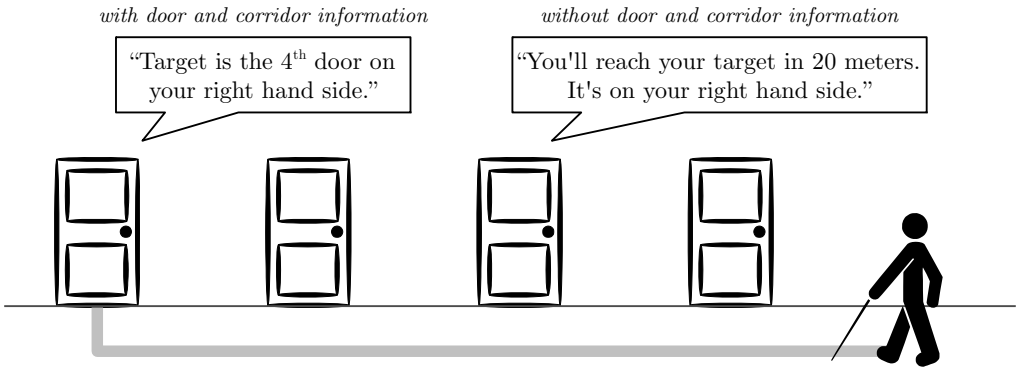


Figure 11.1.: Navigation Instruction with and without Door and Corridor Information

floor and wall conditions are further information which may help the user to make sure that he follows the navigation instructions correctly.

corridors help blind users to orientate because they can follow the wall. This is much easier than orienting at halls without a wall as a reference.

objects like desks, chairs, flowerpots, etc. are likewise barriers and reference points and should therefore be considered.

12. Navigation System's User Interface

The user interface follows the user interface known from Google Maps. Therefore, users who used Google Maps before, feel familiar when using MoCaInfo's navigation system for the first time.

At the top of the screen, there is a large actionbar, about twice the size of a usual actionbar. It shows the current position in a textual representation, e.g. "*Room 1.07, Building A20*". If the user starts a point-to-point navigation, the left corner holds a visual representation of the next navigation instruction and the distance till the navigation instruction should be followed. Beneath that, there is a tiny symbol indicating the upcoming navigation instruction. Below the actionbar, there is the actual map view.

As visually impaired users cannot benefit from the detailed map view and tiny navigation instructions, there is a dedicated low-vision navigation user interface. This low-vision user interface only shows two upcoming navigation instructions and the current position as a text. Both can be seen in figure 12.1.

If the user just wants to explore the map, he can select buildings by touching them. This opens a list of the building's levels. From this list, the user can choose one level for showing. The list is implemented, using the navigation drawer pattern explained in section 7.2.1.5. This has the advantage that the user is able to open the list to select a building or level, even if the campus' buildings are currently out of sight. As another positive side-effect, the list is independent from the actual map view, which makes it reusable for the low-vision user interface.

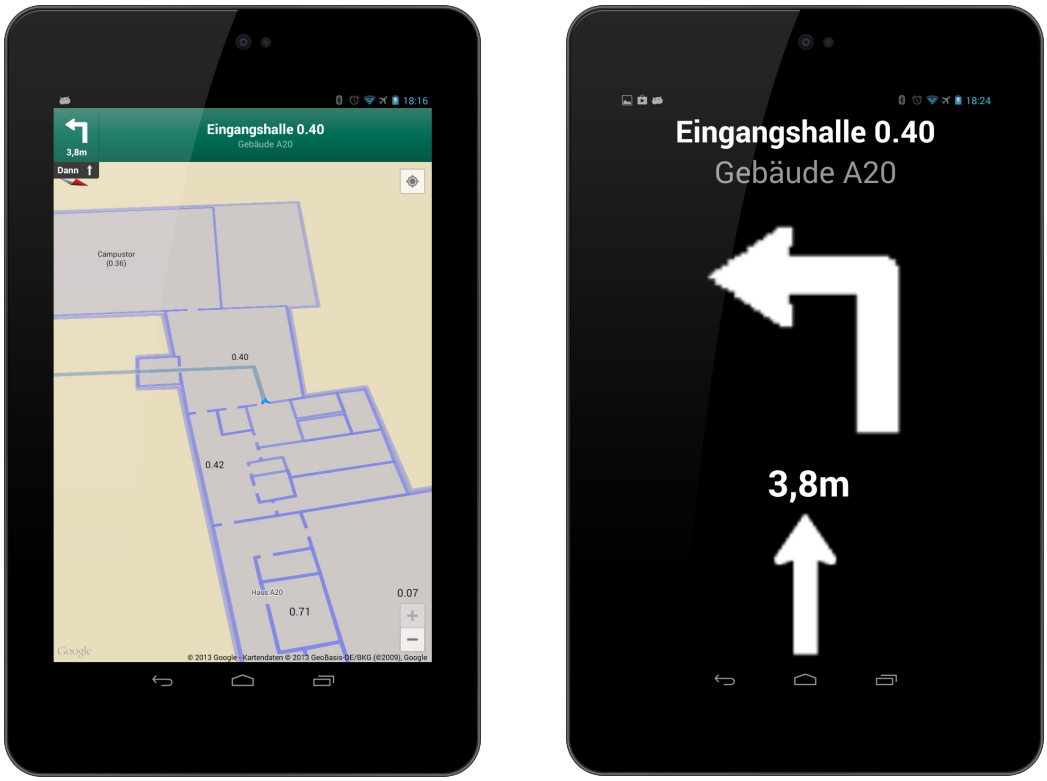


Figure 12.1.: Screenshot of MoCaInfo's Navigation Component

13. Navigation System's Software Architecture

The following section gives an overview about the navigation component's architecture. It should be noted that the architecture presented here is not exhaustive because a detailed explanation would be too lengthy without added value.

Figure 13.1 shows the main classes of the component. For reasons of clarity, a simplified UML notation is used which does not include packages, attributes, methods and class types.

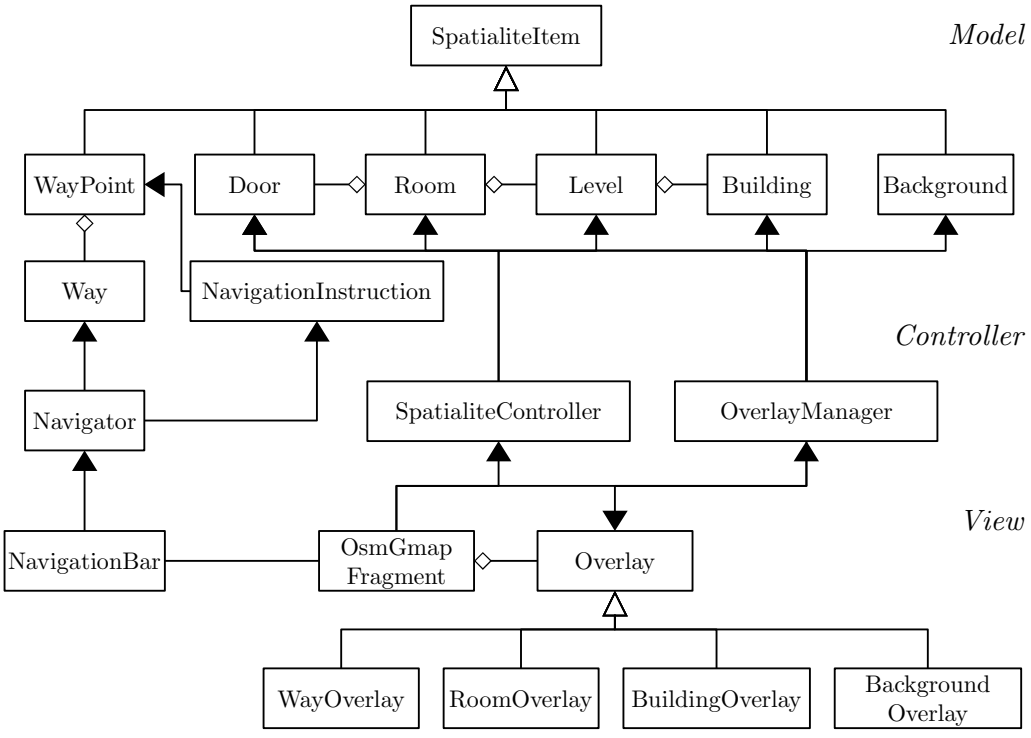


Figure 13.1.: Navigation Component's Architecture

The architecture follows the well known Model-View-Controller (MVC) pattern. MVC intends that classes either have a model, a view or a controller character. View classes usually provide the visual representation of model classes. These model classes only hold the data. Controller classes mediate between view and model and handle user interactions.

In case of this architecture, mostly any model is derived from the abstract class *SpatialiteItem*. *SpatialiteItem* holds basic values of a SpatiaLite dataset, such as an id and arbitrary key value properties. Any element, needed for indoor and outdoor navigation, which is stored in the SpatiaLite database, is implemented as a child of *SpataliteItem*. To these belong the classes *Building*, *Level*, *Room*, *Door* and *WayPoint*.

The loading of the data is realized by *SpatialiteController*. It creates the *SpatialiteItems* based on the database's data by using the SpatiaLite library, which is not shown in the figure.

To draw the different parts of the building, appropriate *Overlays* need to be created. The efficient creation is managed by the *OverlayManager*. Furthermore, the *OverlayManager* links between an *Overlay* and its model class.

The *Navigator* manages navigation related features. It takes care of the pathfinding when requested, and creates the needed *NavigationInstructions*. The *NavigationBar* is responsible for visualizing the computed *Way* and the *NavigationInstructions*.

The class *OsmGmapFragment* is the main entry point for the user. Besides the *NavigationBar* and *Overlays*, it holds the Google Maps view and other view elements. The actual user interaction can be implemented by using listeners, e.g. a *RoomClickedListener*. Among others, they are not shown in the figure to keep clarity. This listener concept enables other developers to use the navigation component in their applications, with customized user interactions. Furthermore, it should be noted that the navigation component itself, has no dependencies on other MoCaInfo components, which facilitates the usage of it in other projects.

Part IV.

Indoor and Outdoor Positioning

14. Introduction to Indoor and Outdoor Positioning

Positioning is an essential feature of a point-to-point navigation system. The best-known positioning system today is the Global Positioning System (GPS) which is described in section 15.1. As GPS is based on a line of sight to satellites, it is not available in buildings. Therefore, alternative positioning approaches are evaluated and developed in this part of the thesis, to achieve accurate indoor and outdoor positioning.

In so doing, it is differentiated between absolute and relative positioning. Absolute positioning estimates an absolute position at earth or within a building. Relative positioning approaches are only able to estimate changes in position.

The following section describes well-known and often used absolute positioning approaches. Followed by a section which deals with the topic of relative positioning. Right after this general description, approaches which are used for MoCaInfo's positioning system are explained and evaluated in detail. The part ends with an explanation of MoCaInfo's dead reckoning system, which combines absolute and relative positioning to improve the accuracy.

15. Absolute Positioning

Absolute Positioning techniques try to estimate a user's absolute position at earth. Some of the approaches deliver a three-dimensional position, e.g. for GPS this is longitude, latitude and altitude. Others do not take care of the height above sea level. Therefore, the estimated position is just two dimensional. This chapter introduces various absolute positioning methods, which are applicable for modern smart phones.

15.1. Global Positioning System

The Global Positioning System (GPS) is the most popular global navigation satellite system (GNSS) worldwide. It was developed by the U.S. military, to provide precise estimates of positions. Today, it is also available for free, civil usage.

GPS is based on a simple and ancient idea: an object's position can be determined by having distances to other objects whose positions are known. In case of GPS, satellites broadcast their position, so these are the objects at known locations. The distance between a GPS receiver and a GPS satellite is measured by using the transit time of the signal from the satellite to the receiver. The transit time can be calculated easily by having synchronized clocks at the satellites and the receivers[27].

The accuracy of GPS increases with the number of satellites which can be seen by the receiver, whereas four is the minimum amount to determine a three-dimensional position. According to the Federal Aviation Administration's (FAA) report from August 2013, the median position dilution of precision (PDOP) was 2.8 meters, having six satellites in sight in 99.99% of the test

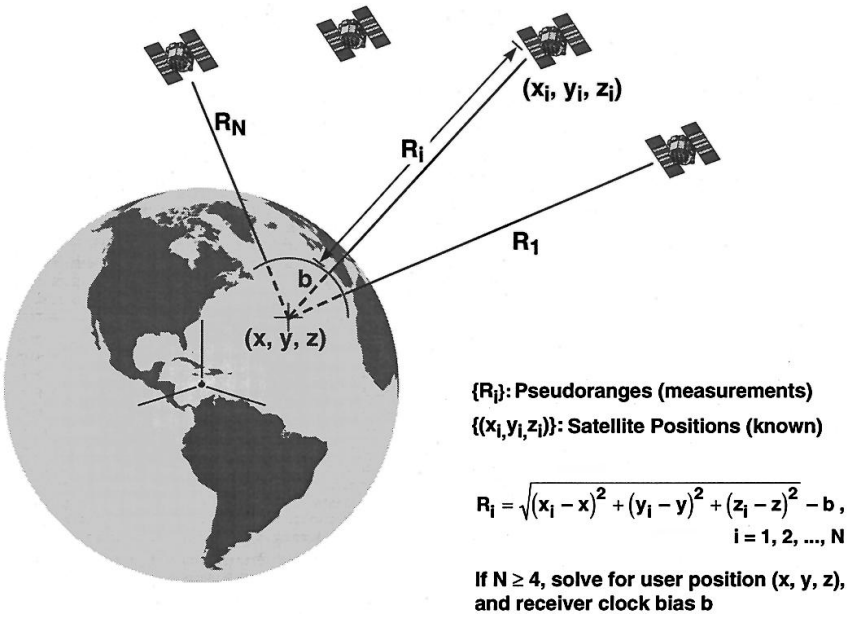


Figure 15.1.: Principle of Satellite Positioning. Illustrated in [27].

cases[28]. The worst position had an error of 7.4 meters. The test system uses the 24 satellites, available for civil usage.

Besides GPS, there are other global navigation satellite systems such as the Russian GLONASS and Europe's Galileo, which is still in development. Because GPS receivers need to have a line of sight to at least four satellites, positions cannot be determined inside of buildings. In order to enable GPS positioning in buildings, GPS repeaters can be installed. Those devices have to be installed at positions with satellites in sight and repeat the signal via an antenna inside the building. It should be noted that GPS receivers inside the building will only receive the repeater's position, not the actual position of the receiver itself.

15.2. GSM

Another widely-used positioning approach, especially for cell phones, is to use the GSM network itself. GSM, short form of *Global System for Mobile Communications*, is a standard set which describes protocols for digital cellular networks, also known as 2G or second generation protocols. Most cell phone

carriers worldwide implement this standard. To mention some numbers, in 2011 there were 656 million active GSM SIM cards in the European Economic Area, which means that any European owned approximately 1.3 SIM cards at an average[29].

15.2.1. Cell Identification

The simplest approach to estimate the position of a GSM device is to assume the device's position as the position of the cell tower to which it is associated. Therefore, accuracy depends on the dimension of the area covered by the cell tower. In rural environments, these areas have a radius up to 35 km, and even in urban territory usual cell towers cover an area with a radius of 500 meters to 5 kilometers. The accuracy can slightly be improved if more than one cell tower is in sight of the device. The position can then be estimated to be inside the intersection of the cell tower's areas. According to a research of the University of Zagreb, the positioning accuracy can considerably be improved up to 50 meters in urban environments. For indoor environments, the accuracy can be improved by placing femtocells, which usually have a range of 10 to 50 meters. [30]

15.2.2. Angle of Arrival

A more sophisticated way of estimating a GSM device's position is called Angle of Arrival (AoA). AoA requires at least two cells in sight and a complex antenna array at each of them. The antennas work together to estimate the angle of the signal which then can be used to triangulate the handset's position. AoA does not work well in urban environments because the radio signals are reflected by buildings or other obstacles which leads to wrong estimations of the angle.[30]

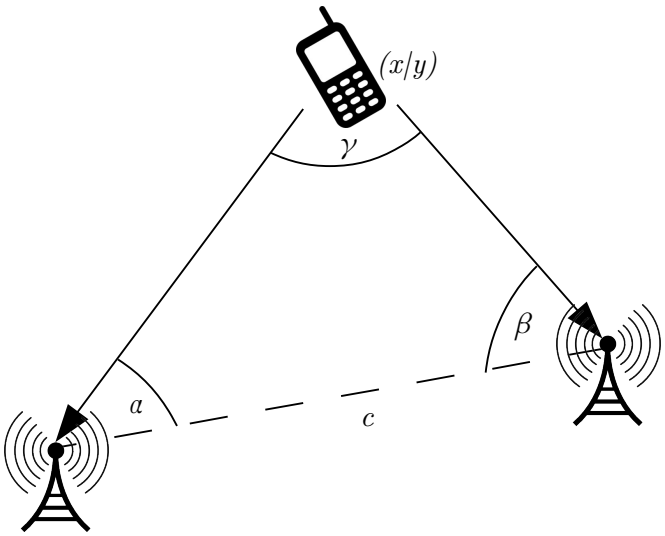


Figure 15.2.: Illustration of a AoA Triangulation. α and β are Known Because of the Antenna Arrays

15.2.3. Time of Arrival

The Time of Arrival (ToA) approach uses the signal's transit time from the mobile device to a number of base stations to determine the position. In order to measure the signal transit time, mobile devices and base stations need to be synchronized in time, and the signal itself needs to be tagged with the submission time. In accurate synchronized networks, an average error between 125 and 200 meter is achievable. The accuracy deteriorates drastically if base stations and mobile devices are non-line-of-sight. The reasons are reflections or attenuation due to objects, passed by the signal, which influence the signal's transit time.[30]

15.2.4. RSSI-based positioning

In contrast to ToA and AoA, RSSI-based positioning takes place at the mobile device itself. RSSI, short form of received signal strength indication, describes the signal quality to a base station. As the signal strength depends on lots of factors like buildings or other obstacles laying between the handset and the base stations, it is not possible to estimate the position directly from the RSSI.

To determine a position from RSSI values, fingerprints have to be created. A fingerprint associates a position to RSSIs measured at this position. Those fingerprints are usually stored in a database. Later, these fingerprints are compared to the RSSIs measured by the handset, in order to estimate its position. Because the creation of fingerprints for large areas is a time consuming task, some projects like *CellSense*[31] compute fingerprints, based on a histogram of signal strengths for the base stations. In case of *CellSense*, they were able to achieve an accuracy of 30.05 meters in an urban environment.

15.3. Wi-Fi

Another radio-based approach is to use wireless local area networks for positioning. In this work, the term Wi-Fi is used for products based on the IEEE 802.11 standards, similar to the definition of the Wi-Fi Alliance[32].

15.3.1. Market Overview

Wi-Fi-assisted positioning systems are used the most in indoor positioning systems. However, there is nothing like a standard, and most solutions are only developed for scientific research and less for daily usage. Nevertheless, more and more well-known companies are releasing products. Many of these products are only considered for usage in experimental environments. For example, the Mozilla Foundation started their free service *Mozilla Location Service*[33] in the end of October 2013. The Wi-Fi cell information is provided by users, whereas any application developer can to use a REST API to get location information later on. The *Google Location Service* is a similar project by Google, which also provides a free to use interface for application developers, but there is no possibility to add own Wi-Fi information.

In addition to those companies, other device manufacturers such as Apple[34] and Cisco[35] have their own Wi-Fi-based positioning systems, as well. However, none of those companies make precise statements about the functional principle and accuracies.

15.3.2. Angle of Arrival

In general, position determination approaches for Wi-Fi-based systems are similar to the ones for GSM. One method, also applicable for Wi-Fi is the Angle of Arrival approach, explained in section 15.2.2. Like for GSM, special wireless access points with antenna arrays are needed to determine the position at the access point and send it back to the client.

Based on the AoA concept, Kawauchi et al.[36] developed a system which adds directional information to the *beacon*. The beacon is a tiny data package of the IEEE 802.11 standard, which is send in intervals and used to determine nearby access points and their signal strength. Therefore, they mounted a motorized, rotatable directional antenna to a wireless access point and modified the firmware, to add the current rotation of the antenna to the send out beacon. The location can now be estimated based on this angle and the known position of the wireless access point by using triangulation.

15.3.3. Time Of Arrival

Time of Arrival (ToA) is a trilateration-based positioning system, whereas the distance to at least three access points is determined based on the signal runtime. The distance can be seen like a circle around the access points, whose position is known. The mobile device's location is the intersection point of these circles. As the currently used IEEE 802.11a,b,g,n standard does not include signal runtimes, needed for ToA, modified hardware and drivers are needed to implement an accurate positioning system.

However, Ciurana et al.[37] developed a software-based solution, working without the need of specialized hardware. The Wi-Fi clients sends a data frame to the access point and waits for the acknowledgement. The time between sending and receiving the acknowledgement is assumed to be the round-trip time (RTT). As those RTT might be influenced by various external factors, such as walls or other radio frequency devices, 1000 RTTs are measured and filtered based on statistical information before the position is estimated. They tested the distance determination indoor and outdoor and got an average absolute error of 1.7 meters indoor, respectively 1.84 meters outdoor.

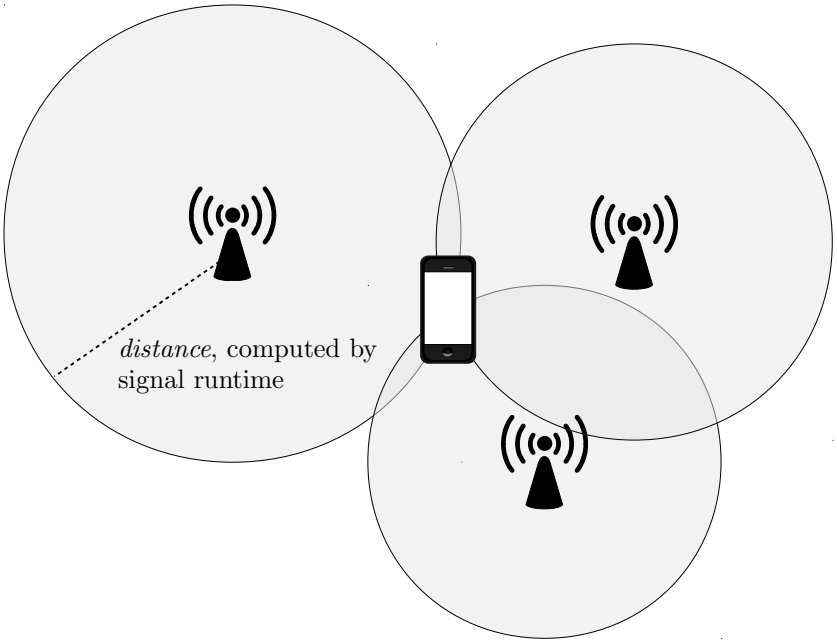


Figure 15.3.: Illustration of Time of Arrival (TOA)

15.3.4. RSSI-based Fingerprints

ToA- and AoA-based systems are pretty rare as they require specialized hardware or modified drivers to achieve a reasonable accuracy. Because of that, most research systems rely on signal strength fingerprints[38],[39],[40] which do not need modified hard- or software.

The idea behind that approach is quite simple. The mobile device receives beacons from nearby access points and their RSSI. All received access points and their RSSI values, can be seen as a fingerprint for the device's current location. In order to determine the mobile device's position, a database is needed which links fingerprints to positions.

Therefore, an offline learning phase is used to measure fingerprints which are linked to positions manually. To determine the mobile device's position later on, the currently measured fingerprint is compared to the fingerprints in the database. Different approaches for comparing fingerprints and determine locations are described in section 17.5.

The common accuracy of existing research systems under good conditions is about 2 to 7 meters in average[38],[41],[42].

15.4. Optical

Today's smart phones and tablets usually have a camera, which is capable of taking high definition photos and videos at their back. Many devices even have a second front-facing camera. Therefore, optical indoor positioning is an option worth considering. The following description is mostly based on a work done by Mautz and Tilch[43]. They surveyed several different optical positioning systems, whereas most of them are using one of the three following approaches.

15.4.1. Floor Plan Method

The most sophisticated optical indoor positioning approach is called the *floor plan method*. The picture, taken by the camera is interpreted to extract properties of a room, such as walls, doors but also desks, chairs, bookshelves and other furniture. This abstracted image information is compared with data from a database to determine the position. In order to have such a reference database, the rooms are described with a CAD-like software, as for instance CityGML.

To extract properties of a room or floor, different approaches are used. A program called Kohoutek uses a special 3D camera to create a 3D model of the area. A simpler approach, using a casual camera, has been developed by Hile and Borriello[44]. Their software detects edges in images and compares them with the edges of simple 2D floor plans. Under ideal circumstances, the error was between 0 and 15 cm in more than 70% of the test cases and never more than 60cm. The problem of this approach is that it gets confused easily. For example, people walking at the floor or non-recorded objects, like baskets influence the edge detection. In those test cases, a location could not be determined in more than 30% of the time.

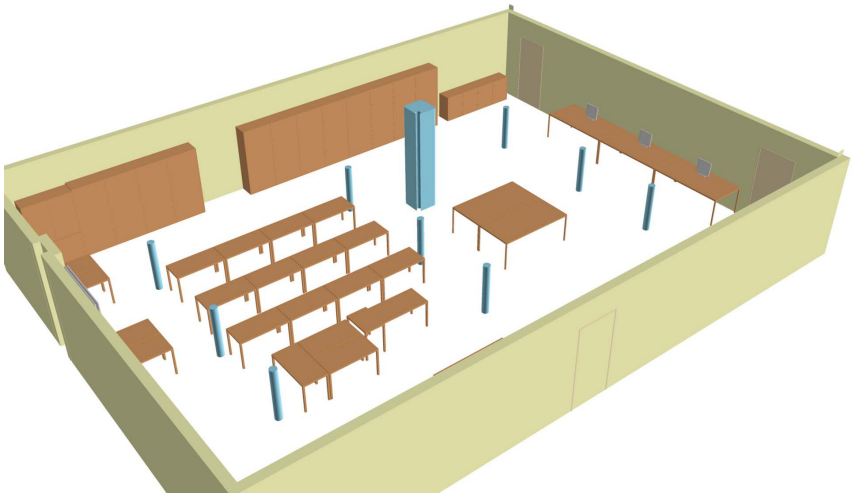


Figure 15.4.: Model of a Room in CityGML[43]

15.4.2. Template Matching

A second method to determine a user's position with a camera is template matching. In an initial phase, images are taken and linked with a position. To estimate a position, the user's camera takes a photo which is compared to the existing images by calculating a correlation coefficient. The position of the image which fits best is used as the determined position. Like the *Floor Plan* method, results depend highly on a non-changing area. A simple change, like a new picture at the wall, might bring this method to fail.

15.4.3. Deployed or Projected Targets

In order to increase robustness and improve accuracy of optical positioning, dedicated targets or markers are used. Common types of targets include QR codes, barcodes or patterns like colored dots. They are placed at positions which are accessible easily by the phone's camera, e.g. at the floor. Depending on the approach, the pattern contains the positioning information directly, or it just represents a reference which is looked up in a position database.

Besides printed markers, there are scientific projects which use projected laser markers instead. In opposite to printed markers, which usually stand for one

Figure 15.5.: Projected Markers used by CLIPS[45]

position, the projected laser spots are interpreted to determine a more accurate position. The position determination is comparable with astronomic navigation, used by former seamen. Those systems are usually used in robotics environments, where it is no problem to have a camera pointing to the ceiling. The accuracy of this approach is quite impressive as it is reported to be better than 1 cm[43],[45]. It should be noted that those results depend on calibrated cameras, mounted to robotic vehicles. This results in more reliable image data compared to a pedestrian who uses a smart phone camera to capture the projected targets.

15.5. Near Field Communication

Near Field Communication (NFC) is a commonly used technology to provide location-based information. NFC itself is a set of standards to establish wireless communication between two devices which are in close proximity or even touching each other. The standard includes protocols and data exchange formats which are based on RFID.

One major advantage over other wireless technologies like Wi-Fi and Bluetooth is that one of the communication partners can be an unpowered NFC chip, often called *tag*. Those tags are pretty cheap and easy to install because they do not need any power supply or network connection.

The used data format for data exchange is the *NFC Data Exchange Format* (*NDEF*). The message structure is specified by the NFC Forum™ citeNDEFSpecification. The maximum payload size of an NDEF message is $2^{32} - 1$ bytes.

This specified maximum is not even nearly reached by commonly used NFC tags. Typically they are designed to hold less than 1 kilo byte of data. For this reason, the payload often only has an identifying or referencing role.

As the NFC reading device, e.g. an NFC capable smart phone, needs to touch an NFC tag in order to read it, NFC cannot be used for continuous position determination.

15.6. Roundup

Various different technologies and approaches for positioning, with a focus on indoor positioning, are introduced on the last pages. This section summarizes the results as a table.

Approach/Name	**Advantages**	**Disadvantages**	**Reported Accuracy**
		Satellite	
GPS	widespread and approved	only outdoors	2.8 meters[28]
		GSM	
Cell Identification	any phone has GSM	accuracy	50 meters in urban environments, worse in rural areas[30]
Angle of Arrival	any phone has GSM	modified GSM cell needed, location determination at the GSM cell not at the phone	150 meters[46]
Time of Arrival	any phone has GSM	time synchronization needed	125 to 200 meters[30]
RSSI	any phone has GSM	offline learning phase	30.05 meters in urban environments[30]
		Wi-Fi	
Angle of Arrival	any smart phone supports Wi-Fi	modified access points needed	7.5 meters[36]
Time of Arrival	any smart phone supports Wi-Fi, depending on the approach no hardware or driver modification needed	accurate systems depend on modified hardware	1.7 meters[37]
Fingerprints	any smart phone supports Wi-Fi, easy to implement	offline learning phase	2.5 to 7 meters[41]
		Optical	
Floor Plan	any smart phone has a camera, accuracy	computation time, easy to confuse	30 cm
Deployed or Projected Targets	any smart phone has a camera, high accuracy	visual intrusion into a room's look, non natural position of the phone	below 1 cm

Table 15.1.: Summary of Absolute Positioning Approaches

16. Relative Positioning

In opposite to absolute positioning, relative positioning is just able to determine position changes. On the assumption that an absolute position is known, new positions can be reckoned from this reference position and estimated position changes. This process, of calculation the current position by using a previously determined position and advancing that position upon sensor data such as speed and direction, is called *dead reckoning*.

Most smart phones offer a lot of sensors, such as accelerometers, magnetometers and gyroscopes, which can be used to determine position changes. Methods to detect steps and determine the user's heading based on those sensors are described in section 18.

The sensors used in relative positioning approaches differ a lot, depending on the sensor availability. Probably the most intelligible approach is to have the direction, provided by a compass and the traveling speed, provided by a speedometer. The traveled distance can easily be calculated as $speed * time$ and because the direction is known, the new position can be reckoned. This approach is widely used in marine and air navigation[47]. However, it becomes irrelevant due to the availability of satellite-based position such as GPS.

This chapter describes existing relative positioning approaches for robots and pedestrians.

16.1. Robot's Positioning

Besides air and marine navigation, dead reckoning is more interesting for indoor robot positioning today. Instead of a compass or a speedometer, dead reckoning in mobile robotics often makes use of engine data because many wheeled mobile robots are using a skid steering. Skid-steered vehicles do not have a moving

steering axle like cars. Instead, they have four or even more wheels at static axles, which can be moved independently from each other. Assuming that the rotation of the wheels is known, orientation and traveled distance can be calculated and used to determine a new position. An advantage of this approach is that no additional sensors are required, as all needed data can be tapped from the unit which controls the engines. A problem with such a kinematic approach is that calculations assume that a wheel never slips or crawls. Any slippage leads to inaccurate rotation and distance data[48].

In order to provide accurate dead reckoning, it is necessary to have kinematic independent sensors which are used to estimate the position. A project developed by Bonarini et al. is following a pretty interesting low cost approach, by using optical computer mice to measure movements of a vehicle. There is one mouse at the left and one mouse at the right side of the vehicle. Because the distance between the two mice is static, angles can be calculated from the measured traveled distance. With an average deviation of 114mm in UMBMark, it is about as accurate as other robots which are using complicated mechanical encoders for dead reckoning[49].

16.2. Pedestrian Positioning

As described in the sections above, dead reckoning depends on a direction and an estimation how far an object has been moved into that direction. Today's smart phones offer compass sensors, which can be used to estimate the moving direction, but there is no such sensor like a speedometer in a car. Therefore, the traveling speed has to be estimated by using the existing accelerometer.

An accelerometer measures the acceleration force on all three physical axes, including the force of gravity. A common approach is to analyze those data to make assumptions whether the user made a step respectively is walking[50],[51]. A detailed description about step detection and compass issues within buildings can be found in section 18.

17. Absolute Positioning in MoCaInfo

Chapter 15 describes various absolute positioning methods which are applicable for smart phones. This chapter starts with a short explanation, why certain absolute positioning approaches are not used for MoCaInfo's positioning system. After that, the usage of NFC for positioning is described briefly. The chapter ends with an introduction and evaluation of the Wi-Fi positioning system, which is used for continuous indoor positioning.

17.1. GSM

Based on the results of various researches[30],[31],[52], GSM seems to be a good approach to determine a position in buildings. Even though, it is not accurate enough for indoor navigation, it could be used to enhance the positioning results of other approaches like Wi-Fi-based positioning. GSM signals are available in most areas of buildings, and as smart phones are the targeted devices, a GSM module and SIM card can be presupposed.

In earlier tests with various Android devices, it has been noticed that many devices just deliver the currently associated GSM cell via the Android API[53, p. 13]. As explained in section 15.2.1, neighbor cells are needed for a reasonable location accuracy.

Using the angle between the mobile device and a GSM base station requires access to the base station. In case of indoor positioning, this means an own infrastructure of GSM femtocells is required. This approach became impracticable with a court decision in July 2013[54], interdicting the operation of private GSM cells in Germany. Because *Time of Arrival* also needs access to the whole GSM infrastructure, it cannot be used for the same reason.

17.2. Optical

The approaches described in section 15.4 are not used for positioning in MoCaInfo for various reasons. For a start, all optical approaches force the user to hold the smart phone or tablet in an unnatural way. This is a problem, especially for visually impaired users because they cannot know how to hold the device to capture a photo which is usable for positioning. In addition, the floor plan and the template matching method rely on a non-changing environment. This is not applicable for a university, where black boards and posters changing all the time and students are walking through the floor. Furthermore, an intrusion into the appearance of a building is not wanted. Therefore, the most promising approach of artificial landmarks cannot be adopted.

17.3. Global Positioning System

GPS receivers are integrated in most modern smart phones, and Android's location API supports their usage. Therefore, GPS is used for positioning whenever it is more accurate than other positioning approaches. This usually happens outside, where the Wi-Fi positioning system, described in section 17.5, is not available. A detailed explanation about the usage of multiple absolute location sources can be found in section 19.3.

17.4. Near Field Communication

With Android API level 9, better known as Android 2.3, Google introduced support for the NFC technology. This enables Android devices with an appropriate built-in NFC receiver to read and write NFC tags.

In API level 14, also known as Android 4.0, the Android *Beam* feature has been introduced, providing message exchange between two Android devices. Common use cases are exchanging contact information or sending short multimedia files like pictures or audio clips.

Figure 17.1.: Device Scanning a Passive NFC Tag

From a developer's point of view, the Android operating system handles the reading of NFC tags. The *Tag Dispatch System* is constantly looking for NFC tags when the device's screen is unlocked [55]. If the device discovers an NFC tag, the *Tag Dispatch System* searches the most appropriate activity to handle the intent created from the NFC tag's data. If there is more than one activity to handle the intent, the activity chooser appears. The desired behavior is to have only one activity for one kind of NFC tag, to prevent the activity chooser from appearing.

In MoCaInfo, NFC tags are one possibility of determining a user's position. For this purpose, points of interest, especially rooms are equipped with an NFC tag. If the user scans a tag, Android's tag dispatching system starts the MoCaInfo application, showing detailed information about the POI. The major advantage against any other localization approach is the accuracy. If the user's device discovers an NFC tag, the application can be sure that the user is at the POI's position.

17.5. Wi-Fi

Wi-Fi is a perfect indoor positioning technology for a university campus because wireless access points are widely distributed in any building, especially those with lecture halls.

Nevertheless, it should be noted that not any existing Wi-Fi infrastructure can be used for positioning. In case of Technische Hochschule Mittelhessen, access points are reconfiguring itself, depending on the number of connected clients, network usage and other device's on the same Wi-Fi channel. This means the sending signal strength and Wi-Fi channel change over time. This is a no-go for the used fingerprinting approach which relies on a non-changing configuration and environment.

For this reason, eight Wi-Fi access points for positioning have been installed at the first floor of building A20. The access points have been configured to send with the highest power and on a static channel. They are not used to transmit any data except the Wi-Fi beacons.

This section describes the developed Wi-Fi positioning system, including filters, fingerprint comparison methods and different ways to estimate the position. The section ends with an evaluation.

17.5.1. Filters

In order to achieve more accurate results, Wi-Fi measurements are filtered for certain properties. For example, a student's discoverable ad-hoc Wi-Fi network should not have an influence on fingerprint comparison because this would distort the results.

In MoCaInfo, there are two types of filters, *Measurement Filters*, which are applied to the current Wi-Fi measurement, and *Selection Filters*, which are applied to the fingerprints in the database.

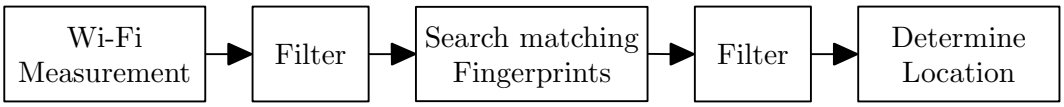


Figure 17.2.: Flow from Wi-Fi Measurement to Location Determination

17.5.1.1. Measurement Filters

A Wi-Fi measurement usually consists of a list of access points or rather their BSSID and SSID plus the respective RSSIs. Measurement filters are used to keep only measurements which are relevant for positioning. They are applied to the current measurement and to the fingerprint's measurements, loaded from the database.

Currently only three filters are in use, but more filters are imaginable to improve the accuracy or reduce computation time further.

SSID Filter: SSID is the short form of service set identifier. It can be seen as a human readable name of an access point, whereas multiple access points can have the same SSID. The SSID filter has a list of valid SSIDs, which are considered for position estimation. Access points with non-matching SSIDs are dismissed.

BSSID Filter: The BSSID, short form of basic service set identification, is a unique identifier of a mobile access points. It is equivalent to the MAC address of the Wi-Fi interface. The filter does pretty much the same as the SSID Filter, but for the BSSID.

MinRSSI Filter: As stated in [53] bad signal levels disturb the location determination. Therefore, the MinRSSI filter rejects all measurements below a certain signal level.

17.5.1.2. Selection Filters

Selection filters are used to reduce the amount of fingerprints which have to be compared by making a pre-assumption based on the current measurement. The implementation of those filters is simply a condition for the SQL where clause.

BSSID Filter: The BSSID filter takes care that only potentially fitting fingerprints are selected from the database. Fingerprints which do not have at least one BSSID in common with the current measurement are dismissed. Without this filter, computation time would increase proportionally with the amount of stored fingerprints.

Compass Filter: One of the approaches to improve positioning accuracy is to store measurements of multiple orientations of a location. This considers the fact that the human body reduces the signal strength if it is between the measuring device and the access point. Furthermore, other influencing factors like reflections or obstacles in line of sight might depend on the user's orientation.

Different researches show that regardless of the comparison and estimation algorithms, accuracy enhances when multiple orientations are considered[39],[41],[56].

Figure 17.3 illustrates the compass filter's principle. Every grey dot represents a fingerprint location, whereas the grey arrows stand for the different orientations of measurements. The black dot represents the user and his orientation. The compass filter is looking for a orientation range, in case of the illustration the range's width is about 90°. The black arrows represent the fingerprints, finally used for position estimation.

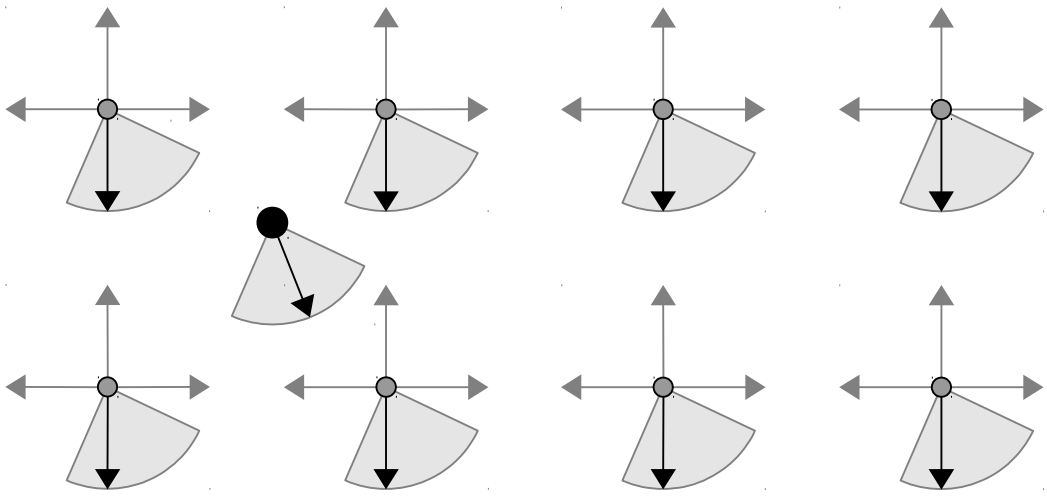


Figure 17.3.: Compass Filter's Functional Principle

17.5.2. Fingerprint Comparison

In order to determine a location, based on the currently measured signal strengths, the current measurement has to be compared to the previously recorded fingerprints.

In current research, two different approaches have been emerged.

17.5.2.1. Deterministic Comparison: RSSI Distance

The first approach is usually called the *deterministic approach* or *distance approach*. To compare the fingerprints, a distance between the RSSIs is calculated. A low distance indicates that the current location is similar to the location of the compared fingerprint.

Different formulas to compute the distance are described in literature[41],[57]. Their main difference is whether and how they amplify big differences or lower small differences. As scientific comparisons of the different distance methods are rare, the following equations are implemented and tested in MoCaInfo. The evaluation can be found in section 17.5.4.1.

$$D_{Euclid}(x,y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{17.1}$$

$$D_{Manhattan}(x,y) = \sum_{i=1}^{N}|x_i - y_i| \tag{17.2}$$

$$D_{BrayCurtis}(x,y) = \sum_{i=1}^{N}\frac{|x_i - y_i|}{x_i + y_i} \tag{17.3}$$

$$D_{Canbarra}(x,y) = \sum_{i=1}^{N}\frac{(x_i - y_i)^2}{x_i + y_i} \tag{17.4}$$

The equations 17.1 to 17.4 can be read as follows. The distance D between two fingerprints x and y is computed. A fingerprint is a vector which holds the signal strength of a measurement, whereas x_i and y_i are the signal strengths for the same access point. N is the number of signal strengths, which has to be equal for x and y. If fingerprint x includes the signal strength of an access

point which is not included in y, a placeholder value is inserted in y and vice versa. This placeholder value represents a pretty bad signal strength.

17.5.2.2. Probabilistic Approach: Naïve Bayes Classifier

Besides the deterministic distance calculation, probabilistic approaches are used in many research projects with promising results[41],[56],[58],[59].

The probabilistic approach, tested within MoCaInfo is called naïve Bayes classifier and is based on the Bayes theorem, which defines the probability P of the class C under the assumption that x is given as follows:

$$P(C|x) = \frac{P(C)P(x|C)}{P(x)} \tag{17.5}$$

The different elements of the formula are denoted as follows:

$$\text{posterior} = \frac{\text{prior} * \text{likelihood}}{\text{evidence}} \tag{17.6}$$

In case of fingerprinting, $P(C|x)$ describes the probability that fingerprint x belongs to the class C which represents a position. x is a vector of RSSI values.

The uniqueness of the naïve classifier, compared to other classifiers, is the assumption that all values of the input vector x are independent of each other. For this reason, it is easy to calculate the conditional probability $P(x|C)$, which is the product of the probability of each element in x given class C.

$$P(x|C) = \prod_i P(x_i|C) \tag{17.7}$$

The likelihood $P(x|C)$ depends on the training data, whereas there are several possibilities to compute it. A common approach[41],[60, p. 36] which is used in MoCaInfo is the following:

$$P(x_i|C) = \frac{1}{n}\sum_{j=1}^{n} K_{Gauss}(x_i, y_j) \tag{17.8}$$

$$K_{Gauss} = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-y)^2}{2\sigma^2}\right) \tag{17.9}$$

K denotes the kernel function. x is the observed fingerprint and y are all fingerprints, recorded for location C. n is the number of recorded fingerprints for location C.

As it is assumed that all fingerprints have the same probability, $P(x)$ is assumed to be 1. Therefore, it can be omitted in the equation:

$$P(C|x) = P(x|C)P(C) \tag{17.10}$$

The computation of $P(C)$ has again, multiple possible implementations. In case of this thesis, two cases are implemented and tested. The first implementation assumes that all locations are equally probable at any time, which results in $P(C) = 1$. Another approach considers the distance between C and the last estimated location L as an indicator for the probability of C. A short distance between C and L results in a high probability, and a long distance corresponds to a low probability. All locations within a three meter radius of the last estimated location are considered to be equally probable. Fingerprints further away than three meters, are considered to be less probable. Figure 17.4 illustrates this approach.

$$P(C) = \frac{3}{max(3, \sqrt{(C_x - L_x)^2 + (C_y - L_y)^2})} \tag{17.11}$$

17.5.3. Position Determination

The results of fingerprint comparison are used to determine the actual position. In this work, two widely-used approaches are described and compared.

Figure 17.4.: Distribution of $P(C)$ with a Fingerprint Distance of Two Meters

17.5.3.1. Best Fit

The easiest approach to determine the user's position is to assume that he currently is at the best fitting fingerprint's position. In case of the *distance approach* described in section 17.5.2.1, this is the fingerprint with the lowest distance, formalized with equation 17.12. For probability-based approaches like the *naïve Bayes classifier*, the best fitting fingerprint is the one with the highest probability, see equation 17.13.

$$\text{choose } y_i \text{ if } D(x, y_i) = \min_k D(x, y_k) \tag{17.12}$$

$$\text{choose } C_i \text{ if } P(C_i|x) = \max_k P(C_k|x) \tag{17.13}$$

17.5.3.2. Weighted Nearest Neighbor

A more sophisticated approach is the nearest neighbor method, often called *K-nearest neighbor* or just *KNN*. Instead of using the best fitting fingerprint, the K best fingerprints are used to interpolate a position.

In order to consider the different fittings, a *weighted* nearest neighbor is implemented for MoCaInfo's positioning. The algorithm makes sure that a fingerprint with a lower RSSI distance, respectively a higher probability, has a stronger impact on the resulting position than a fingerprint with a higher distance, respectively lower probability. To avoid overweighting of a position, only the best fingerprint for one position is taken into account. Otherwise, positions

with a larger amount of measured fingerprints might impact the interpolated position stronger than positions with fewer fingerprints.

Equation 17.14 shows the implementation of the weighted nearest neighbor algorithm, used together with the naïve Bayes approach. C is a vector of locations, sorted by the probability $P(C|x)$ of each location C, beginning with the highest probability.

$$C_{KNN} = \frac{\prod_{i=1}^{k} C_i * P(C_i|x)}{\sum_{i=1}^{k} P(C_i|x)} \tag{17.14}$$

17.5.4. Evaluation

The following section compares the different approaches and filters to estimate the best combination for accurate Wi-Fi indoor positioning. For that purpose, a test environment at the first floor of building A20 has been created. Wi-Fi fingerprints at 67 different locations have been recorded. The fingerprint locations are distributed as equally as possible with a distance of two meters. In this way, an area of about 280 m^2 is covered. For any location, four orientations have been measured. Three fingerprints for each orientation, resulting in an overall amount of 804 fingerprints.

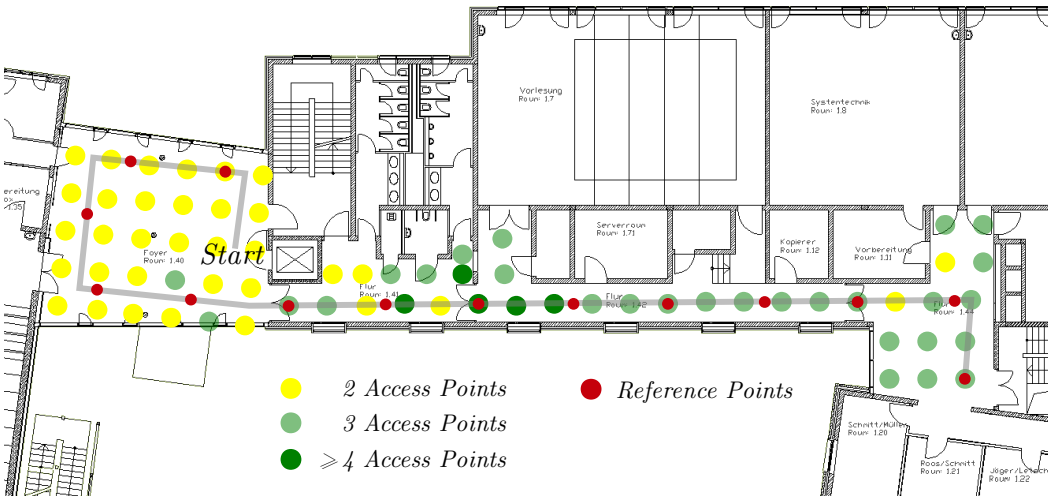


Figure 17.5.: Wi-Fi Positioning Test Area with Fingerprints

Eight wireless access points, with a non-changing configuration, are available for positioning. Even though about one quarter of all fingerprints only contain the data of two access points, whereas it is known that the accuracy increases with the number of access points.

For the evaluation, a track of 70 meters has been walked in various speeds, with different devices and in different directions. At the track, 14 reference positions have been marked. Those known reference positions are compared with the estimated positions, to determine the accuracy of the different approaches. In order to get representive test results, the test track has been walked with three different smart phones: a Nexus 4, a Nexus 7 and a Samsung Galaxy S3. Figure 17.5 shows the test environment, including the test track which is illustrated by a grey line.

17.5.4.1. Comparison of Distance Methods

At first, the four distance methods are compared. Thus, the positions of all runs are estimated, using best fit and nearest neighbor with three neighbors, for each distance method.

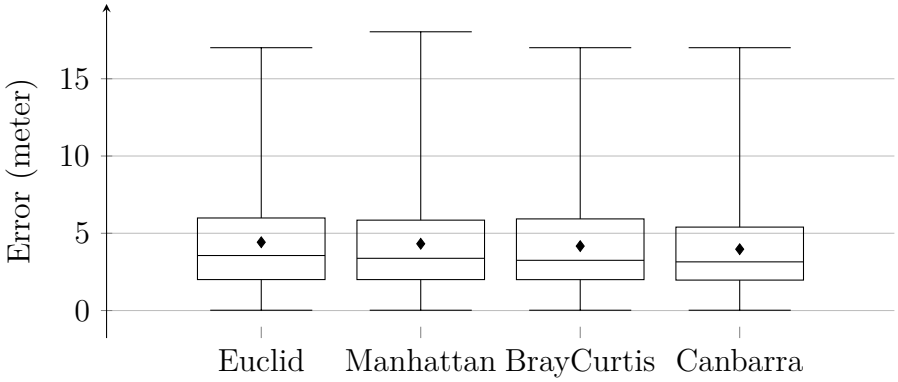


Figure 17.6.: Comparison of Various Distance Methods as a Box Plot

The box plot in figure 17.6 visualizes the result. The bottom and top lines represent the minimum, respectively the maximum error. The box in the middle represents 50% of the results. The line in the middle of the box illustrates the median error and the diamond shows the average value. No significant difference in accuracy could be determined. The mean error is between 3.97 and 4.42 meters, depending on the distance method. The most interesting finding is that

the widely-used *Euclidean distance* performs the worst. Furthermore, the *Canbarra distance* performs best, which is surprising because no other publication has been found which uses the *Canbarra distance* for RSSI comparison.

17.5.4.2. Naïve Bayes: Distribution of P(C)

After evaluating the distance methods, the same evaluation is done for the naïve Bayes method. As explained in section 17.5.2.2 the probability that a location is estimated by the naïve Bayes algorithm $P(C)$ can either be assumed to be equal for all locations, or depend on the distance to the last estimated location.

In order to get an idea, whether the impact of $P(C)$ differs between *best fit* and *nearest neighbor*, both approaches are compared in the box plot of figure 17.7.

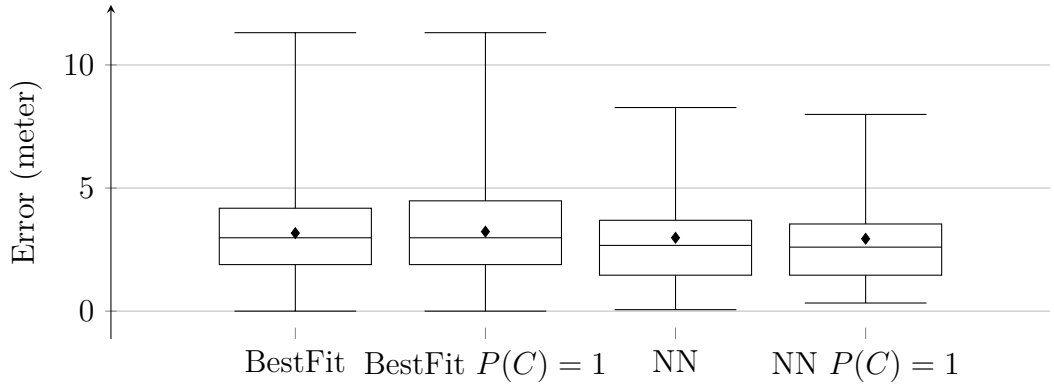


Figure 17.7.: Impact of a Last Location Depended $P(C)$ on the Accuracy as a Box Plot

The results are quite interesting. First of all, the naïve Bayes approach performs better than the distance approach. An average error of 2.98 to 3.23 meters compared to 3.97 meters with the Canberra distance is significant. As one could expect, considering the last estimated location with $P(C)$ has a positive effect together with the best fit approach. Surprisingly it has a slightly negative effect combined with the nearest neighbor interpolation. This is mainly because larger errors, for example, at locations with few access points, have a negative influence on the accuracy of future positions.

Another negative effect of $P(C)$, considering the last estimated location, is an increased computation time. Therefore, it is highly recommended to assume an equally distributed $P(C)$ together with nearest neighbor interpolation.

17.5.4.3. Weighted Nearest Neighbor: Number of Neighbors

The number of neighbors used for the position interpolation is variable. The following evaluation shows the impact of this number on the positioning accuracy. The tests are made in combination with the naïve Bayes approach.

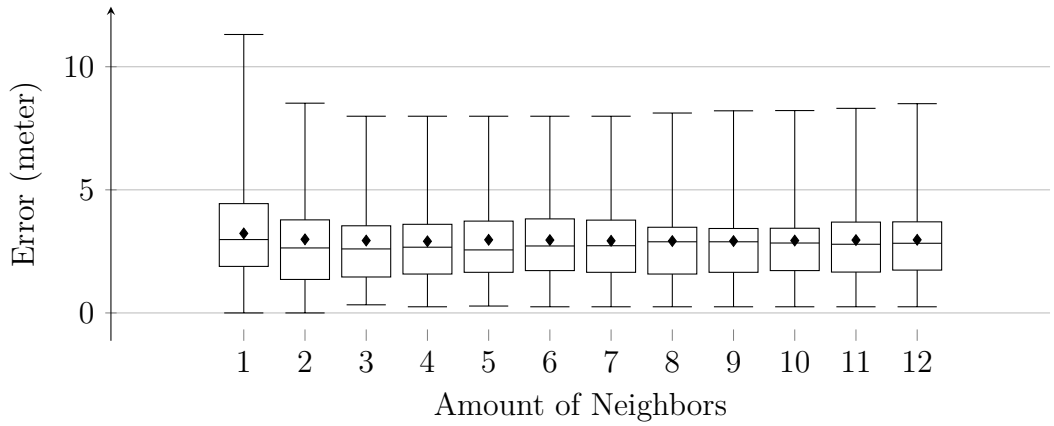


Figure 17.8.: Impact of the Number of Neighbors in Weighted Nearest Neighbor Algorithm on Accuracy

Figure 17.8 again shows the superiority of the *nearest neighbor method* against *best fit*, which is equivalent to *nearest neighbor* with just one neighbor. It is hard to point out the perfect amount of neighbors because nearly any amount of neighbors has its own advantages. Anyways, an amount of three neighbors leads to the second best average error, combined with a good lower and upper quartile. Furthermore, a lower amount of neighbors reduces the computation time. Therefore, three neighbors are used for all following KNN computations, unless otherwise stated.

The often mentioned problem of overfitting[61],[62], which basically means that the interpolation gets significantly worse from a certain amount of neighbors, cannot be discovered in this tests. The reason for this can be seen in the strict weighting. For this reason, locations with lower probabilities only affect the resulting location marginally.

17.5.4.4. Distance vs. Naïve Bayes

The last subsections evaluated that the *Canberra distance* is the best distance approach. Furthermore, the naïve Bayes classifier works best assuming that all locations are equally probable. The KNN method turned out to be the more accurate than the best fit approach, whereas the interpolation works best with an amount of three neighbors.

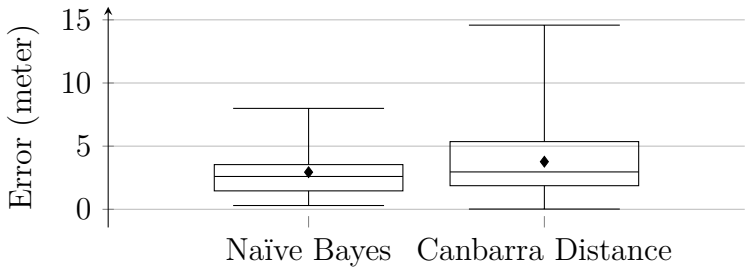


Figure 17.9.: Estimated Positions with Naïve Bayes Classifier and Canberra Distance

The box chart in figure 17.9 compares the accuracy of the distance and the naïve Bayes approach. Both are using a KNN with three neighbors. The average accuracy of 2.94 meters, achieved by the naïve Bayes classifier is about 28% better than the average error occurred using the Canberra distance. Even more impressive is the difference of the maximum error, which is about 6.59 meters better when using the naïve Bayes classifier.

Due to this unambiguous result, the *naïve Bayes classifier* in combination with the *nearest neighbor interpolation* is chosen for MoCaInfo's Wi-Fi positioning component. Figure 17.10 shows the estimated track of both approaches.

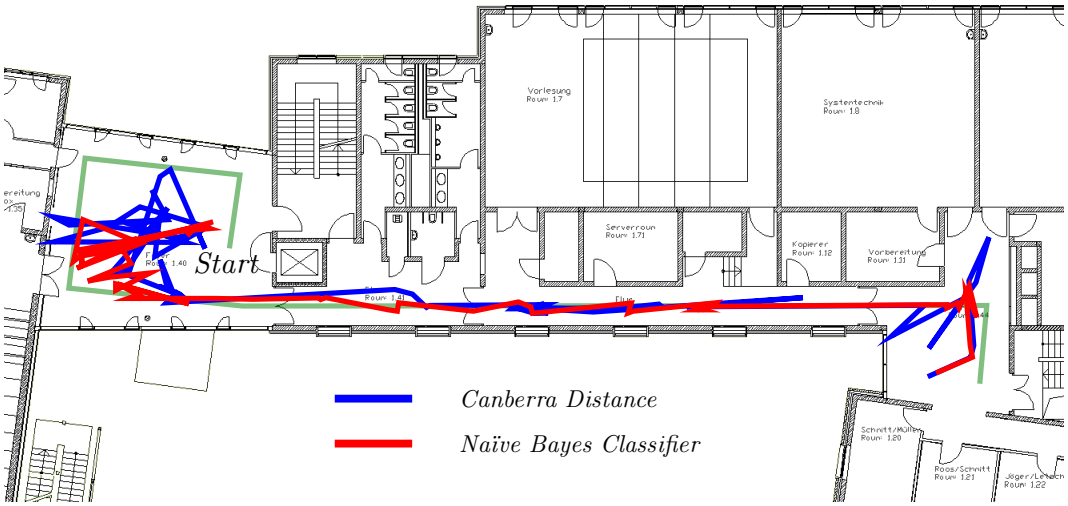


Figure 17.10.: Map Showing Two Tracks, Estimated with Canberra Distance Approach and Naïve Bayes classifier.

18. Relative Positioning in MoCaInfo

The following sections describe the relative positioning system developed for MoCaInfo. In order to achieve a pedestrian relative positioning system, steps have to be detected, which is explained in the section 18.1. The subsequent section describes different approaches to implement a compass, using a smart phone's sensors. This is needed in order to determine the walking direction.

18.1. Step Detection

In order to achieve an accurate dead reckoning system, the location software has to notice, whether a user is moving or not. For pedestrians this means, it has to be recognized whether the pedestrian is walking or not. Commonly used accelerometer-based approaches to detect steps use sensors, placed at the pedestrian's foot[63], [64]. With those foot-mounted sensors, it is even possible to detect the roll of the foot while walking. Kim et al.[63] developed a precise system, based on this data with a 100% accuracy in step detection and an error of less than 5% in step stride determination.

The approach of roll-over detection cannot be used in MoCaInfo's positioning system because it has to be independent of any external sensors except the built-in smart phone sensors. Ying et al. had a similar requirement since they used accelerometers of medical implants to detect steps. They implemented and compared three different algorithms for step detection. The first one is the *Pan-Tompkins* method, a real-time algorithm usually used for peak detection in ECG signals. Secondly, they tried to recognize steps, with a *template matching* approach, assuming that the signal of a step can be determined by comparing it with sample signals of steps. The last algorithm is a simple *peak detection* algorithm. They recommend peak detection because it is as accurate as the Pan-Tompkins method but has less computational cost. Template

matching was not as accurate as the other methods[65]. This concurs with the approach of the FootPath project[50], which uses an Android device with a peak detection algorithm to detect steps.

In the context of this thesis, the FootPath algorithm has been integrated into a test application, to approve the promising results stated by Link et al.[50]. Unfortunately, the results at the Android test device were less accurate than expected, having an error rate above 20%. It should be noted that Link et al. do not mention any error rate for the step detection itself, but their overall positioning accuracy of 1.6 meters is impressive. The step detection errors in their project are partially compensated by a path matching algorithm which knows the path the user walks in advance. As the MoCaInfo positioning system cannot draw on navigational paths, this approach cannot be adopted.

A novel step detection algorithm is implemented, combining ideas of the Pan-Tompkins-method and the FootPath peak detection algorithm because of those disappointing test results.

18.1.1. Signal Preparation

The signal is prepared for the peak detection based on methods used by Pan-Tompkins. At first, the z-axis accelerometer data is smoothed by a low-pass filter. For a more detailed explanation of this filter, see section 18.2.2.1.

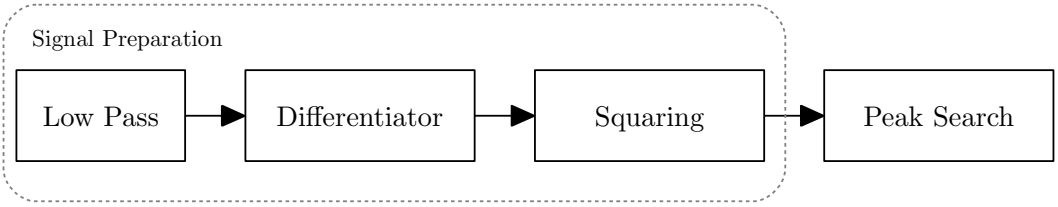


Figure 18.1.: Steps of Signal Preparation

A derivative operator uses the low-pass filtered values to suppress low-frequency components and enlarge the high frequency components from the high slopes. Equation 18.1.1 is a slightly modified version of the equation explained in [65]. The modification was necessary since the measurement frequency in Ying's

work was higher than suitable for Android devices. Furthermore, all negative $y(n)$ are set to zero because negative values disturb the following peak detection. $x(n)$ represents the low-pass filtered acceleration at the z-axis of measurement number n.

$$y(n) = \begin{cases} \frac{1}{4}[2x(n) + x(n-1) - x(n-3) - 2x(n-4)] & \text{if y(n)} > 0 \\ 0 & \text{else} \end{cases} \tag{18.1}$$

The derivative operator's output is squared in order to emphasize high values additionally, whereas the following equation is used. It makes sure that values below 1 are not further lowered.

$$y(n) = (1 + y(n))^2 - 1 \tag{18.2}$$

Figure 18.2 shows a sample signal of five steps, whereas the user was standing for about one second before he began to walk. Every single step produces an amplitude. The low-pass filter (b) smooths the signal which reduces the disturbances between the steps. As a result of the derivative operator, realized by implementing equation 18.1.1, disturbances are filtered out (c). The final squaring just emphasizes this result (d).

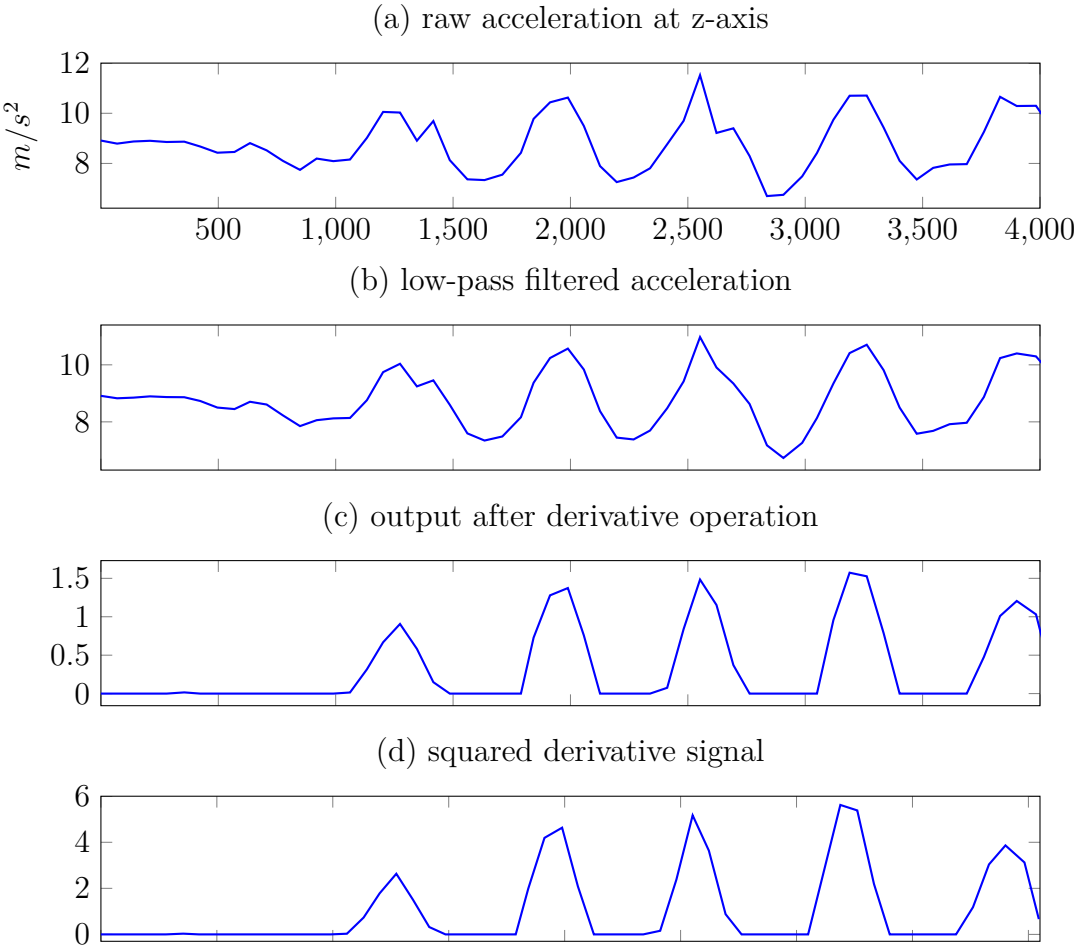


Figure 18.2.: Acceleration Measurements after Various Enhancement Steps

18.1.2. Peak Detection

The actual peak detection is based on the algorithm of the FootPath project. Just a few constant values, such as the threshold and the inspected time span, have been adjusted after empirical analyzes.

The algorithm detects the peaks of the prepared signal. Therefore, it examines the relations between the currently measured value (from now on *current)*, the value before *current* (from now on *predecessor*) and the first value of the examined region (from now on *oldest*). This region is marked light blue in figure 18.3 and the analyzed values are marked with red squares. A region which covers a time span of about 200ms, or four measurements, has been determined empirically, to deliver the most accurate results. First of all, the algorithm tests

whether *current* is less than *predecessor*. If so, *predecessor* is a potential peak. To make sure that it is a peak, *oldest* has to be less than *predecessor* as well. If this is also true, the difference between *oldest* and *predecessor* is computed and checked against a threshold. If the difference is greater than the threshold, *predecessor* is estimated to be a peak and therefore a step. A threshold of 1 leads to the best results in various test walks.

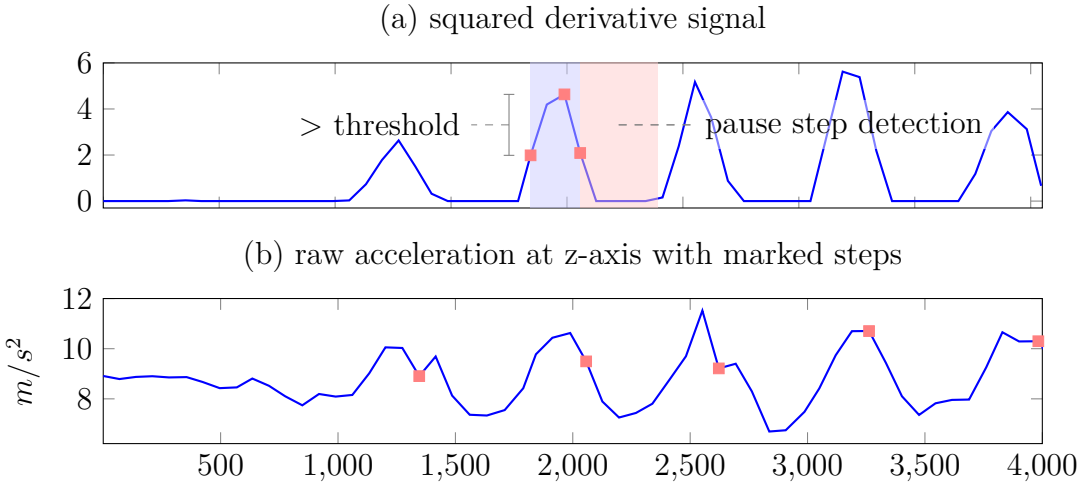


Figure 18.3.: Step Detection Example. Red Squares in Figure (b) Represent Detected Steps

If a step has been detected, the step detection pauses for 330ms to prevent a double detection of one step. This timeframe is shown by a light red area in figure 18.3 (a). The red squares in the image below illustrate the detected steps.

The overall process of step detection is also illustrated by figure 18.4, which might be easier to understand than the previous textual description.

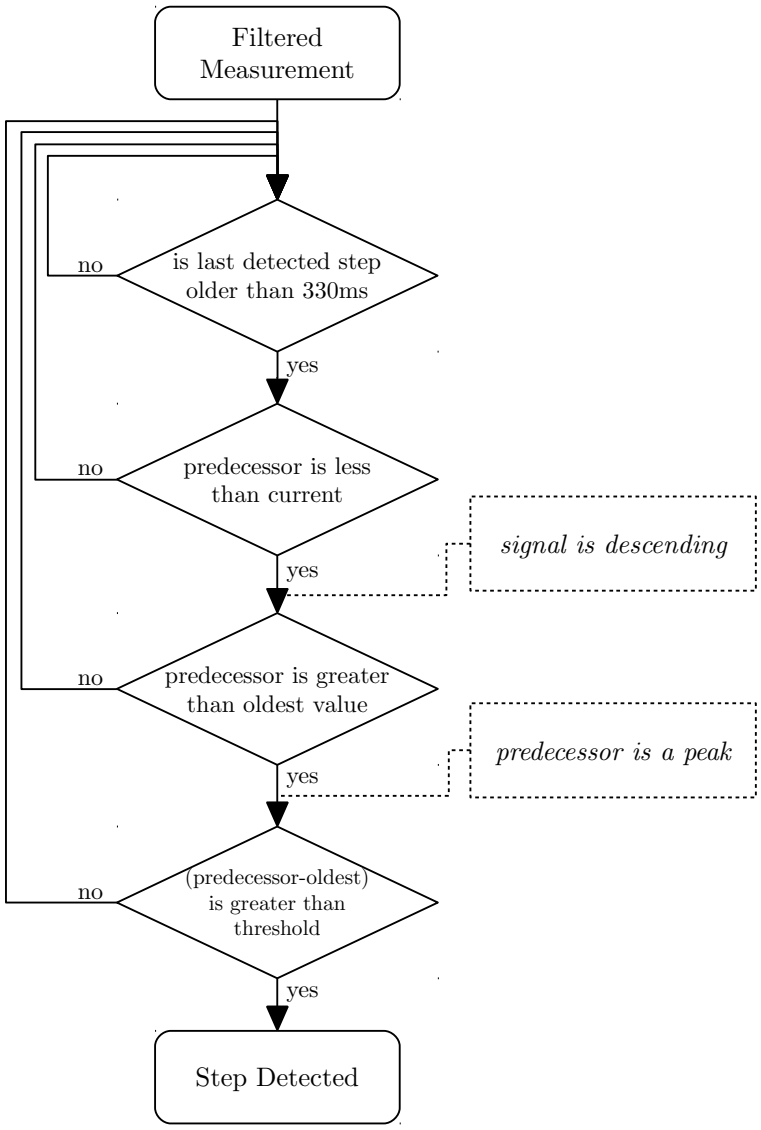


Figure 18.4.: Procedure of Peak Detection

18.1.3. Evaluation

The main purpose of the evaluation is to find out, if the signal preparation has a positive effect on the step detection's accuracy or not. Furthermore, it reveals whether the gathered accelerometer measurements differ significantly depending on the user who walks.

Therefore, five test persons with different genders, ages and body heights have walked the same test track. One of those testers has a low vision. He walked

two test tracks. The first one is the same which was walked by the other test persons. It covers an area he knows quite well. The other test track is in an area he only knows poorly. Ulterior motive for those two different areas was the presumption that blind and visually impaired people walk more carefully and uncertainly in unfamiliar areas, which might affect the step detection. Doors and other barriers had to be handled by the testers themselves.

The test with a visually impaired user shall clarify, whether other algorithms are needed for this user group or if their steps can be detected using the existing approach.

The first test run of the visually impaired test person, was noticeable worse than the tests with sighted people before. The reason for this, were movements of his left hand which held the smart phone. It should be noted that the test person usually does not use smart phones and was unfamiliar with its handling. After a short introduction and an explanation why it is important to hold the smart phone steady, the repeated test run and all following tests were comparable to the test runs with sighted people. Due to these irregularities in the first test run, it has not been taken into account for the evaluation. Even the unfamiliar area had no impact on the step detection itself, whereas it should be noted that the average step stride in this area was lower than in the other test runs. As the step detection does not cover stride detection at the current state of development, this observation is unimportant at the moment. Altogether, the tests revealed that no adjusted step detection algorithm for visually impaired people is needed.

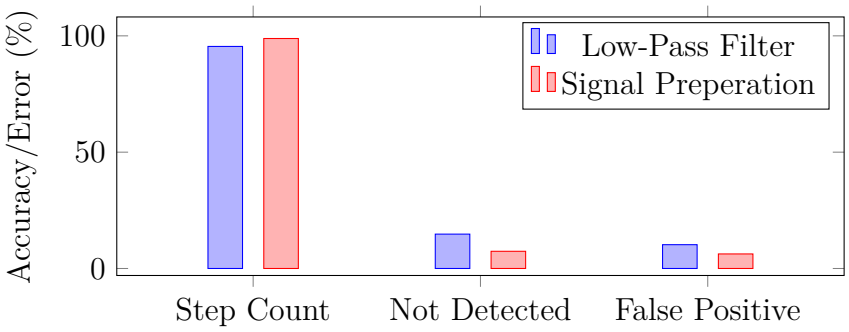


Figure 18.5.: Bar Chart Comparing the Step Detection Accuracy

The evaluation shows that the step detection in general is reasonably accurate for relative positioning. The overall step count is 98.86% accurate with the prepared signal and 95.45% accurate with a low-pass filtered signal. The low-pass filtered approach is equivalent to the FootPath approach[50]. Certainly, the overall step count is not a good indicator for the actual error, which is slightly higher. For the prepared signal, 7.4% of the steps have not been recognized, and 6.25% of the detected steps are false positive phantom steps. In case of the low-pass filtered signal, 14.8% of the steps are not detected, and 10.2% are false positive detections.

Both error classes compensate each other to some degree, especially if a false positive detection occurs shortly after an undetected step or vice versa. However, this compensation does not work, if the users changes his direction in the meantime.

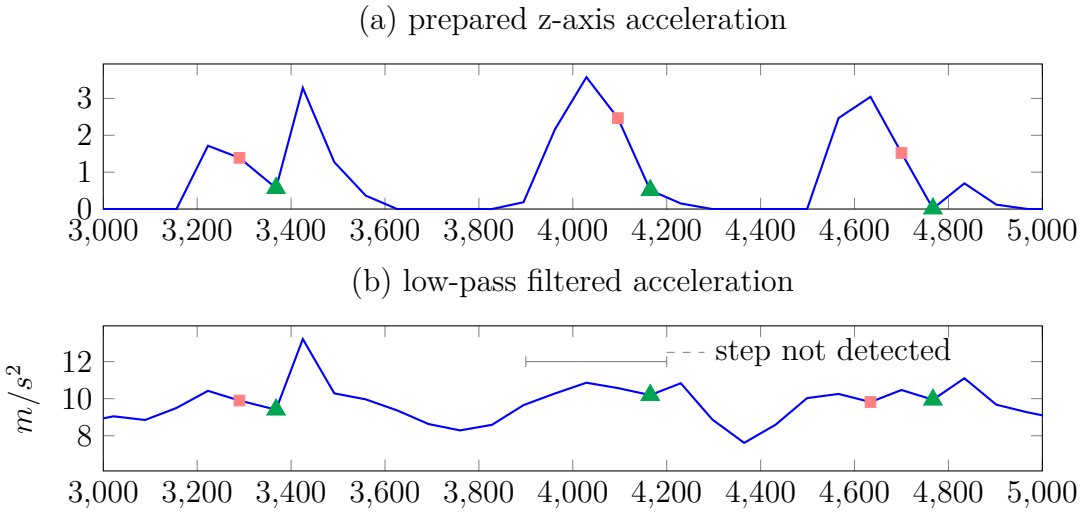


Figure 18.6.: Comparison of Prepared and Low-Pass Filtered Signal

Figure 18.6 shows acceleration measurements gathered in one of the test runs. The red boxes represent a step detection, the green triangles represent the approximate time when the foot hits the ground. The figure illustrates why the signal preparation explained in section 18.1.1 leads to a superior step detection. The signal in the figure has been captured while walking draggingly, which means that the feet barely left the floor. The peak at the 4th second is not distinct enough in the low-pass filtered signal, and therefore can not be

recognized as a step. The prepared signal shows a distinguished peak, which has correctly been interpreted as a step.

In conclusion, it can be stated that steps of visually impaired people can be detected as good as steps of sighted people. Furthermore, the number of undetected steps can be halved with the signal preparation. In addition, the number of phantom steps is reduced by a third. For this reason, the prepared signal is used for MoCaInfo's step detection component.

18.2. Compass

In addition to step detection, the user's orientation is needed in order to estimate relative position changes. The following section describes the calculation of the azimuth, using the earth's gravity and magnetic field. Since previous project members[53, p. 34] and other researchers[66, p. 35] experienced problems caused by magnetic interference and sensor inaccuracy, filters and alternative approaches are introduced and evaluated, trying to minimize the influence of those confounding factors.

18.2.1. Azimuth Calculation

The magnetic azimuth, which describes the bearing of a compass, can be calculated by using the accelerometer and the magnetic field sensor. The first sensor captures the gravity acceleration force ($\vec{g}$) in all three directions. The second sensor measures the magnetic field ($\vec{m}$), also at all three axis.

The following describes the *vector cross product method*[67] to determine the azimuth, based on these two vectors. In the first place, the vector product, also known as cross product, of gravity and magnetic vector is calculated. The resulting vector is the perpendicular of the plane, composed of gravity and magnetic vector. This normal describes the *east direction* of the navigation frame. The *north direction* is calculated through the cross product of the *east direction* and $\vec{g}$. Finally both vectors are getting normalized.

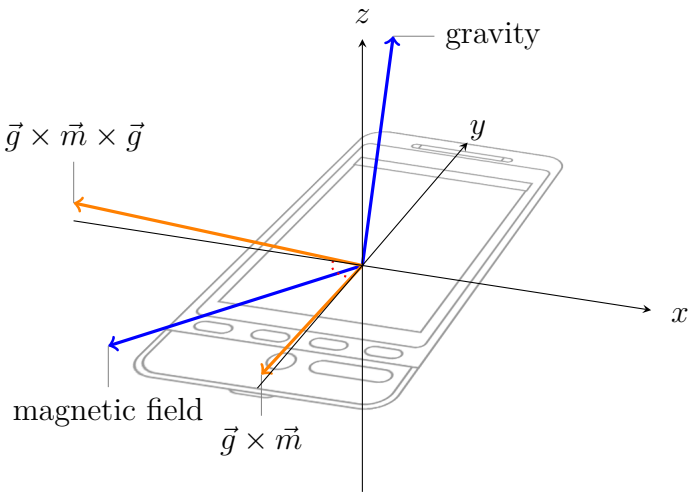


Figure 18.7.: Normalized Vectors, used for Azimuth Calculation. Example Vectors result in an Azimuth of approximately 90°

$$\text{East Direction: } \vec{E} = \vec{g} \times \vec{m} \tag{18.3}$$

$$\text{North Direction: } \vec{N} = \vec{g} \times \vec{m} \times \vec{g} \tag{18.4}$$

$$\text{Normalize: } \vec{E'} = \frac{\vec{E}}{|\vec{E}|}, \quad \vec{N'} = \frac{\vec{N}}{|\vec{N}|} \tag{18.5}$$

The azimuth (α)is the angle between the y values of the two normalized vectors, whereas some case differentiation needs to be regarded, in order to return the appropriate quadrant of the computed angle. This is the reason for the usage of $arctan2$ instead of $arctan$.

$$\alpha = arctan2(E'_y, N'_y) \tag{18.6}$$

To enhance the accuracy of the compass, a reference gravity vector $\vec{g}_0$ and a reference magnetic vector $\vec{m}_0$ are needed. Those reference vectors are known

to be correct, and not influenced by external forces, such as electromagnetic fields.

$$\text{Reference Direction: } \vec{R} = \vec{g}_0 \times \vec{m}_0 \times \vec{g}_0 \tag{18.7}$$

$$\alpha = arctan2(E'_y * R_y, N'_y * R_y) \tag{18.8}$$

Except the calibration, all explained calculations are implemented in Android's `SensorManager` to simplify the calculation of the compass bearing for developers.

18.2.2. Filter

The accuracy of a pedestrian dead reckoning system depends on the compass' and step detection's accuracy. The accuracy of the compass is an issue, particularly in indoor environments. The reasons of this inaccuracy in buildings are electromagnetic fields, which distort the magnetic field vector, measured by the phone's sensor. Sources of those electromagnetic fields in buildings are power supply lines and electrical devices. Furthermore, ferromagnetic objects, such as heaters, pipes or steel beams distort the azimuth estimation. Those objects are either magnetized by design or get magnetized by the earth's magnetic field or nearby electromagnetic fields.

Tests measurements, which simulated walks inside buildings, show that those confounding factors often just appear over a short period of time, e.g. if one is passing a heater. This suggests that filtering methods might improve the breakdown susceptibility.

The following subsections describe different filters ending with a comparison in section 18.2.4.

18.2.2.1. Low-Pass Filter

A low-pass filter passes low-frequency signals and reduces the amplitude of signals with a frequency higher than the cutoff frequency. As a consequence,

disturbances caused by sensor inaccuracies or external confounding factors are lowered.

$$x_{new} = x_{old} * \alpha + x_{new} * (1 - \alpha), \forall \alpha \in \mathbb{R} | \alpha \geq 0 \wedge \alpha \leq 1 \tag{18.9}$$

Equation 18.9 shows a low-pass implementation, whereas α is the smoothing factor. Android's API documentation recommends calculating α as described in equation 18.10[68]. The formula considers the measurement delivery rate Δt and a time-constant t.

$$\alpha = \frac{t}{t + \Delta t} \tag{18.10}$$

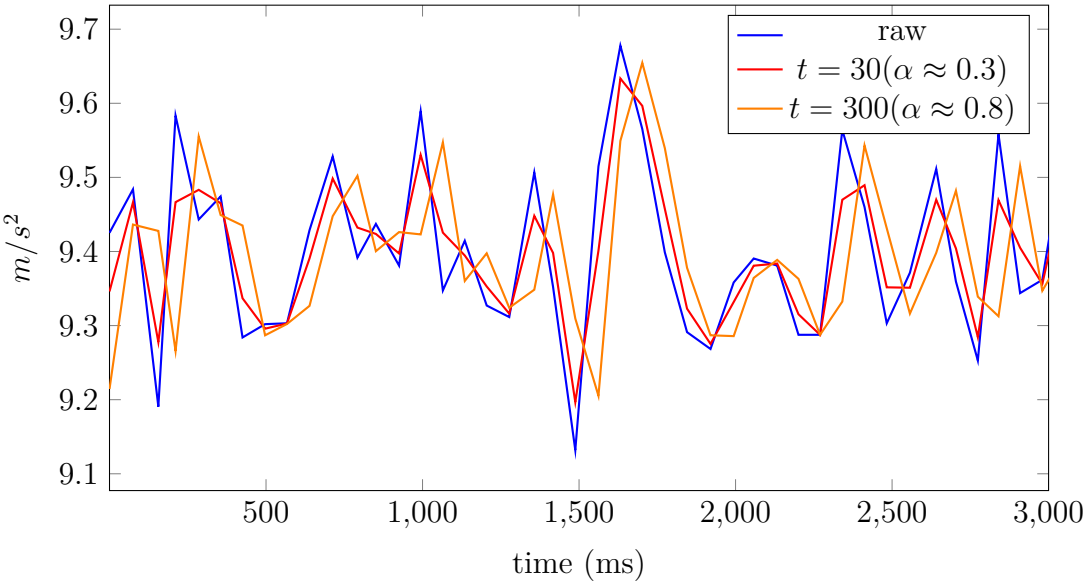


Figure 18.8.: Low-Pass Filter Applied to Z-Axis of an Acceleration Vector over Time

Figure 18.8 shows the decreasing amplitude when a low-pass filter is applied to the raw measurements, whereas a lower value of α leads to a lower amplitude. Furthermore, the chart shows a shift at the time-axis which increases with a bigger value for α.

18.2.2.2. Exponential Moving Average

Exponential moving average (EMA) can be seen as an extension of the low-pass filter. Instead of only having the last and the current sensor data, EMA considers a stack of past values. In equation 18.11, this stack is represented by the indexed variable x, whereas newer values have a lower index. n represents the number of values which are considered for the computation.[69, chapter 6.4.2]

$$x_0 = \frac{x_{new} + \sum_{k=1}^{n}((1-\alpha)^k * x_k)}{\sum_{k=0}^{n}(1-\alpha)^k} \tag{18.11}$$

Similar to the low-pass filter, α is a smoothing factor. The particular trait of EMA, compared to other average calculations, is the exponential decrease of weights. For that reason, the second value, which is newer, has a stronger impact on the result, than the older third value. Figure 18.9 shows the decreasing weights of an EMA.

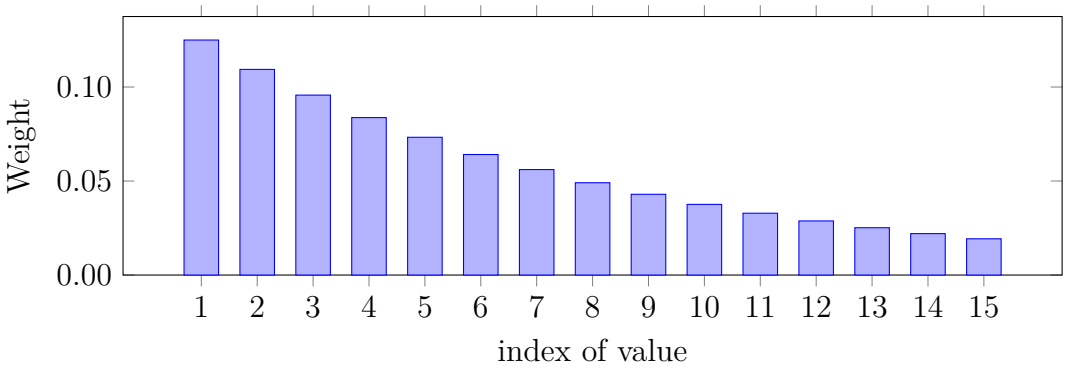


Figure 18.9.: Decreasing Weights of an EMA with $\alpha = 0.125$

Similar to figure 18.8, figure 18.10 illustrates the raw z-axis acceleration for a time frame of three seconds. The EMA filtered values are a lot flatter than the raw values. The amplitudes decrease slightly with the number of elements n which are considered for the computation. Even though, it should be noted that the difference between 5 and 50 values on the EMA stack is not significant because the older values affect the result decreasingly.

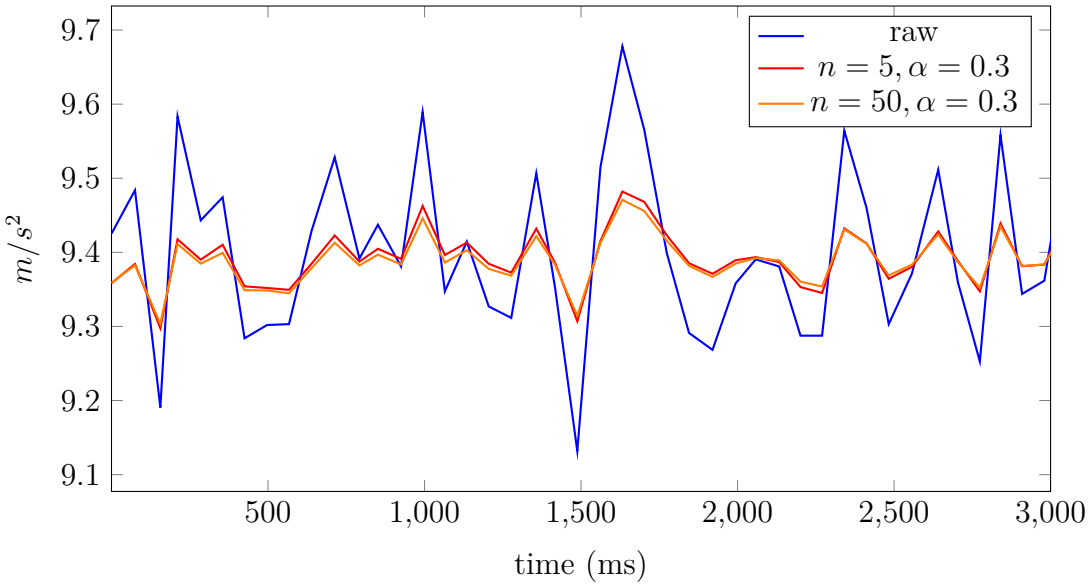


Figure 18.10.: EMA Filter Applied to Z-Axis of an Acceleration Vector over Time

18.2.3. Sensor Fusion with Gyroscope

Another approach to reduce the influence of magnetic interference is the usage of a gyroscope[70][71, p. 54ff]. A gyroscope is another sensor which can be found in many Android devices. It measures the angular speed at the x-,y- and z-axis in radians per second.

Figure 18.11 illustrates the three independent sensors, used to calculate a fused orientation vector. Accelerometer and magnetometer are used to calculate a basic orientation as described in section 18.2.1. This orientation is called *AccMag Orientation* in the figure. In parallel a *Gyro Orientation* is calculated, only based on the gyroscope measurements and the time between those measurements. The low-passed filtered *AccMag Orientation* and the high-passed filtered *Gyro Orientation* are combined to a *Fused Orientation*, which is less susceptible to magnetic disturbances.

To compute a fused orientation, several steps are necessary. First of all, the gyroscope matrix G is initialized as an identity matrix. Secondly, a rotation matrix R is calculated, based on the orientation vector $\vec{o}$ which is determined,

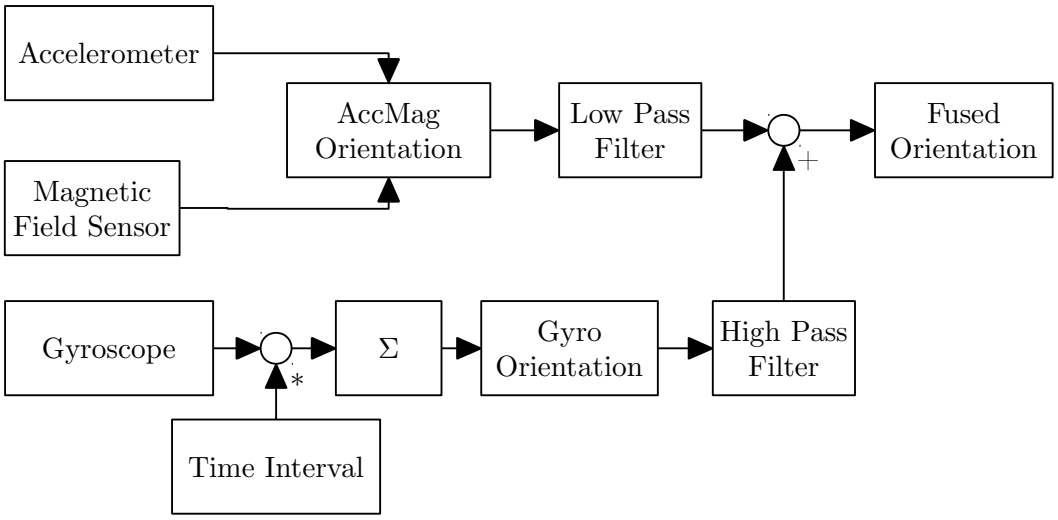


Figure 18.11.: Structure of a Sensor Fusion Model, Using Accelerometer, Magnetometer and Gyroscope for Orientation Estimation

as described in section 18.2.1, by accelerometer forces and the magnetic field vector.

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{18.12}$$

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(o_y) & sin(o_y) \\ 0 & -sin(o_y) & cos(o_y) \end{pmatrix} * \begin{pmatrix} cos(o_z) & 0 & sin(o_z) \\ 0 & 1 & 0 \\ -sin(o_z) & 0 & cos(o_z) \end{pmatrix} * \begin{pmatrix} cos(o_x) & sin(o_x) & 0 \\ -sin(o_x) & cos(o_x) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{18.13}$$

$$G = \begin{pmatrix} atan2(R_{1,2}, R_{2,2}) \\ asin(-R_{3,2}) \\ atan2(-R_{3,1}, R_{3,3}) \end{pmatrix} * G \tag{18.14}$$

After the initialization, the gyroscope sensor data $\vec{g}$ is used to compute a rotation vector $\vec{r}$. The rotation vector is transformed to a rotation matrix R, which represents the measured gyroscope rotation in the elapsed time Δt. The gyroscope orientation matrix G is updated by multiplying it with the delta rotation matrix.

The orientation $\vec{p}$ can finally be calculated, by the same trigonometric functions as for the accelerometer-magnetometer-method.

$$\vec{g'} = \frac{\vec{g}}{|\vec{g}|} \tag{18.15}$$

$$\vec{r} = \begin{pmatrix} sin(|\vec{g}| * \Delta t) * g'_x \\ sin(|\vec{g}| * \Delta t) * g'_y \\ sin(|\vec{g}| * \Delta t) * g'_z \\ cos(|\vec{g}| * \Delta t) \end{pmatrix} \tag{18.16}$$

$$R = \begin{pmatrix} 1 - 2{r_2}^2 - 2{r_3}^2 & 2r_2r_3 - 2r_3r_4 & 2r_1r_3 + 2r_2r_4 \\ 2r_1r_2 + 2r_3r_4 & 1 - 2{r_2}^2 - 2{r_3}^2 & 2r_2r + 3 - 2r_1r_4 \\ 2r_1r_3 - 2r_2r_4 & 2r_2r_3 + 2r_1r_4 & 1 - 2{r_1}^2 - 2 * {r_2}^2 \end{pmatrix} \tag{18.17}$$

$$G = G * R \tag{18.18}$$

$$\vec{p} = \begin{pmatrix} atan2(G_{1,2}, G_{2,2}) \\ asin(-G_{3,2}) \\ atan2(-G_{3,1}, G_{3,3}) \end{pmatrix} \tag{18.19}$$

The final step is to fuse the orientation determined by the gyroscope $\vec{p}$ with the orientation computed by the accelerometer and magnetometer $\vec{o}$. A factor γ defines the weighting factor of the two different orientations. Since negative orientations may occur, different cases have to be considered to translate the negative bearings into positive ones.

$$f_i = \begin{cases} \gamma * (p_i + 2\pi) + (1 - \gamma) * o_i & \text{if } p_i < \frac{-\pi}{2} \wedge o_i > 0 \\ \gamma * p_i + (1 - \gamma) * (o_i + 2\pi) & \text{if } o_i < \frac{-\pi}{2} \wedge p_i > 0 \\ \gamma * p_i + (1 - \gamma) * o_i & \text{else} \end{cases} \tag{18.20}$$

$$\vec{f} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \tag{18.21}$$

Figure 18.12 compares the raw azimuth, calculated by accelerometer and magnetometer, with an azimuth only based on the gyroscope and two fused orientations with different weightings. The orientation, calculated just by using the relative gyroscope orientation changes, tend to drift away from the real

orientation. In literature even a static gyroscope error rate is mentioned, often called *gyro drift*[66], [72]. This static drift could not be observed at the Nexus 5 and Nexus 7 device, used for testing. This suggests that gyroscopes used in today's smart phones are more accurate than the gyroscopes used in those works. The chart also shows that the fused signal is much smoother than the raw orientation without gyroscope.

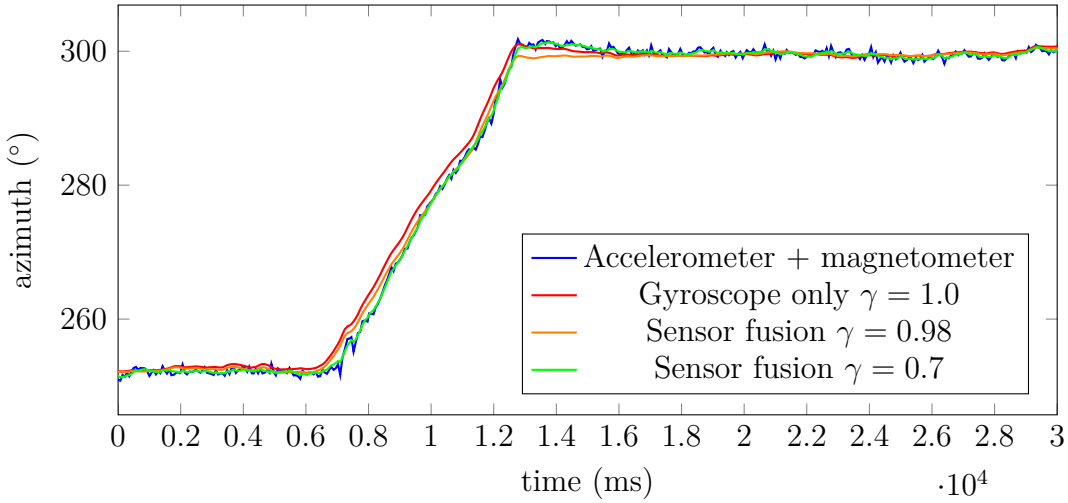


Figure 18.12.: Azimuth Computed by Fusing Data from Gyroscope, Accelerometer and Magnetometer

18.2.4. Evaluation

In order to find the best approach for orientation determination in indoor environments, various test cases have been recorded with an Android application. Three of them are presented below.

The first two test cases are artificial. They simulate a strong magnetic disturbance. In both cases, the recording Android device lays upon an empty desk, to make sure that there are as less artificial magnetic fields as possible and that there is no movement of the device. The artificial magnetic disturbance has been achieved with a little magnet near to the device. In the first test, illustrated by figure 18.13, the magnet was placed at the smart phone just before the sensor recording started. It was removed 6 seconds later. Due to the magnetic disturbance, the azimuth has been determined completely wrong as

long as the magnet was near the device. None of the filters or alternative approaches were able to reduce the disturbance. After removing the magnet, the azimuth has been calculated correctly by most filters within a short period of time. The orientation determination which only uses the gyroscope is not able to get a correct azimuth in this test because the magnetic reference orientation is wrong. Furthermore, sensor fusion with a strong weight on the gyroscope needs about 20 seconds until it reaches the correct orientation. Another interesting observation can be seen at the EMA filter. It tends to produces peaks at strong sensor changes, which is exactly the opposite of what it is intended to do.

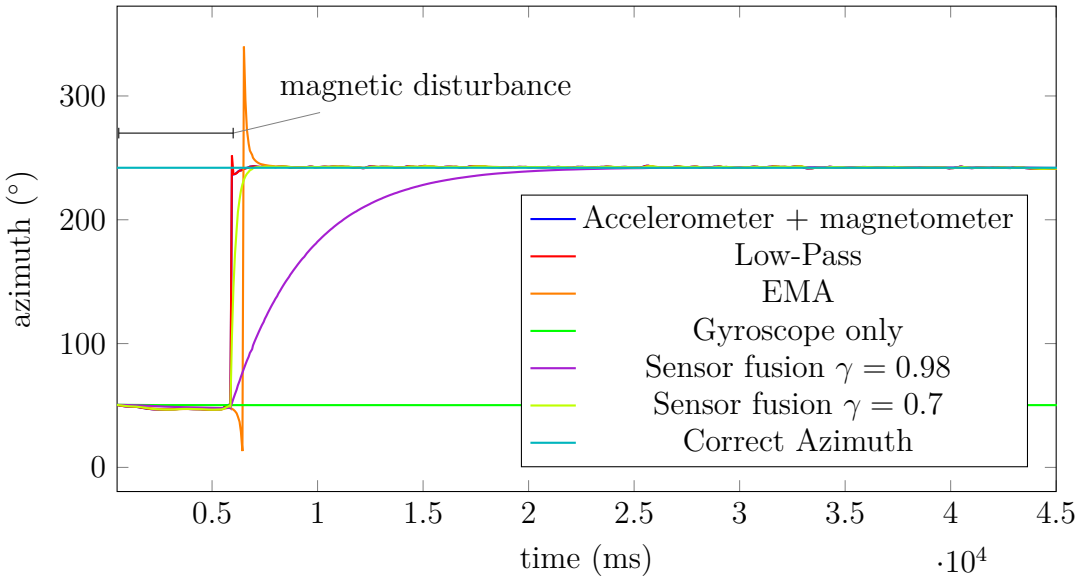


Figure 18.13.: Comparison of Compass Filters and Approaches. Determination with a Magnetic Disturbance

The second test starts with an undisturbed magnetic field. After 20 seconds, a magnet is placed near the device for about 4 seconds. As expected, the impact of the magnetic disturbance is pretty strong for magnet-based approaches. Low-pass and EMA filter are not able to reduce the magnetic disturbance significantly. Gyroscope depended approaches benefit from the additional uninfluenced sensor, and are disturbed less. The *gyroscope only* method is by far the best. The test also shows that a high weighting of the gyroscope ($\gamma = 0.98$) has a lower maximum error, but it also takes about ten seconds after the disturbance until the error is fully compensated.

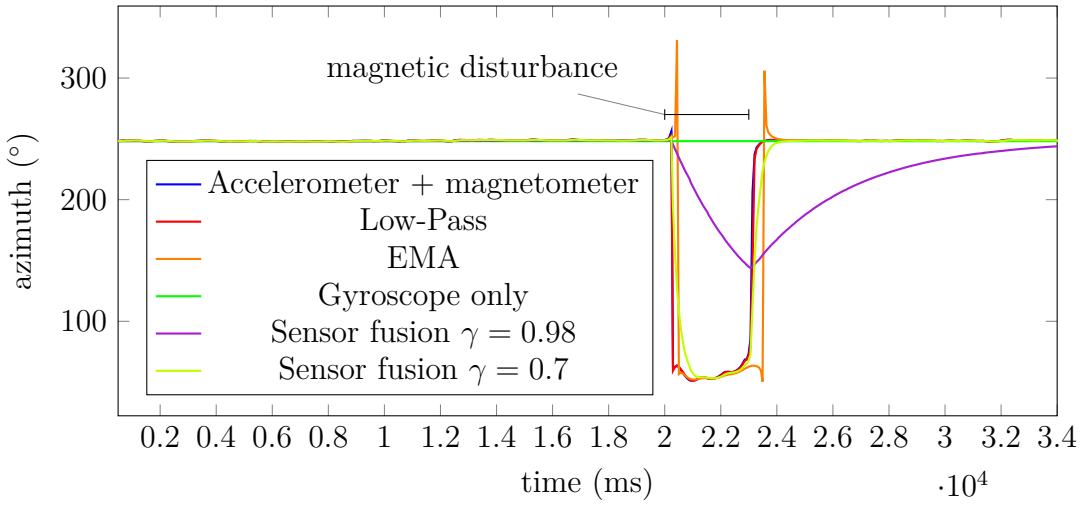


Figure 18.14.: Comparison of Compass Filters and Approaches. Magnetic Disturbance Between Second 20 and 24

The last test is a real life test, which was performed at building A20's first floor on the campus of Technische Hochschule Mittelhessen. A corridor of about 30 meters has been passed, whereas the test person turned to the right after about 23 meters. Figure 18.15 shows a pretty imbalanced orientation caused, on the one hand, by the accelerometer which is affected by device's movement and, on the other hand, by the changing magnetic disturbance within the building. The strongest magnetic disturbance occurred while passing a fire door, which caused an orientation error of about 40°.

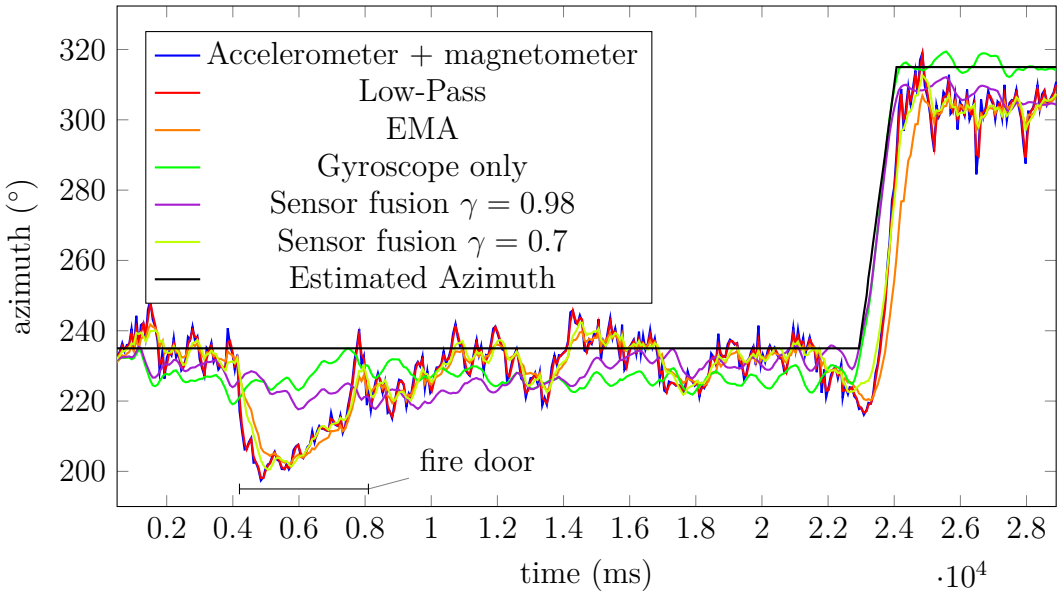


Figure 18.15.: Comparison of Compass Filters and Approaches. Walking Down a Floor and Rotate Approx. 80° After 23 Seconds

18.2.4.1. Conclusion

All things considered, the most promising method for indoor orientation is sensor fusion. Even if it takes a long period of time till huge magnetic disturbances are be compensated, it performs best with medium magnetic disturbance as one can expect it in indoor environments. Using the gyroscope without fusing it with a magnetic orientation is no option because a good magnetic reference orientation cannot be guaranteed. However, the weighting factor γ should not be too small either. As seen in the comparisons, a γ of 0.7 has nearly no advantage against non-gyroscope filters. *Sensor fusion with a γ of 0.98* seems to be a good tradeoff to get a robust compass, which is also able to compensate faulty magnetic references within a reasonable amount of time.

The numbers listed in table 18.1 confirm this conclusion. Admittedly, the non-gyroscope-based approaches perform better in the first test scenario, but the second test shows a far better maximum error of 104.23° compared to more than 196° by the other methods. Though, it should be stated that both errors are too large for an accurate dead reckoning system. The real life test, shows the real strength of sensor fusion, especially with a strong impact of the gyroscope's measurements. The average error of sensor fusion was slightly better than the

error	**unfiltered**	**low-pass**	**EMA**	**gyroscope**	**fusion (0.98)**	**fusion (0.7)**
magnetic disturbance at start						
average	24.3	24.3	26.9	191.7	37.4	24.9
avg. (during dist.)	191.8	191.9	194.0	191.6	192.8	193.4
avg. (after dist.)	0.5	0.6	3.6	191.7	15.7	1.3
maximum	195.4	195.4	228.8	191.7	194.1	195.3
magnetic disturbance in the middle						
average	16.2	16.4	17.8	0.1	15.3	16.1
avg. (before dist.)	0.5	0.5	0.45	0.1	0.3	0.48
avg. (during dist.)	189.7	190.1	180.2	0.1	60.8	179.4
avg. (after dist.)	0.9	1.32	8.12	0.1	30.7	3.35
maximum	196.8	196.8	197.6	0.1	104.23	194.87
walk down a corridor						
average	10.3	10.2	10.3	6.7	7.5	9.5
maximum	44.3	45.7	56.5	15.9	17.3	38.9

Table 18.1.: Average and Maximum Errors of all Three Test Scenarios and Filters in Degrees

magnetic azimuth's deviation - 7.5° compared to about 10°. The significant advantage of the sensor fusion approach can be seen in the maximum error which is about 30° better compared to the magnetic approaches. This 30° make a big difference in a dead reckoning use case.

19. Dead Reckoning

Everything needed for a pedestrian dead reckoning system is explained and evaluated in the last sections. The heading can be estimated using a sensor fusion compass. Movements can be detected using the step detection algorithm explained in section 18.1, and absolute reference positions are provided by the Wi-Fi positioning system described in section 17.5.

The idea of dead reckoning is fairly simple. The gaps between the absolute Wi-Fi positions are filled with relative position information, gathered by the step detection algorithm and the compass.[73],[47]

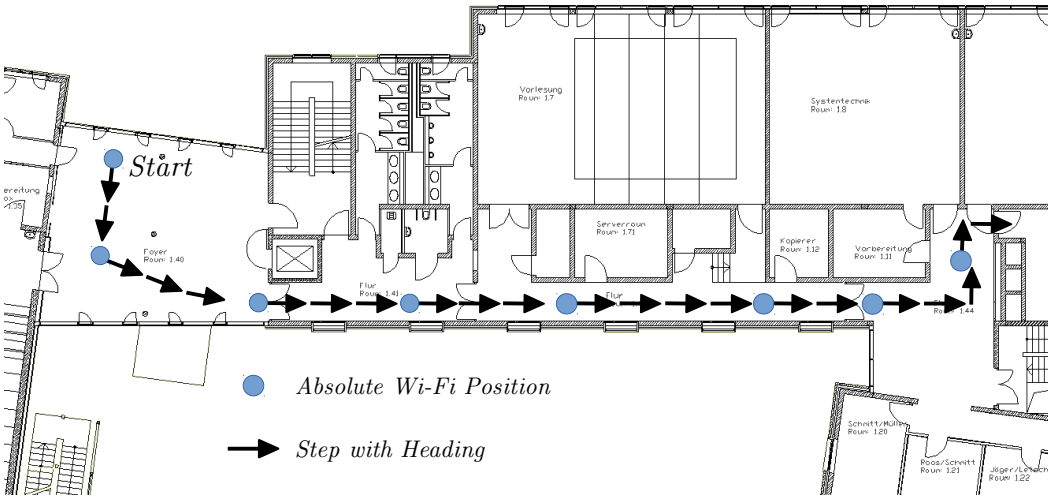


Figure 19.1.: Concept of Dead Reckoning

19.1. Weighting of Location Sources

Depending on the accuracy of the absolute location source, it might make sense to reckon the relative position in, even if an absolute location has been

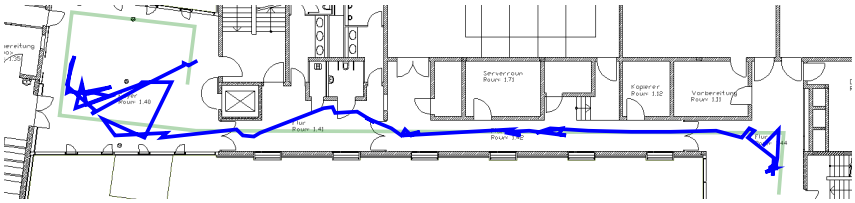
determined. This is different to usual dead reckoning systems, where the absolute position overwrites past determinations. The approach is evaluated for MoCaInfo because Wi-Fi positionings' maximum error of about 8 meters is quite inaccurate.

The easiest form to achieve a hybrid location is by using a factor α which defines the weighting of the estimated Wi-Fi position L_{WiFi} for the new position L.

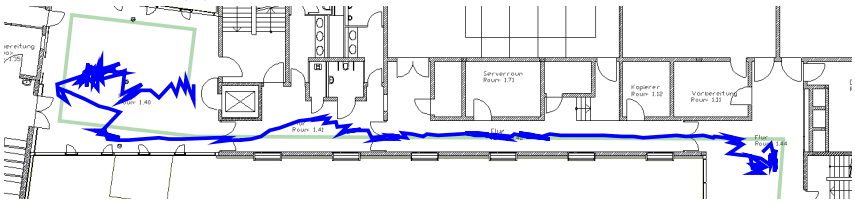
$$L = L_{WiFi} * \alpha + L_{LastRelativePosition} * (1 - \alpha) \tag{19.1}$$

Figure 19.2 shows position determinations with different values for α, whereas $\alpha = 1$ is equivalent to a Wi-Fi-only positioning system, and $\alpha = 0$ is like a completely relative positioning system. This relative approach uses the first determined Wi-Fi position as the reference position.

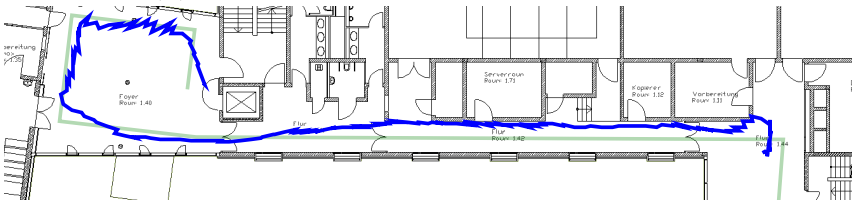
As the figure shows, the position determination can be significantly improved, especially in larger rooms like the foyer at the left side of the floor plan. On the other hand, relative positioning tends to drift sideways because of compass inaccuracies. If the influence of this drift becomes to big, the estimated position might show a nearby room instead of the correct one. This is a problem which occurs rarely when using a Wi-Fi-only positioning system because the signal attenuation caused by the walls leads to distinguishable fingerprints.



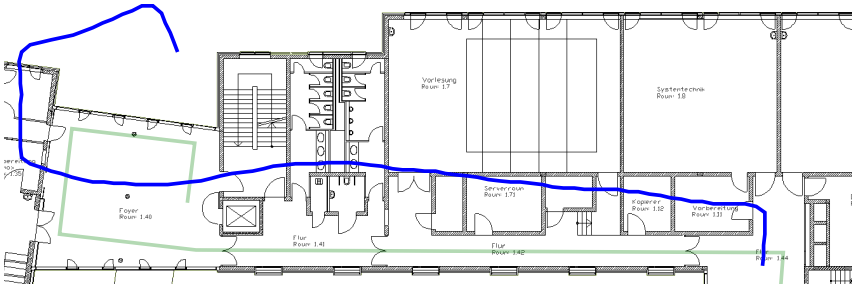
(a) *alpha* = 1, equivalent to Wi-Fi-only



(b) *alpha* = 0.5, slightly improved accuracy



(c) *alpha* = 0.9, better accuracy but drifts into corridor's wall



(d) *alpha* = 0, equivalent to relative only. Absolute reference position, determined by Wi-Fi in the lower right corner

Figure 19.2.: Comparison of Different Weightings Between Location Sources

19.2. Adaptive Weighting

In order to reduce the problem of drifting while still benefiting from relative positioning, an adaptive α has been implemented. Assuming that the absolute Wi-Fi-based position is more accurate when many access points are available for the estimation, α is calculated based on this estimated accuracy. It should be noted that this accuracy-estimation approach is pretty basic and that more sophisticated approaches exist[42]. However, for the given use case, this approach leads to reasonably good results.

Besides the number of access points, α is influenced by the time span between two Wi-Fi measurements. In some cases, e.g. if the measuring device tries to connect to a nearby Wi-Fi access point, the duration between to Wi-Fi measurements increases up to 10 seconds or even more. As the position in such long periods may have changed tremendously, the Wi-Fi position is weighted stronger than usual.

Furthermore, α depends on the step detection. If no step has been detected between two Wi-Fi measurements, the new estimated Wi-Fi position influences the final position estimation minimally. As a result, the user's position is pretty fix if he is not moving. This is a valuable improvement, compared to Wi-Fi-only positioning where the position of a not moving user tends to hop around the actual position, with every Wi-Fi measurement.

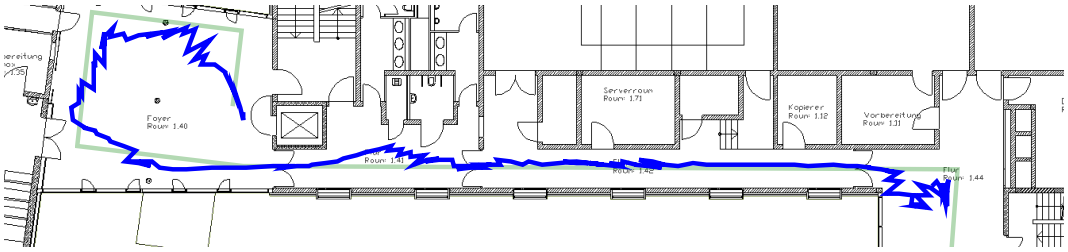


Figure 19.3.: Dead Reckoning with an Adaptive α Between 0.5 and 0.95

Finally, the box chart in figure 19.4 shows the improved accuracy compared to the Wi-Fi-only approach. At the test track, an *average accuracy of 1.67 meter* has been achieved. Furthermore, 75% of the tested reference points have an error of 2.29 meter or less which is even better than the average Wi-Fi deviation of 2.94 meters.

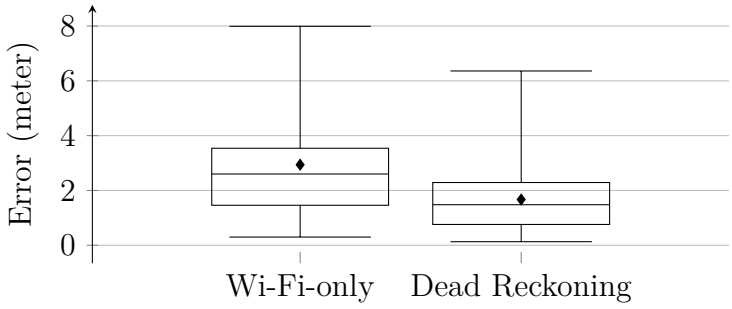


Figure 19.4.: Comparison of Wi-Fi-only Positioning and Dead Reckoning

19.3. Multiple Absolute Location Sources

Another distinctive property of this dead reckoning approach is the ability to support multiple absolute location sources. This is particularly needed, to be able to determine locations inside and outside without a noticeable interruption.

As stated above, GPS is used for outdoor positioning and Wi-Fi within buildings, or rather whenever GPS positions with reasonable accuracy are unavailable. Furthermore, NFC tags with a linked geo location can be scanned. Their highly accurate position replaces the previously estimated location.

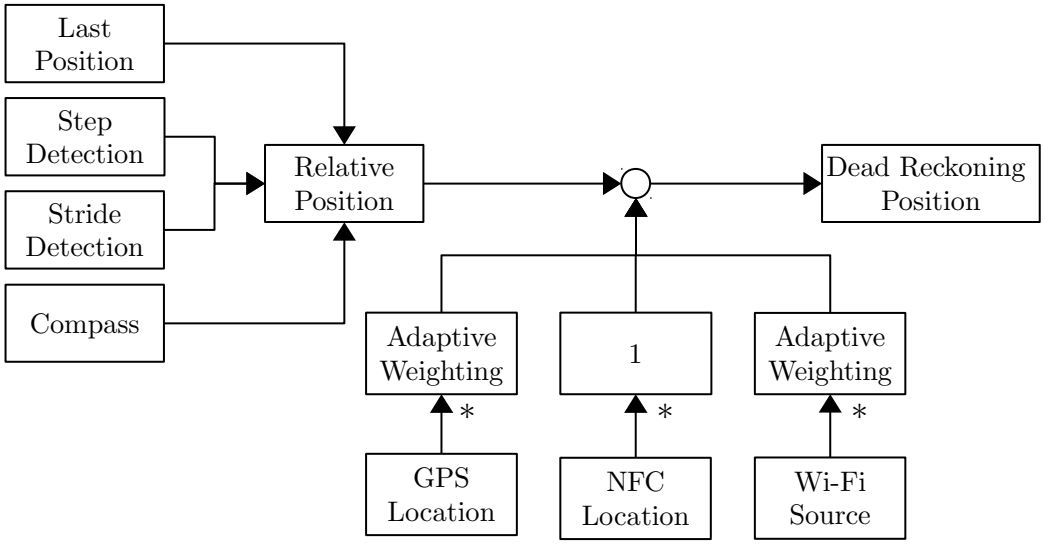


Figure 19.5.: Concept of Dead Reckoning with Multiple Absolute Location Sources

Figure 19.5 illustrates the interaction of the different location sources. Essential for a good positioning accuracy are the weighting methods, as they make sure that an inaccurate GPS position does not overwrite a good Wi-Fi position or vice versa.

As the different weightings and the usage of multiple absolute location sources in are still in a quite experimental state, the following describes the weighting in a pretty general manner, without mentioning precise equations or algorithms.

NFC Weighting is the simplest form of the used weighting methods. As the position of a scanned NFC tag is known, and the scanning device has to be within a range of a few centimeters to the NFC tag, the position is fully weighted. This means, the NFC tag's position replaces the last position completely. Therefore, the value of α is 1.

GPS Weighting is based on the GPS position's accuracy. The better the accuracy the higher it is weighted. If the GPS positions' accuracy is better than the last location's accuracy, the GPS position overwrites prior estimations. If the accuracy is worse than the last position's accuracy, it is weighted depending on the difference between the GPS' accuracy and the last position's accuracy.

Wi-Fi Weighting is currently based on the number of access points and the time between the last Wi-Fi position estimation. The less Wi-Fi access points are available for the position estimation, the less it will influence the new position. Furthermore, a long time span between two Wi-Fi scans, leads to a stronger impact of the Wi-Fi position because it has to be assumed that the old position deviates strong from the current position. A more detailed explanation can be found in section 19.2.

Part V.

Conclusions and Future Work

20. Conclusions

This thesis contains an introduction into location-based information systems, focusing on the difficulties which appear when visually impaired users, as well as sighted users, are targeted. It shows that even if more and more helping utilities for visually impaired users exist, it still is a challenging task to provide a good user experience with a mobile application for both, blind and sighted users.

Furthermore, the thesis demonstrates that indoor and outdoor positioning with smart phones is possible on condition that the infrastructure meets certain requirements. In this context, existing approaches are evaluated and combined with new ideas for further improvements.

The following sections, discuss the different topics more deeply.

20.1. Mobile Application

The mobile application's distinctiveness is the user interface. The app offers a usual graphical user interface, following well known UI design patterns and a separate low-vision user interface. It is still hardly discussed, whether dedicated low-vision user interfaces are worth the effort. According to Google, applications are accessible for everyone if certain things are considered by graphical user interface. The various accessibility tools, provided by Android shall enable visually impaired users to use those applications.[10]

According to researches, which are described more detailed in section 7.5, low-vision user interfaces are still superior to graphical user interfaces which follow universal design principles. Those finding is supported by Niehaus[13], a former project member, who is visually impaired himself. For his final paper, he interviewed several other visually impaired people and concludes that customizable

low-vision user interfaces offer a better user experience than graphical user interfaces[13, p. 69ff]. The author of this thesis therefore recommends separate low-vision user interfaces for mobile applications, in particular if visually impaired people belong to the target user group.

By a strict obedience to the MVC pattern, the additional code needed for a low-vision user interface only affects the view component. Admittedly, changes in the model or the controller often affect two view classes, which results in a higher effort for maintenance.

20.2. Navigation

This thesis shows that the current indoor navigation situation is not entirely satisfying. Google Indoor Maps just support public floor plans, which is not wanted by many institutions. Furthermore, indoor navigation is only available to a few indoor maps. The OpenStreetMap community has proposed a data structure for floor plans, but there are no libraries or applications which actually support their usage.

A combination of OpenStreetMap, its toolchain and the great rendering performance of Google Maps actually leads to a usable indoor and outdoor navigation component, with a reasonable implementation effort.

The open data structure of OpenStreetMap allows to add arbitrary information to maps. This is used to store information like the condition of the floor, in order to assist visually impaired users during navigation additionally. In so doing, MoCaInfo's navigation system explores new avenues, as such information is not considered by any other available indoor navigation system.

20.3. Indoor and Outdoor Positioning

The research in this thesis examines that indoor positioning with accuracies, reasonable for the usage in a point-to-point navigation system, is possible if the infrastructure meets certain requirements. This requirements are basically

an area-covering non-dynamically Wi-Fi infrastructure in buildings, and an offline learning phase for generating equally distributed fingerprints.

The accuracy with an average error of 2.94 meters, achieved with Wi-Fi positioning, was similar to the results of other projects[41],[42]. It was interesting to see that the quite simple probabilistic approach, *naïve Bayes*, performed way better than the distance-based approaches. Admittedly, other researches have come to similar conclusions[41], but the difference revealed in this thesis was way more significant.

With an average error of 2.94 meters and 25% of position estimations having a deviation between 3.54 and 7.99 meters, Wi-Fi positioning alone is barley usable for indoor navigation. Especially navigation for visually impaired people demands on a better and more reliable positioning accuracy.

To improve the positioning accuracy and fill the gaps between two Wi-Fi position estimations, relative positioning is used. It turned out to be more accurate as expected. Even though the FootPath[50] project demonstrates the possibility of accurate relative positioning, it only works in combination with path matching. Path matching checks the currently estimated relative position, against an assumed position. This assumption is based on the predetermined navigation path. MoCaInfo's positioning system is more flexible, as it determines positions independent from navigation paths. Therefore it can be used for navigation and for location-based information too.

In conclusion, the additional usage of relative positioning improved the accuracy significantly, up to an average error of 1.67 meters. This corresponds an improvement of more than 43%. The graphical comparison of the estimated positions in figure 19.2 19.3 and 20.1 is even more impressive.

Due to the concept of the dead reckoning system, which reckons the absolute position with the last estimated position and an adaptive factor, multiple location sources were quite easy to add. As a result, the dead reckoning system is able to use the best available location source. Currently, those absolute positions are provided by NFC, Wi-Fi or GPS. A detailed evaluation of the dead reckoning system using multiple location sources remains to be done.

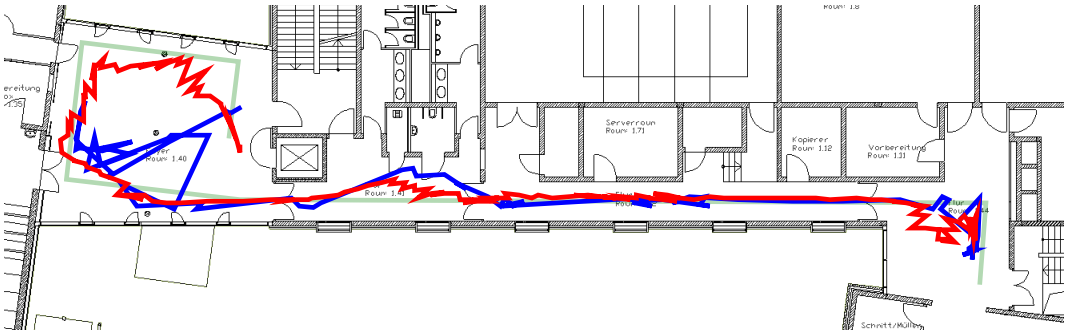


Figure 20.1.: Position Estimation with Dead Reckoning (Red) and Wi-Fi-Only (Blue). Green Line Shows the Actual Walked Track

20.3.1. Step Detection

The first examined step detection approach revealed an error of 25%. About 10% of those errors were caused by phantom steps. Phantom steps are false positive step detections, which means that a step has been detected by the algorithm but the user wasn't walking at all. Furthermore, about 15% of the steps have not been detected.

To reduce these errors, a signal preparation algorithm, which is executed before the actual step detection, has been implemented. This signal preparation is adapted from the Pan-Tompkins[65] method. By means of this algorithm, phantom steps could be reduced by one third. Furthermore, the amount of undetected steps has been improved from 14.5% to 7.4%.

Nevertheless, it should be mentioned that the current step detection strongly depends on the user, who has to hold the smart phone steadily. If the user moves the hand that holds the smart phone, steps will be detected, even if the user is not walking at all. This is a general problem when working with smart phone's acceleration measurements.

20.3.2. Compass

The thesis shows that a reliable compass, which does not get confused by magnetic interference, is not easy to achieve. The first approach, to filter the incoming magnetic and gravity data, did not yield good results. Magnetic interference still had a strong impact on the calculated azimuth.

The approach to fuse the detected rotational movements, gathered by the gyroscope, with the magnetic azimuth was way better. Magnetic interference still impacts the azimuth estimation, but the maximum deviation between the actual and the estimated azimuth is significantly lower. As a disadvantage the magnetic interference influences the azimuth estimation for a longer period of time.

21. Future Work

The following chapter describes ideas and possible approaches to further improve the different components. It starts with some thoughts about improvements of the mobile application. The following section deals with refinements for the navigation component. The chapter ends with possibilities to enhance indoor and outdoor positioning, followed by some final words.

21.1. Mobile Application

The current mobile application offers any necessary feature for a basic location-based information system, but there are still some features which might improve the user experience.

21.1.1. Global User Profiles

The first thing, which is actually less of a mobile application feature but a feature which affects multiple components, are global user profiles. Currently, the user can bookmark certain POIs or channels, but those settings are only stored at the mobile device. If the user has multiple devices or buys a new one, settings are not synchronized.

21.1.2. Notifications

Another currently missing feature is an active notification system. Such a system notifies the user when certain information is changed or added, e.g. if a lecture has been canceled. An implementation for this could be based on Google's Cloud Messaging for Android, also known as GCM[74]. GCM is a

push message service, which enables developers to send tiny messages to an Android application, whereas the message itself only contains a payload of maximum 4kb. Therefore, those GCM messages could be used to trigger the application to synchronize its data and notify the user if important events occur.

21.1.3. Machine Manuals

Niehaus[13, p. 44] proposed an interesting feature to assist visually impaired people in using the different types of automates disposed throughout the campus. This includes beverage vending machines, copying machines, automatic teller machines, etc. Most of them are barely accessible for visually impaired users. Therefore, a digital user manual for those machines would remarkably increase the independence of visually impaired users.

21.2. Navigation

The current navigation component has two major parts which remain room for further improvement. The first part is the visualization of floor plans. Currently, only shapes of the rooms are drawn. Doors are visualized as gaps in the wall. A more detailed visualization would be desirable. For instance, doors should be drawn including the opening direction and stairways should be recognizable as such. This changes are relatively easy to implement, as they can be realized by using Google Maps' polylines.

21.2.1. Navigation Instructions

The second part which could be improved are navigation instructions. The current navigation instructions are really basic and only a few navigation instructions, which might ease the orientation for visually impaired people, are implemented. The still missing instructions are those which are not directly connected to the way the user currently walks. This includes obstacles which are near the way but not directly at it, such as chairs, desks or plants. But

also other guidance notes like curbsides or drains would help visually impaired users to orientate.

Furthermore, the current navigation component lacks of an orientation assistant. Such an assistant is especially needed to help a visually impaired user finding the initial orientation, after requesting a navigation. Niehaus describes different approaches addressing that problem[13, p. 61ff]. One of it which seems to be reasonable is the clock model. Instead of having imprecise commands like *turn sharp left*, the clock model would say *turn to 7 o'clock*. This technique is widely known because it is part of blind people's rehabilitation and education program.

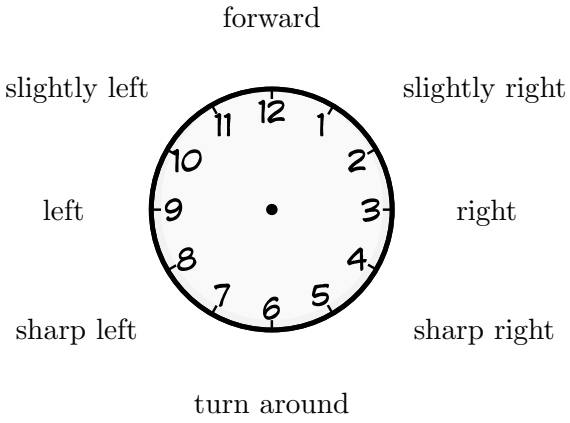


Figure 21.1.: Clock Model for Orientation

21.3. Indoor and Outdoor Positioning

Although, the current indoor and outdoor positioning system works reasonably well, there are still a lot of ideas and concepts, which might further improve the reliability and accuracy.

21.3.1. Adaptive Location Source Weighting

Section 19.1 and 19.2 show the influence of different weighting of available location sources. It can be noticed that a static weighting leads to less accurate positions than an adaptive weighting.

The currently used approach, for estimating the weighting, is pretty simple. It just uses the amount of Wi-Fi access points, regarded for the estimated Wi-Fi position, or the GPS accuracy. Especially for the adaptive weighting of Wi-Fi positions, there are more sophisticated approaches worth considering for an adaptive weighting. A starting point for further improvement may be the research by Lemelson et al. who developed different error estimation algorithms, applicable for Wi-Fi fingerprinting[42]. Together with those more sophisticated error estimation algorithms, it should be possible to choose a better Wi-Fi weighting.

21.3.2. Location-Aware Positioning

Another possibility to further improve the positioning accuracy is to consider the current location's meta information for positioning. If a user is walking down a corridor, and there is no door nearby, it is highly probable that his next position is also at the corridor and not in a nearby room.

Following this pattern, it makes sense to consider a lot of additional predictions. E.g. it is more likely that the upcoming position is in front of the last position and not behind it because changes of direction during a walk should be relatively rare. Furthermore, the user's position should never jump through walls, if there is no door nearby.

If the user is in navigation mode, those predictions can be really strict because it can be assumed that the user follows the way computed and shown by the navigation component.

21.3.3. Bluetooth Low Energy

For future indoor positioning, the usage of Bluetooth instead or in combination with Wi-Fi, may come in handy. Bluetooth is a wireless technology, similar to Wi-Fi, whereas it is designed to enable data exchanges over short distances between two devices. Bluetooth Low Energy is a power saving specification, which enables manufactures to build devices with a battery lifetime of several years or even self-powered devices with solar panels. With the adaption of

Bluetooth Low Energy into the Bluetooth 4.0 standard, more and more devices support Bluetooth Low Energy[75]. Thanks to this development, RSSI-based positioning with Bluetooth is a promising option for future indoor positioning systems.

Figure 21.2.: Gimbal Proximity Beacon Series 10 (left) and Series 20 by Qualcomm[76]

Apple Inc. even proposed an indoor positioning specification based on Bluetooth 4.0 recently. It is called *iBeacon*[34]. Besides Apple's mobile operating system iOS, Android is also supported by iBeacon. Products implementing iBeacon are currently pretty rare, but many companies have announced products for the near future. Many of those companies are startups, who have been founded to develop and sell iBeacon hardware, but also old-established firms like Qualcomm announced products. Their *Gimbal Proximity Beacon - Series 20* achieves a battery life of one to three years with four standard AA alkaline batteries. Depending on the purchasing volume, series 20 beacons are available for as little as $10 each[76],[77].

An advantage of many iBeacon devices against Wi-Fi access points is that they are battery-powered. This facilitates the equal distribution of iBeacon senders inside a building. Furthermore, the price of $10 is just a fraction of the cost of a Wi-Fi access point. As a consequence, iBeacon devices can be placed closer to each other which should result in a better positioning accuracy.

21.3.4. Step Detection

The current step detection algorithm works pretty reliable if the user holds the device steady. Movements of the smart phone which are not related to a step, lead ineluctably to false positive step detections.

It might be possible to reduce those false positive detections if the algorithm also observes the shape of the accelerometer signal, instead of just looking for changes in the signal. Thus, at least smart phone movements which differ significantly from a step movement could be detected and ignored for step detection.

21.3.4.1. Stride Detection

Another topic, which is completely ignored in this thesis is stride detection. The current implementation just uses a static stride of 70 cm, which has been estimated empirically. This solution produces large errors, especially for people significantly taller or smaller than the average.

Furthermore, test walks with a visually impaired person revealed that he makes pretty tiny steps when he is expecting an obstacle or is unfamiliar with the area. Even though a walk with one tester with low vision is not representative, it can be assumed that this is a general behavior of most visually impaired people.

Kim et al. are describing an accelerometer-based stride detection algorithm with a pretty impressive accuracy of about 96%[63]. Unfortunately, this algorithm cannot be adapted because Kim et al. are using a foot-mounted accelerometer. This sensor enables them to measure the time when the foot swings back and forth. But they also noticed a relation between the absolute gravity force and the stride length. Maybe this is an approach which can be followed for stride detection with an Android device.

Another possibility to improve stride detection could be a self-adjusting system. The system could try to adjust the stride detection, by using the distance between absolute locations and divide them by the steps walked in between.

A similar approach has been followed by Ladetto who also considers the step frequency for stride estimation[78].

21.3.5. Compass

The sensors in modern smart phones are accurate enough to achieve a good azimuth estimation, using the magnetometer and the accelerometer. Fused with the gyroscope, the azimuth is less vulnerable against magnetic interference. To further improve those results, the ratio between magnetic azimuth and the relative gyroscope rotation should be adaptive. Currently this is a fixed value. As a result, magnetic interference still effects the azimuth estimation. An adaptive approach would detect magnetic interference and ignore the magnetic azimuth completely, as long as the interference occurs.

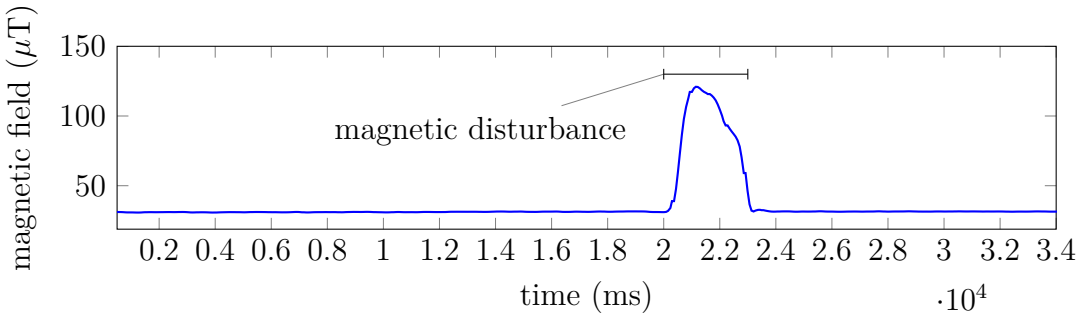


Figure 21.3.: Magnetic Field with Artificial Magnetic Interference

An approach worth examining, is the monitoring of changes in the magnetic field's strength. If the magnetic field's strength changes, it can be assumed that there is some magnetic interference. As a result the magnetic azimuth should be ignored for a certain period of time or till the field's strength is in an expected value range again. The difficulty in this case probably will be to differ between usual and artificial changes of the magnetic field. Figure 21.3 shows the magnetic field, captured during the test described in section 18.2.4 where a magnet has been placed on the smart phone after about 20 seconds. As the figure illustrates, the magnetic field changes pretty strongly. A detection of such changes should be easy.

Looking at the magnetic field in figure 21.4, which has been captured during a walk in building A20, the changes of the magnetic field are less significant.

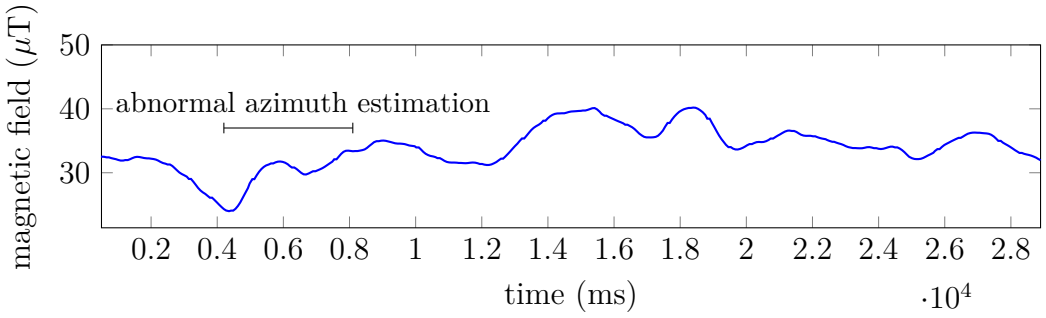


Figure 21.4.: Magnetic Field During a Walk in a Building

The according azimuth is shown in figure 18.15. The magnetic field is pretty unstable during the whole measurement, whereas the only noticeable impact on the resulting azimuth was during second 4 and 8. Therefore, it is a challenging task to distinguish interfering magnetic fields from the usual variations which may appear during a walk.

22. Final Words

This thesis, as well as other researches, shows that indoor and outdoor navigation and positioning is still a challenging topic. Even though, the capabilities of smart phones have been improved dramatically over the past few years, the approaches used for indoor positioning are still based on thoughts made 14 years ago[39], a time were smart phones did not even exist.

Also interesting is the fact that there seems to be little effort in developing a reliable smart phone compass for indoor purposes. An approach, which weights gyroscope and magnetic compass, based on assumed magnetic interference, seems to be obvious. Nevertheless, no research paper or product has been found which claims to use such a method.

In the near future, advancements in indoor positioning are expectable. Thanks to Apple's iBeacon specification, accurate indoor positioning will probably find its way out of the university environment into real-world applications. On the basis of cheap, standardized, battery-powered iBeacon senders, accurate indoor positioning systems are possible for the masses. Sooner or later, iBeacon is likely to be used in many public places such as shopping centers, train stations or airports. The main usage will probably be location-aware advertising, but this infrastructure also enables more useful usages, like indoor navigation.

With spread of an indoor positioning system like iBeacon, existing indoor maps and navigation will likely be improved too. Nevertheless, the project described in this thesis will still have eligibility as it examines a perspective which is usually ignored: The perspective of visually impaired users.

A. Bibliography

[1] Dean Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Pearson Education, 2010. ISBN: 978-0321635846.

[2] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004. ISBN: 0-321-12521-5.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0321127420.

[4] Frank Buschmann et al. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Chichester, UK: Wiley, 1996. ISBN: 978-0-471-95869-7.

[5] Gartner, Inc. *Market Share Analysis: Mobile Phones, Worldwide, 4Q13 and 2013 Summary*. accessed: February, 23rd 2013. URL: `https://www.gartner.com/newsroom/id/2665715`.

[6] Wiebe Elsinga. *Will HTML5 kill native apps?* accessed: February, 23rd 2014. Oct. 2011. URL: `http://wiebe-elsinga.com/blog/will-html5-kill-native-apps/`.

[7] Patrick Winter. "Entwicklung eines Campusinformationssystems". Bachelor Thesis. 35390 Gießen, Germany: Technische Hochschule Mittelhessen, Aug. 2011.

[8] SQLite. *SQLite Frequently Asked Questions*. accessed: July, 25th 2013. URL: `http://www.sqlite.org/faq.html`.

[9] Yifeng Chen. "Programmable Verifiers in Imperative Programming". In: *Unifying Theories of Programming: Third International Symposium*. 2010.

[10] Google Inc. *Accessibility*. accessed: July, 26th 2013. URL: http://developer.android.com/design/patterns/accessibility.html.

[11] Molly Follette Story, James L. Mueller, and Ronald L. Mace. *The Universal Design File - Designing for People of All Ages and Abilities*. NC State University, The Center for Universal Design, 1998.

[12] Javier Sánchez Sierra and Joaquín Selva Roca de Togores. "Designing Mobile Apps for Visually Impaired and Blind Users". In: *ACHI 2012 : The Fifth International Conference on Advances in Computer-Human Interactions*. 2012. ISBN: 978-1-61208-177-9.

[13] Christoph Niehaus. "Konzeption eines für blinde und sehbehinderte Nutzer geeigneten mobilen Informations- und Navigationssytems". Diploma Thesis. 35390 Gießen, Germany: Technische Hochschule Mittelhessen, Feb. 2012.

[14] Google Inc. *Making Applications Accessible*. accessed: July, 29nd 2013. URL: http://developer.android.com/guide/topics/ui/accessibility/apps.html.

[15] Sonia Sharma. *Designing Accessible Android Applications*. presented at Droidcon, Berlin. 2013.

[16] Artur Klos. "Near Field Communication und dessen Einsatz im mobilen Campusinformationssystem". Bachelor Thesis. 35390 Gießen, Germany: Technische Hochschule Mittelhessen, Aug. 2011.

[17] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co. Inc., 1995.

[18] Google Inc. *Patterns*. accessed: July, 29nd 2013. URL: http://developer.android.com/design/patterns/index.html.

[19] Juhani Lehtimaki. *Smashing Android UI*. Smashing Magazine Book Series. John Wiley & Sons, Inc., 2012. ISBN: 9781118387337.

[20] Google Inc. *Navigation Drawer*. accessed: February, 2nd 2014. URL: http://developer.android.com/guide/topics/connectivity/nfc/nfc.html.

[21] Statistisches Bundesamt. *Statistik der schwerbehinderten Menschen 2011*. Statistisches Bundesamt, Wiesbaden, 2013.

[22] Google Inc. *What is indoor Google Maps?* accessed: November, 26th 2013. 2013. URL: https://maps.google.com/help/maps/indoormaps/.

[23] OpenStreetMap Foundation. *OpenStreetMap stats report*. accessed: January, 28th 2013. URL: http://www.openstreetmap.org/stats/data_stats.html.

[24] OpenStreetMap Foundation. *IndoorOSM*. accessed: January, 28th 2013. URL: http://wiki.openstreetmap.org/wiki/IndoorOSM.

[25] OpenStreetMap Foundation. *Elements*. accessed: January, 28th 2013. URL: http://wiki.openstreetmap.org/wiki/Elements.

[26] OpenStreetMap Foundation. *JOSM*. accessed: January, 28th 2013. URL: http://josm.openstreetmap.de/.

[27] Per Enge and Pratap Misra. "Special Issue on Global Positioning System". In: *Proceedings of the IEEE, Vol. 87*. 1999.

[28] Federal Aviation Administration. *Civil Report Card On GPS Performance August 2013*. Aug. 2013.

[29] GSM Association and A.T. Kearney. *European Mobile Industry Observatory 2011*. Nov. 2011.

[30] Tomislav Kos, Mislav Grgic, and Gordan Sisul. "Mobile User Positioning in GSM/UMTS Cellular Networks". In: *48th International Symposium ELMAR-2006, Zadar, Croatia*. June 2006.

[31] Mohamed Ibrahim and Moustafa Youssef. "CellSense: A Probabilistic RSSI-based GSM Positioning System". In: *Proceedings of the Global Communications Conference*. 2010.

[32] Wi-Fi Alliance. *The Wi-Fi Brand*. accessed: December, 23rd 2013. URL: http://www.wi-fi.org/about/wi-fi-brand.

[33] Mozilla Foundation. *Mozilla Location Service - Overview*. accessed: December, 23rd 2013. URL: https://location.services.mozilla.com/.

[34] Apple Inc. *iOS 7: Understanding Location Services*. accessed: December, 23rd 2013. 2013. URL: http://support.apple.com/kb/HT5594.

[35] Cisco Systems, Inc. *Wi-Fi Location-Based Services 4.1 Design Guide*. May 2008.

[36] Kensaku Kawauchi, Takashi Miyaki, and Jun Rekimoto. “Directional Beaconing: A Robust WiFi Positioning Method Using Angle-of-Emission Information”. In: *Location and Context Awareness, 4th International Symposium, LoCA 2009*. 2009.

[37] Marc Ciurana, David López, and Francisco Barceló-Arroyo. “SofTOA: Software Ranging for TOA-Based Positioning of WLAN Terminals”. In: *Location and Context Awareness, 4th International Symposium, LoCA 2009*. 2009.

[38] Moustafa Youssef and Ashok Agrawala. “The Horus WLAN Location Determination System”. In: *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. MobiSys ’05. 2005, pp. 205–218.

[39] Paramvir Bahl and Venkata N. Padmanabhan. *RADAR: An In-Building RF-based User Location and Tracking System*. Tech. rep. Microsoft Research, 2000.

[40] Yongguang Chen and Hisashi Kobayashi. *Signal Strength Based Indoor Geolocation*. Tech. rep. Princeton University, 2002.

[41] Ville Honkavirta, Tommi Peraläand Robert Ali-Löytty, and Robert Piché. “A Comparative Survey of WLAN Location Fingerprinting Methods”. In: *Processdings of the 6th Workgroup on Positioning, Navigation and Communication 2009*. 2009.

[42] Hendrik Lemelson et al. “Error Estimation for Indoor 802.11 Location Fingerprinting”. In: *Location and Context Awareness*. 2009.

[43] Rainer Mautz and Sebastian Tilch. “Survey of Optical Indoor Positioning Systems”. MA thesis. ETH Zürich, 2011.

[44] Harlan Hile and Gaetano Borriello. “Positioning and Orientation in Indoor Environments Using Camera Phones”. In: 28.4 (Aug. 2008), pp. 32–39.

[45] Sebastian Tilch. “CLIPS - Development of a Novel Camera and Laser-Based Indoor Positioning System”. PhD thesis. ETH Zürich, 2012.

[46] Peter Brída. “Location Technologies For GSM”. In: *TRANSCOM*. 2003, pp. 119–122.

[47] Winfried Böhm. *Handbuch der Navigation. Begriffe, Formeln, Verfahren, Schemata*. Busse-Seewald Verlag, 1978, p. 246.

[48] Anthony Mandow et al. "Experimental kinematics for wheeled skid-steer mobile robots". In: *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego, CA, USA, Oct 29 - Nov 2, 2007*. 2007.

[49] Andrea Bonarini, Matteo Matteucci, and Marcello Restelli. "A Kinematic-independent Dead-reckoning Sensor for Indoor Mobile Robotics". In: *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Politecnico di Milano. 2004.

[50] Jó Ágila Bitsch Link et al. "FootPath: Accurate map-based indoor navigation using smartphones". In: *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. 2011.

[51] Stéphane Beauregard and Harald Haas. "Pedestrian Dead Reckoning: A Basis for Personal Positioning". In: *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC'06)*. 2006.

[52] Christopher Drane, Malcolm Macnaughtan, and Craig Scott. "Positioning GSM Telephones". In: *IEEE Communications Magazine*. Apr. 1998.

[53] Maximilian von Piechowski. "Evaluation von Technologien und Verfahren zur hybriden Indoor-Lokalisierung für Android basierte mobile Geräte". Bachelor Thesis. 35390 Gießen, Germany: Technische Hochschule Mittelhessen, Mar. 2012.

[54] Verwaltungsgericht Köln. *Urteil vom 17. Juli 2013 - Az. 21 K 2589/12*. July 2013.

[55] Google Inc. *NFC Basics*. accessed: July, 22nd 2013. URL: `http://developer.android.com/guide/topics/connectivity/nfc/nfc.html`.

[56] Thomas King et al. "COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses". In: *Proc. of the First ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*. 2006.

[57] Antonio del Corte-Valiente, Jose Manuel Gómez-Pulido, and Oscar Gutiérrez-Blanco. "Efficient Techniques and Algorithms for Improving Indoor Localization Precision on WLAN Networks Applications". In: *Int. J. Communications, Network and System Sciences*. 2009.

[58] Julia Letchner, Dieter Fox, and Anthony LaMarca. "Large-Scale Localization from Wireless Signal Strength". In: *American Association for Artificial Intelligence*. 2005.

[59] Anja Bekkelien. "Bluetooth Indoor Positioning". MA thesis. University of Geneva, 2012.

[60] Moustafa Amin Abdel Azim Yousief Abdel Rehim. "HORUS: A WLAN-BASED INDOOR LOCATION DETERMINATION SYSTEM". PhD thesis. University of Maryland, 2004.

[61] Douglas M. Hawkins. "The Problem of Overfitting". In: *J. Chem. Inf. Comput. Sci.* 44 (2004), pp. 1–12.

[62] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, 2005, pp. 96–106.

[63] Jeong Won Kim et al. "A Step, Stride and Heading Determination for the Pedestrian Navigation System". In: *Journal of Global Positioning Systems* 3 (2004), pp. 273–279.

[64] Oliver J. Woodman. "Pedestrian localisation for indoor environments". PhD thesis. University of Cambridge, 2010.

[65] H. Ying et al. "Automatic Step Detection in the Accelerometer Signal". In: *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*. 2007.

[66] Ubejd Shala and Angel Rodriguez. *Indoor Positioning using Sensorfusion in Android Devices*. Sept. 2011.

[67] Y Sun, M P Wu, and X P Hu. "A NewSolution Algorithm of Magnetic Azimuth". In: *Journal of Physics: Conference Series* 48 (2006), pp. 111–116.

[68] Google Inc. *SensorEvent*. accessed: July, 29nd 2013. URL: `https://developer.android.com/reference/android/hardware/SensorEvent.html`.

[69] Mary Natrella. *NIST/SEMATECH e-Handbook of Statistical Methods*. Ed. by Charline Cleraux et al. NIST/SEMATECH, Oct. 2013. URL: http://www.itl.nist.gov/div898/handbook/index.htm.

[70] Shahid Ayub, Alireza Bahraminisaab, and Bahram Honary. "A Sensor Fusion Method for Smart phone Orientation Estimation". In: *13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*. 2012.

[71] Paul Lawitzki. "Application of Dynamic Binaural Signals in Acoustic Games". MA thesis. Stuttgart Media University, Mar. 2012.

[72] Jeffrey R. Blum, Daniel G. Greencorn, and Jeremy R. Cooperstock. "Smartphone sensor reliability for augmented reality applications". MA thesis. McGill University, Montréal, Québec, Canada, 2011.

[73] Yasutaka Fuke and Eric Krotkov. "Dead Reckoning for a Lunar Rover on Uneven Terrain". In: *Proceedings of the 1996 IEEE International Conference on Robotics and Automation Minneapolis, Minnesota*. Apr. 1996, pp. 411–413.

[74] Google Inc. *Google Cloud Messaging for Android*. accessed: January, 29th 2013. URL: https://developer.android.com/google/gcm/index.html.

[75] Bluetooth SIG, Inc. *Bluetooth Smart Technology: Powering the Internet of Things*. accessed: January, 31st 2014. 2013. URL: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx.

[76] Qualcomm Inc. *Gimbal: Context Awareness and Proximity for Highly Relevant Consumer Engagements*. accessed: January, 31st 2014. 2013. URL: http://www.qualcomm.com/solutions/gimbal.

[77] Qualcomm Inc. *Qualcomm Announces Availability of its Gimbal Proximity Beacons to Enable Customer Engagement Based on Micro Location*. accessed: January, 31st 2014. Dec. 2013. URL: http://www.qualcomm.com/solutions/gimbal.

[78] Quentin Ladetto. "On foot navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering". In: *Proceedings of the 13th International Technical Meeting of the*

Satellite Division of The Institute of Navigation. ION GPS 2000. 2000, pp. 1735–1740.

B. Glossary

Activity *(Android Term)* is a single, focused thing that the user in an Android application can do. Usually Activity classes take care of creating a window which holds the user interface. Therefore, the term Activity is often used equivalent to graphical user interface

Activity Chooser *Android Term* enables the user to choose an Activity which process a given Intent. Will be launched if the description of an Intent fits to multiple Activities

Android is an operating system based on Linux, targeting mobile, touchscreen devices like smart phones or tablet computers. Initially developed by the Android, Inc. and bought by Google Inc. in 2005

Android Support Library is a set of code libraries that provide features from newer Android API Levels to older API Levels. Due to the library, developers are able to use features like *ActionBar* which have been introduced in Android 3.0 on devices running Android 2.1

API level *Android Term* describes the version of Android's official application programming interface. Current API level of Android is 19, as of March 11th, 2014. API Level 19 is provided by Android 4.4

Beacon is a management frame in the IEEE 802.11 Wi-Fi specification. It is sent periodically to announce the presence of a Wi-Fi access point. It contains information about the SSID, BSSID and other connectivity information. It is also used to determine the received signal strength indication (RSSI).

BSSID short form of basic service set identification. It is a unique identifier of a wireless access point, equivalent to its MAC address.

dp *Android Term* density-based pixel. A unit to describe UI elements in Android. It is density-based which means that it scales with the density of the screen.

hybrid application a web-based mobile application which has been packaged to a native application by using the native plattform's SDK. The actual application is written with web technologies, usually HTML5, JavaScript and CSS. Third party frameworks like PhoneGap allow a hybrid application to access native SDK features via JavaScript

Intent *Android Term* an abstract description of an operation to be performed in Android. Often used to start an Activity or communicate between different components of an Android application or even between different Android applications

MySQL is a widely used open-source relational database management system. Today it is owned by Oracle Corporation

Push Message describes messages which are pushed to a device. This means that the publisher of the message initiates the communication to the client. It is the opposite to pull messages, where the request for transmission of information is initiated by the receiver.

REST API REST is the short form for representational state transfer. It was initially described in the context of HTTP. The main principle of REST are stateless messages between client and server. Therefore, any message has to send a representation of the current state to enable the receiver to process the message correctly. Today's REST APIs are usually adressable via HTTP and use JSON or XML for data exchange.

RFID short form of radio-frequency identification. Describes the wireless non-contact use of radio-frequency electromagnetic fields to transfer data. The data usually is used to automatically identify or track objects which have been marked with an RFID tag

self-voicing applications provide an aural interface without the need of a screen reader application. . . . . .

SQLite is a relational database management system. In contrast to other database management systems, it does not run in a separate process. Therefore, it is used embedded in applications. . . . . .

SSID shortform of service set identifier. It is a 1 to 32 byte string to describe a wireless access point in a human-readable form. . . . . .

UMBmark is the short form for University of Michigan Benchmark. It is a method to measure and compare robot positioning approaches. . . . . .